Etude et Réalisation Régulation du courant moteur du Kart

Giovannangeli Julien
Lanoë Michaël
Groupe TP : S2

Le 26 mars 2007

Introduction

Sommaire

Présentation du projet.

- But du projet.
- Cahier des charges.

Partie Electronique/Automatique.

- Offset et réglage de la pédale de l'accélérateur.
- Identification du moteur.

Programmation.

- Objectif et mise en place la programmation
- Le Convertisseur Analogique Numérique (CAN).
- Routine d'interruption.
- Le PWM.
- Relation PC-Moteur.

Présentation du projet.

- **But du projet**

  ➔ Réguler le courant moteur du karting

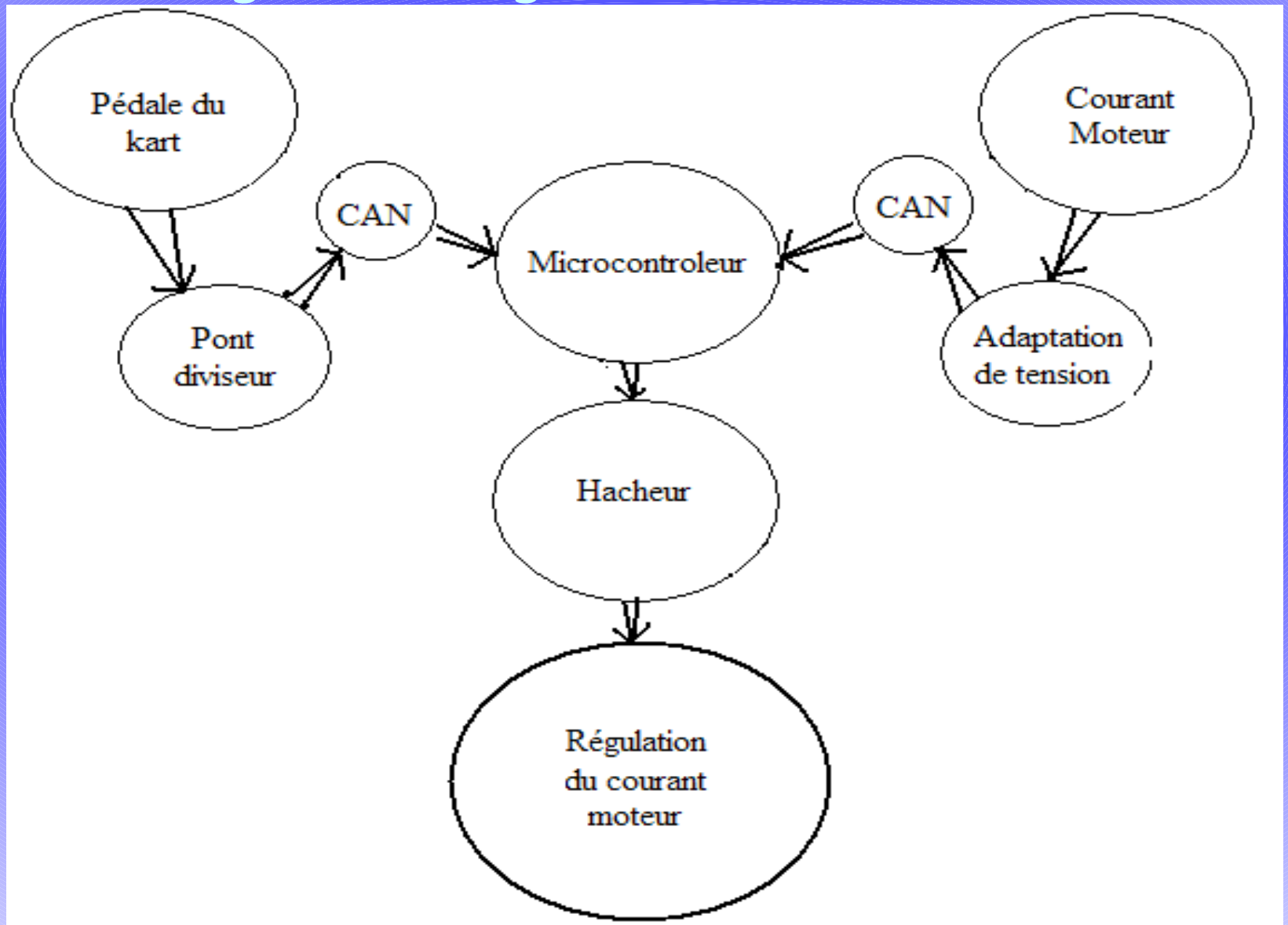
- **Cahier des charges**

  ➔ **Capteur de courant :**
  - Tension de sortie : -4V <Vs< +4V.
  - Il faut adapter cette tension à celle du microcontroleur (0/+5V).

  ➔ **Microcontroleur :**
  - ATmega8535.
  - Tension d'entrée 0/+5V.
  - Utilisation de CodeVision AVR Studio.
  - Utilisation de fonctions: CAN, PWM, routine d'interruption.

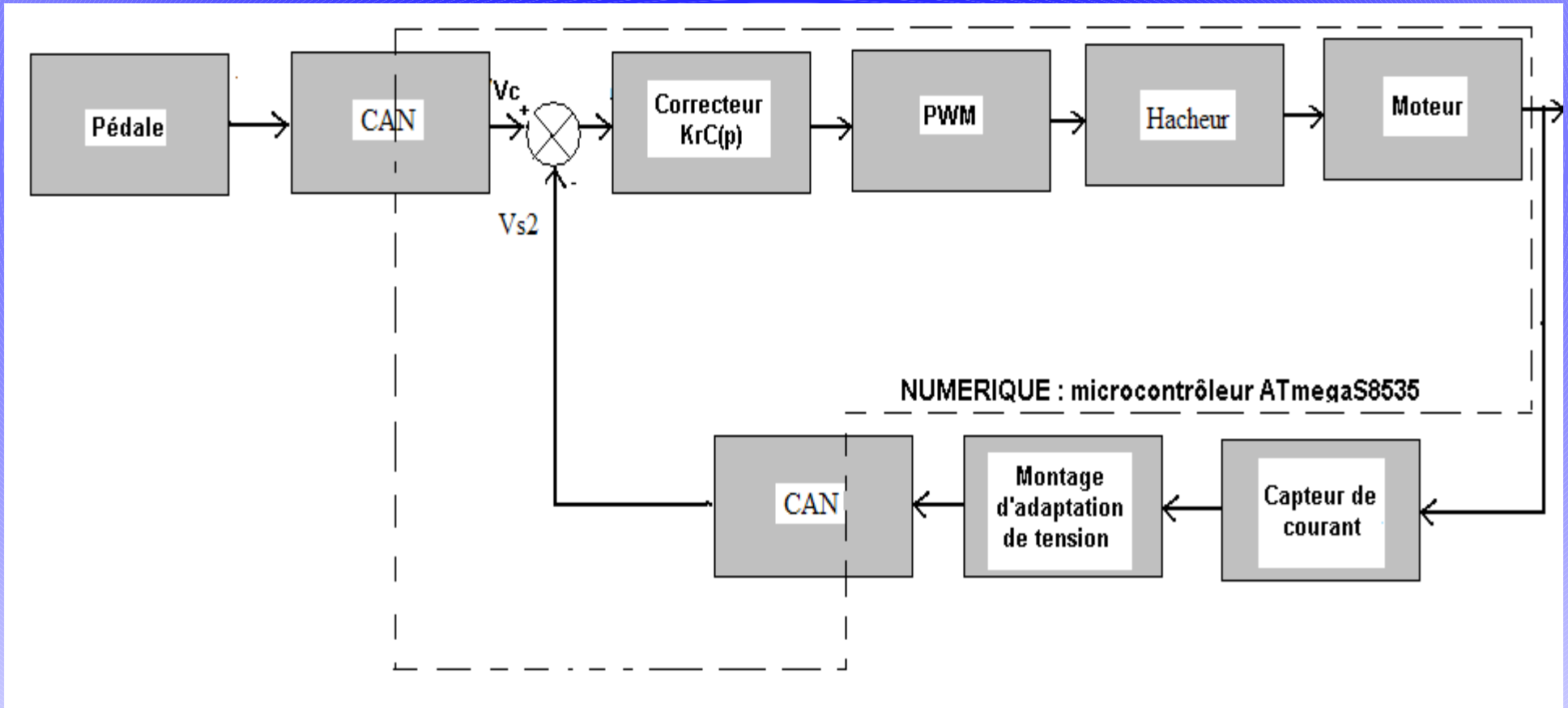
Présentation du projet.

- Cahier des charges
  - ➔Diagramme sagittale



Présentation du projet.

- Cahier des charges
  - ➔Synoptique de la régulation du courant moteur



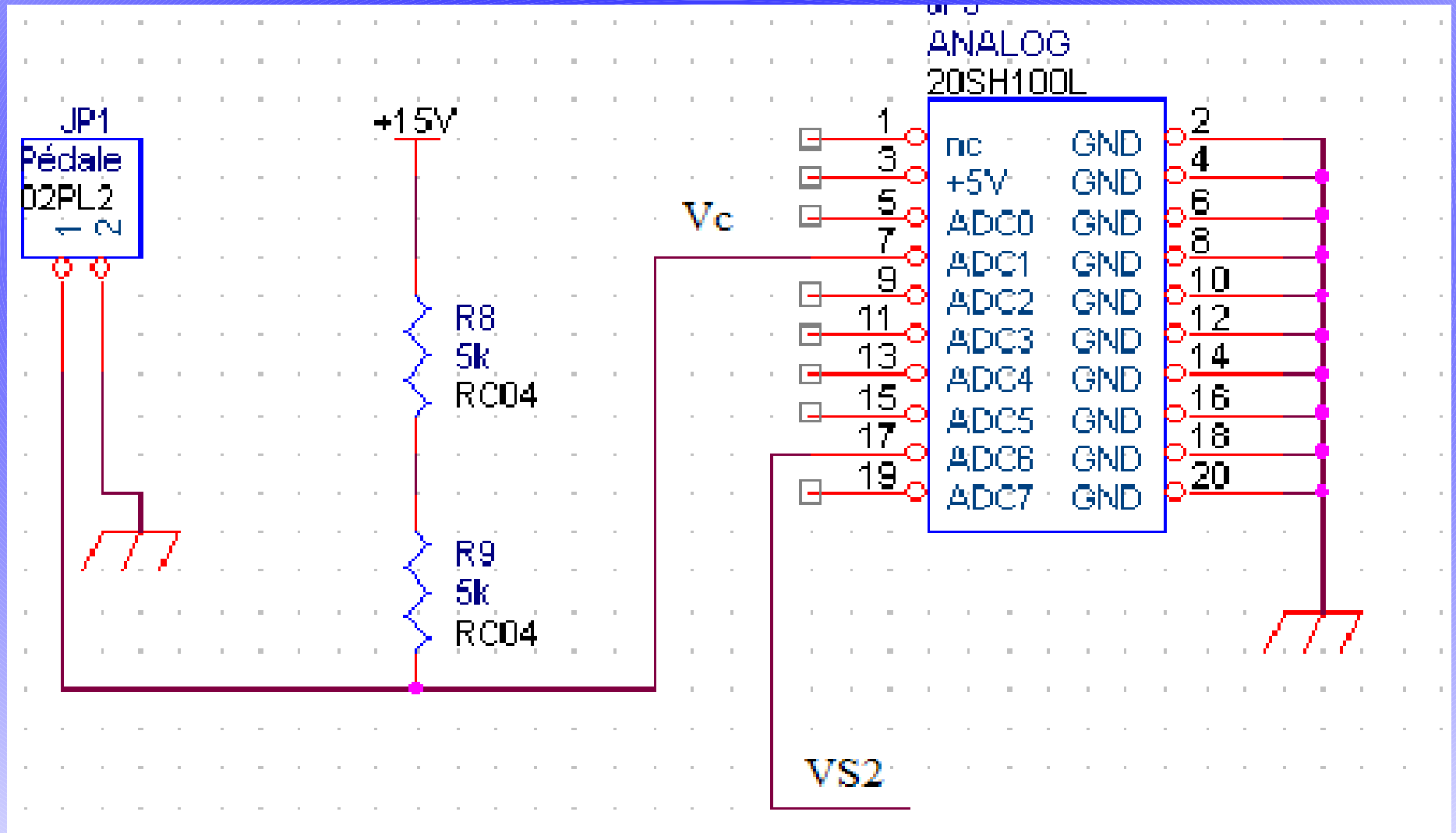
Partie Electronique/Automatique.

La pédale du Kart

Partie Electronique/Automatique.

La pédale du Kart

La pédale est reliée à une résistance variable de 5kΩ



Partie Electronique/Automatique.

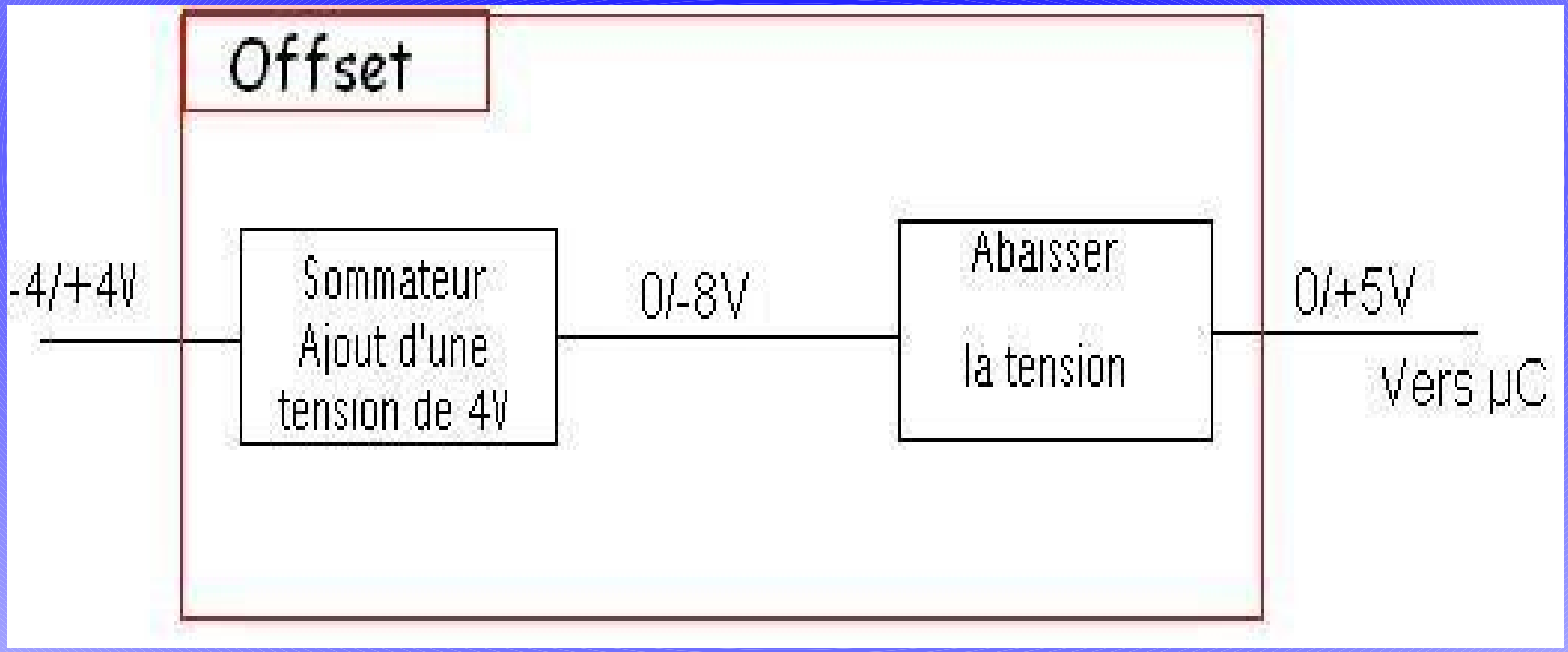
Le capteur de courant

Partie Electronique/Automatique.

Le capteur de courant

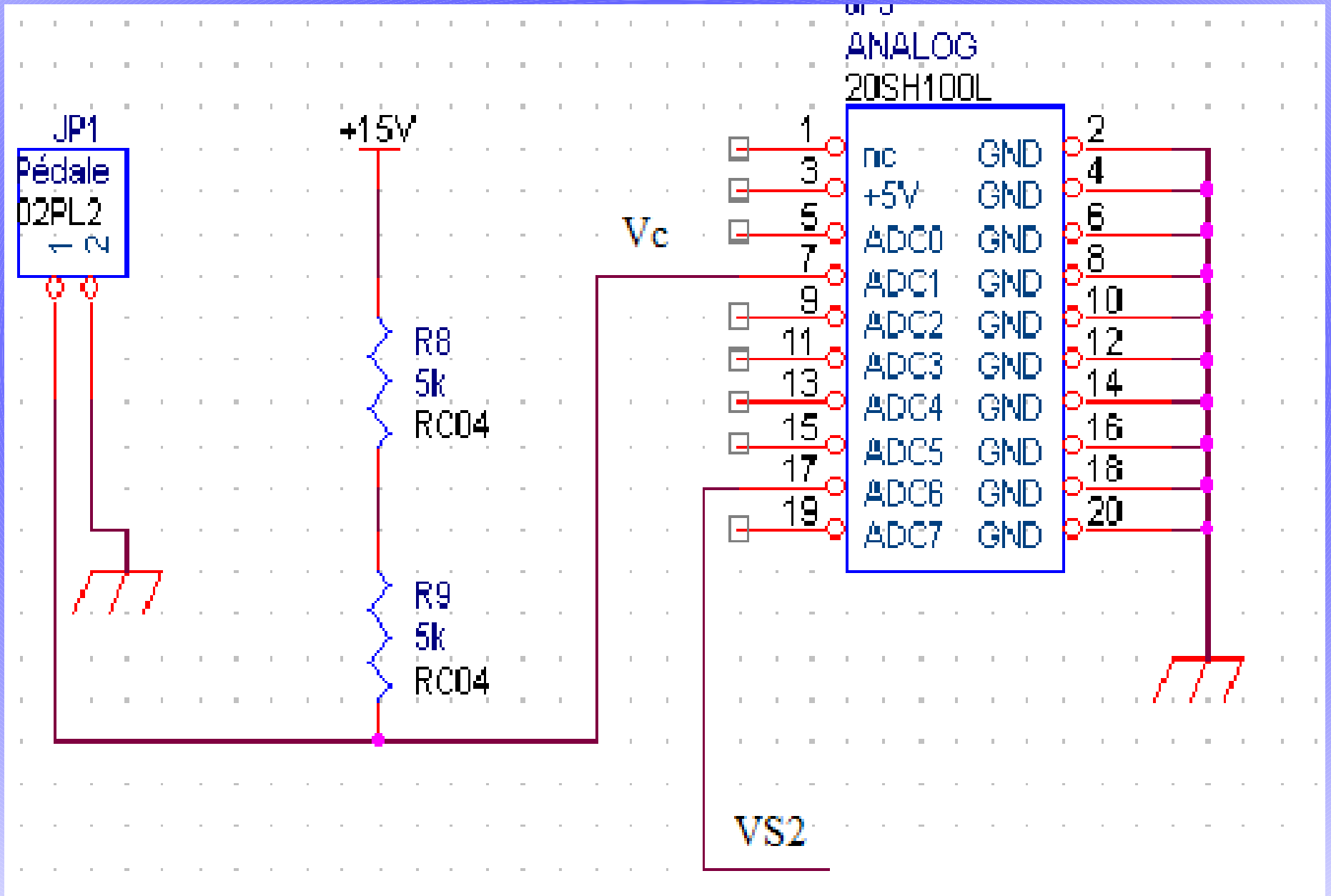
Le capteur de courant délivre une tension entre -4V et +4V alors que le microcontroleur supporte du 0/+5V d'où un montage d'adaptation de tension

- **OFFSET +Abaisseur de tension.**



Partie Electronique/Automatique.

Le capteur de courant

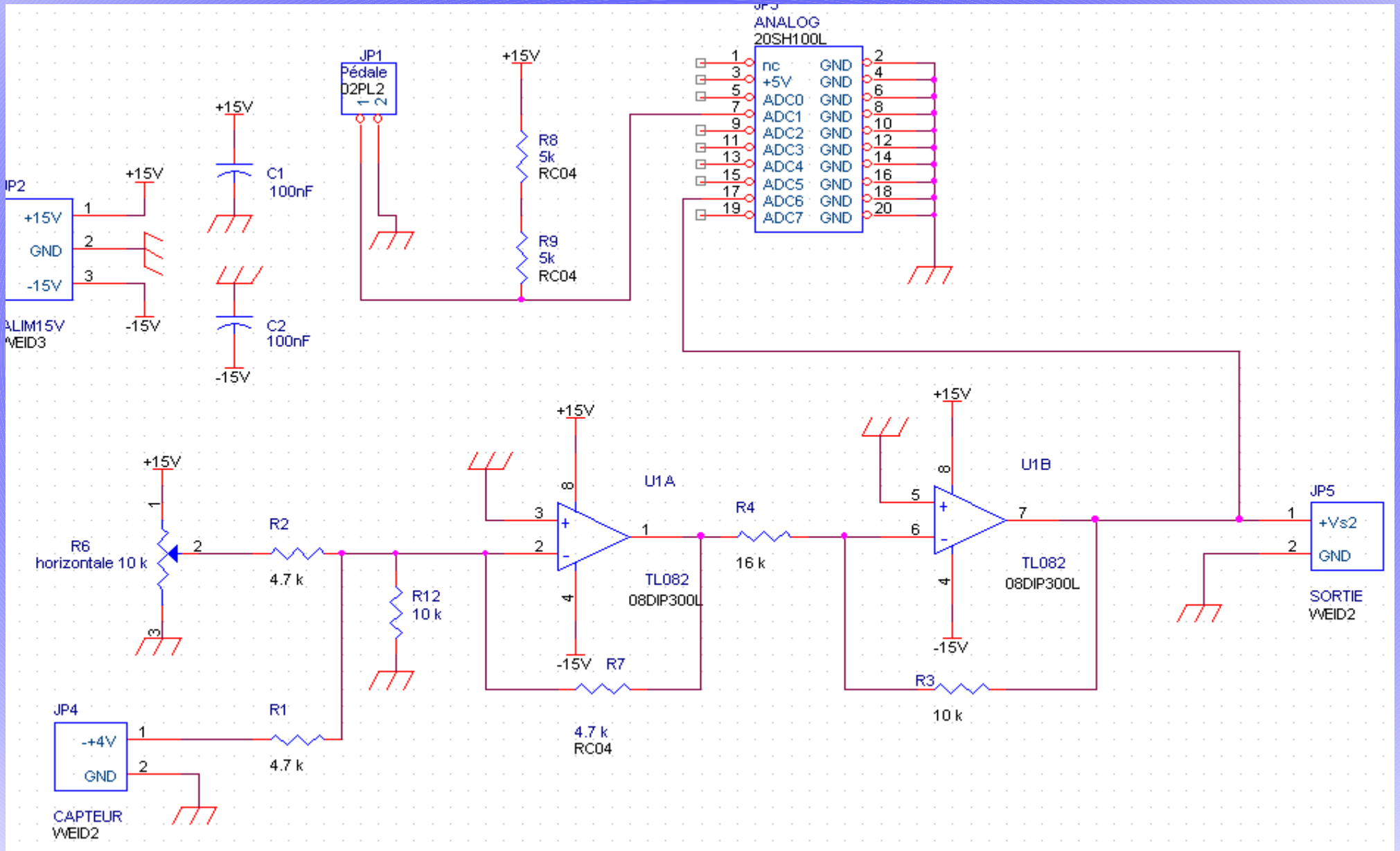


Partie Electronique/Automatique.

Schéma final : pédale + montage éléctronique

Partie Electronique/Automatique.

Schéma final



Partie Electronique/Automatique.

Aspect automatique du projet: Identification du moteur

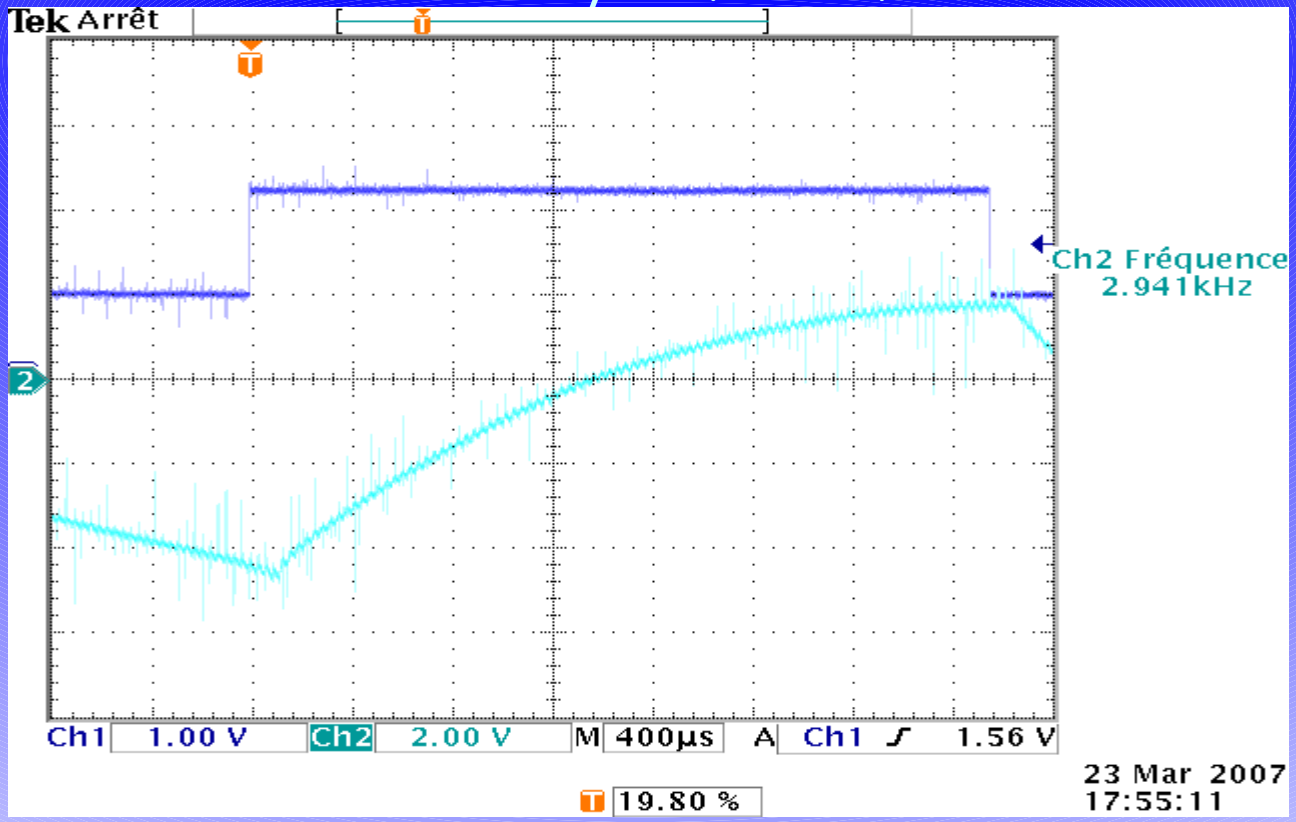
Partie Electronique/Automatique.

Aspect automatique

Pour identifier le système on a plusieurs méthodes d'identification:

- Broïda
- Strejc
- Zeigler & Nichols
- ...

A partir du relevé de la sortie du système soumit à un échelon de tension:



Programmation.

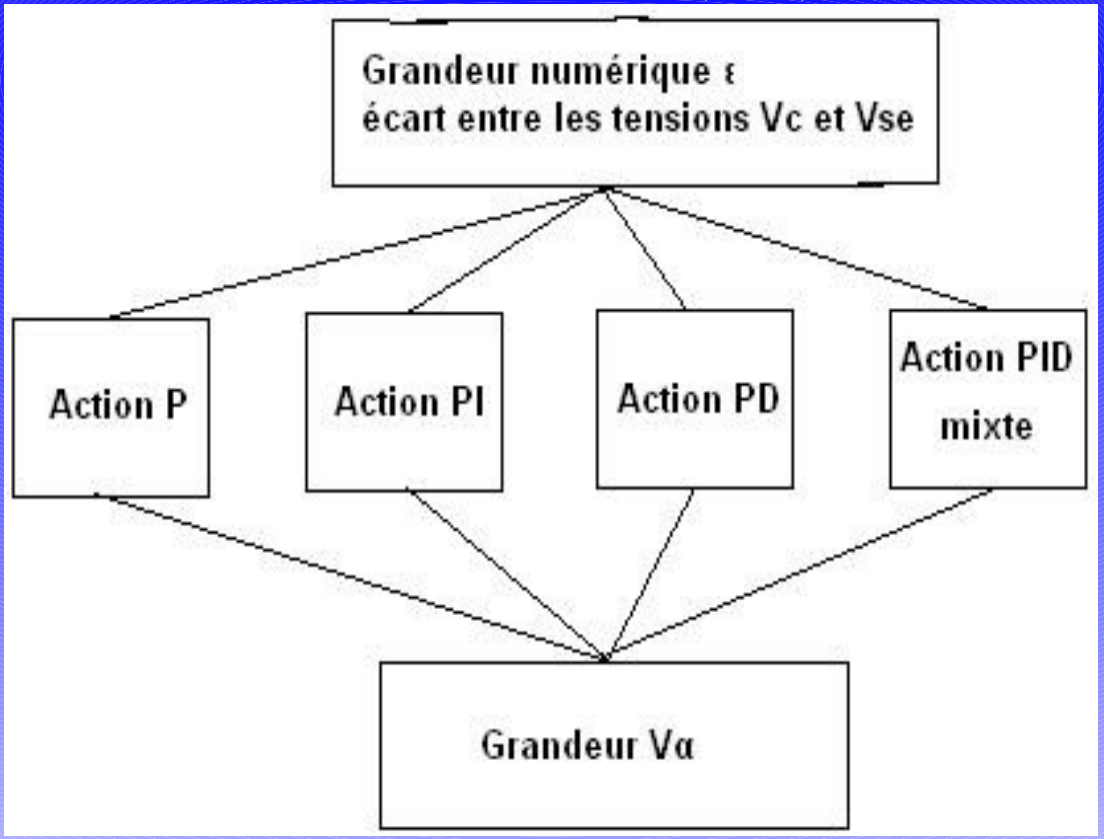
Objectif et mise en place la programmation

Programmation.

Objectif de la programmation

Commander le hacheur

- Envoyer un signal Vα en créneau positif avec une fréquence et un rapport cyclique notée α.
- Comparaison entre 2 tensions : Vc et Vse
- Traitement de l'écart ε : action P, PI, PD ou PID



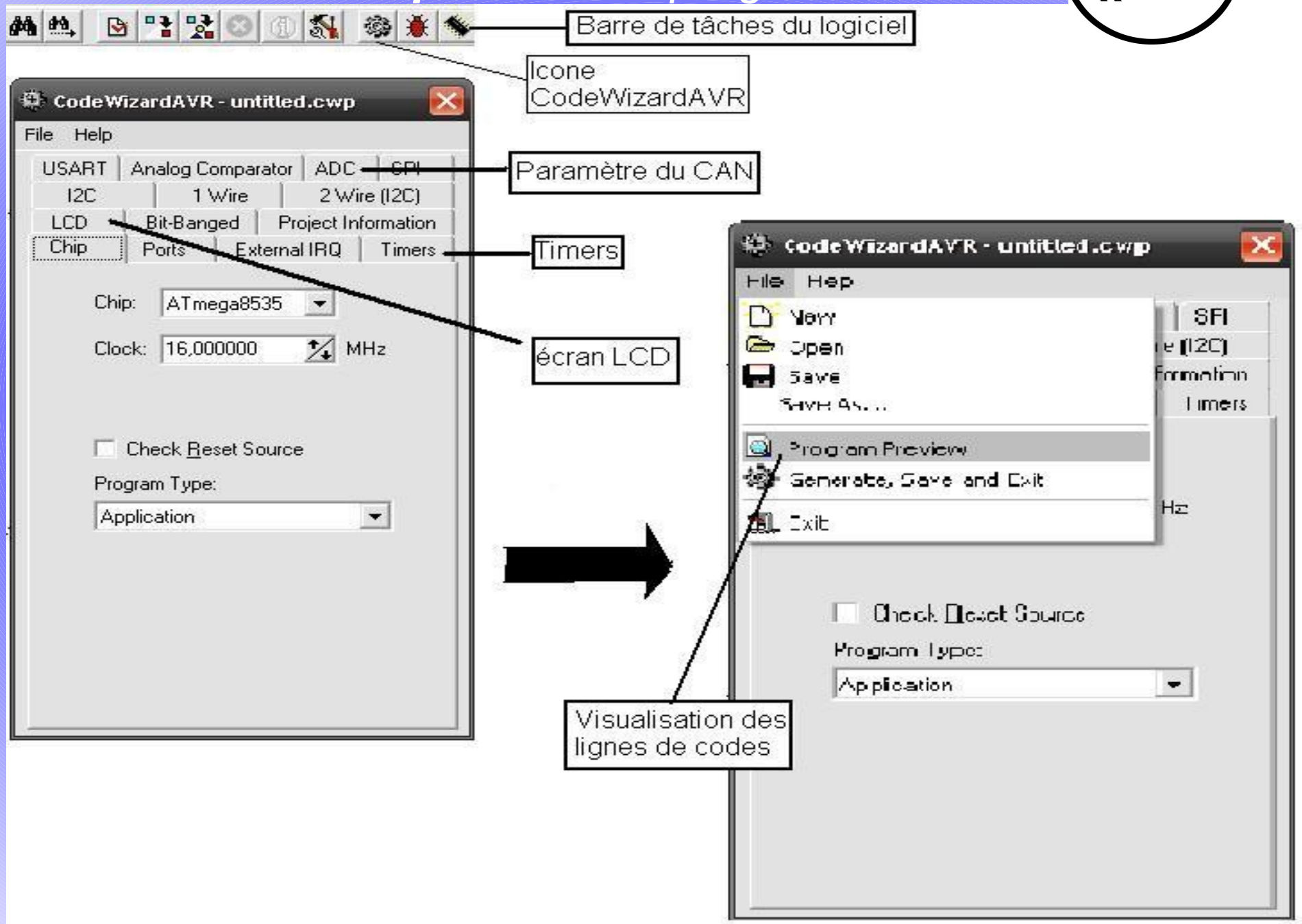
Programmation.

Mise en place de la programmation

- Prise en main du logiciel Code Vision AVR
- Nouvel environnement
- outil Code Wizard : Générateur de programme automatique

Programmation.

Mise en place de la programmation



Programmation.

Le Convertisseur Analogique Numérique (CAN)

Programmation.

Le CAN

➔2 tensions à convertir

- Tension de consige Vc
- Tension de mesure Vm

➔Une seule tension

- Stratégie n°1 : utiliser un ancien programme 'voltemètre.c
- Affichage sur écran LCD

Programmation.

Le CAN

```
/*Déclaration des bibliotèques*/
#include<mega8535.h>
#include<lcd.h>
#include<delay.h>
#include<stdio.h>
/*Prototypes de fonctions*/
interrupt[ADC_INT]void adc_isr(void);
void brdInit(void);
/*Déclaration des variables*/
int tension;
unsigned char tampon[20];

/*programme principal*/
void main (void)
{
        brdInit();
        #asm("sei")
        ADCSRA|=0x40;           //on lance une seule fois la conversion
        while(1)
        {
        }
}
```

Programmation.

Le CAN

➔Définition des fonctions

```
void brdInit(void)
{
    #asm
    .equ __lcd_port=0x15
    #endasm
    lcd_init(16);

    lcd_clear(); // cette fonction
permet d'effacer l'écran LCD.

    ADMUX=0b01000000;
    ADCSRA=0b10101110;
    SFIOR=0x00;
}
```

```
interrupt[ADC_INT]void
adc_isr(void)
{
        lcd_gotoxy(0,0);
        lcd_putsf("   Tension 1   ");


        tension=ADCW;


sprintf(tampon,"V=%5d",tension);
          lcd_gotoxy(0,1);
          lcd_puts(tampon);
         delay_ms(100);
  }
```

Programmation.

Le CAN

➔2 tensions

- Stratégie n°2 : utiliser Code Wizard.
  - Lignes de codes pour configurer le CAN

```
#define ADC_VREF_TYPE 0x00
[suite du prorammme ... ]
// Analog Comparator
initialization
// Analog Comparator: Off
// Analog Comparator Input
Capture by Timer/Counter 1:
Off
ACSR=0x80;
SFIOR=0x00;
// ADC initialization
// ADC Clock frequency:
1000,000 kHz
// ADC High Speed Mode:
Off
// ADC Auto Trigger Source:
None
ADMUX=ADC_VREF_TYPE;
ADCSRA=0x84;
SFIOR&=0xEF;
```

Programmation.

Le CAN

➔ Code Wizard : création d'une fonction

- prototype de la fonction

unsigned int read_adc(unsigned char adc_input);

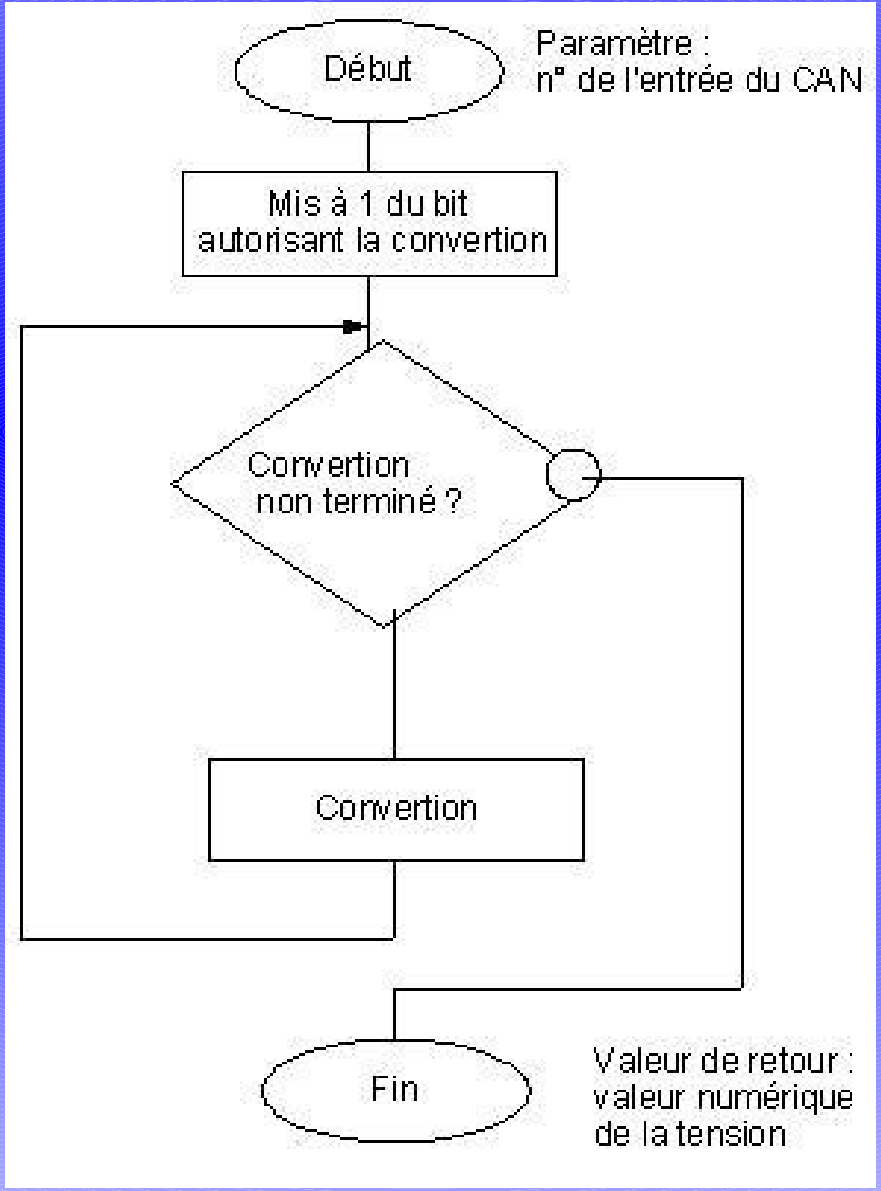
- Définition de la fonction

```
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input|ADC_VREF_TYPE;
// Start the AD conversion
ADCSRA|=0x40; //autorisation de la conversation
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCW;
}
```

Programmation.

Le CAN

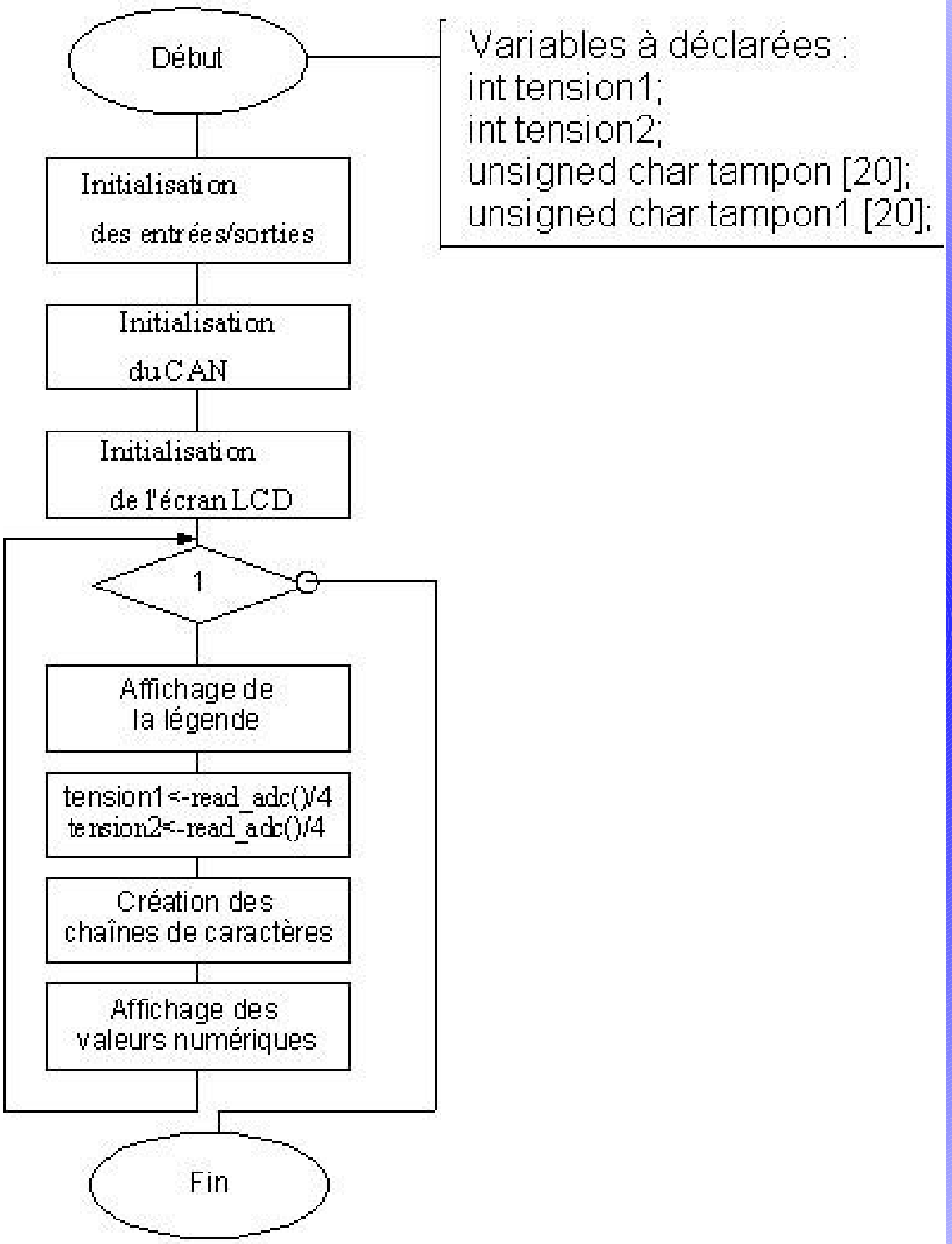
Ordinogramme de la fonction



Programmation.

Le CAN

Ordinogramme : Affichage des deux tensions sur LCD


Début
Variables à déclarées :
int tension1;
int tension2;
unsigned char tampon [20];
unsigned char tampon1 [20];
Initialisation des entrées/sorties
Initialisation du CAN
Initialisation de l'écran LCD
1
Affichage de la légende
tension1<-read_adc()/4
tension2<-read_adc()/4
Création des chaînes de caractères
Affichage des valeurs numériques
Fin

Programmation.

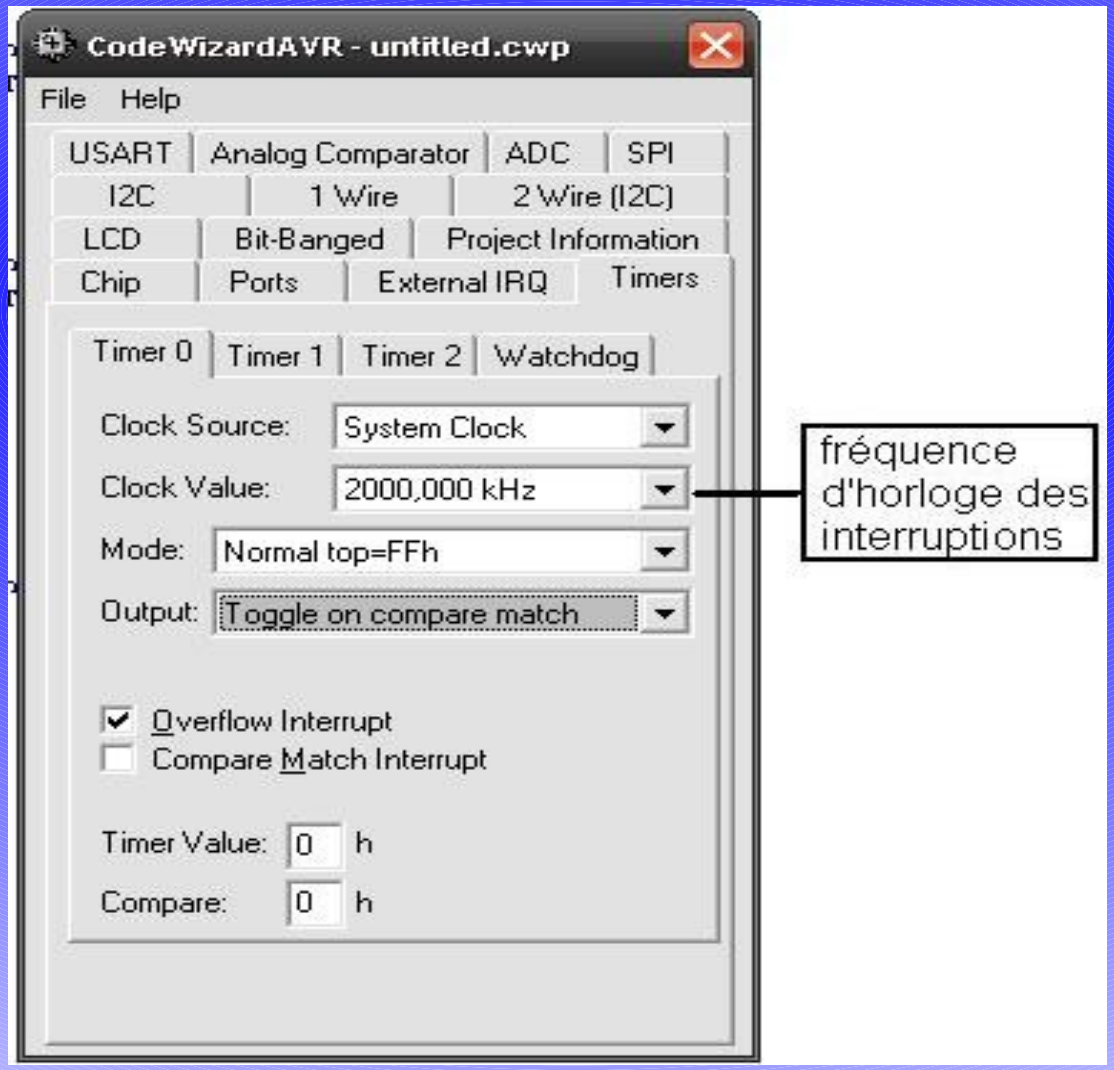
Mise en place d'une routine d'interruption

Programmation.

Mise en place d'une routine d'interruption

Objectif : effectuer des opérations à un interval régulier

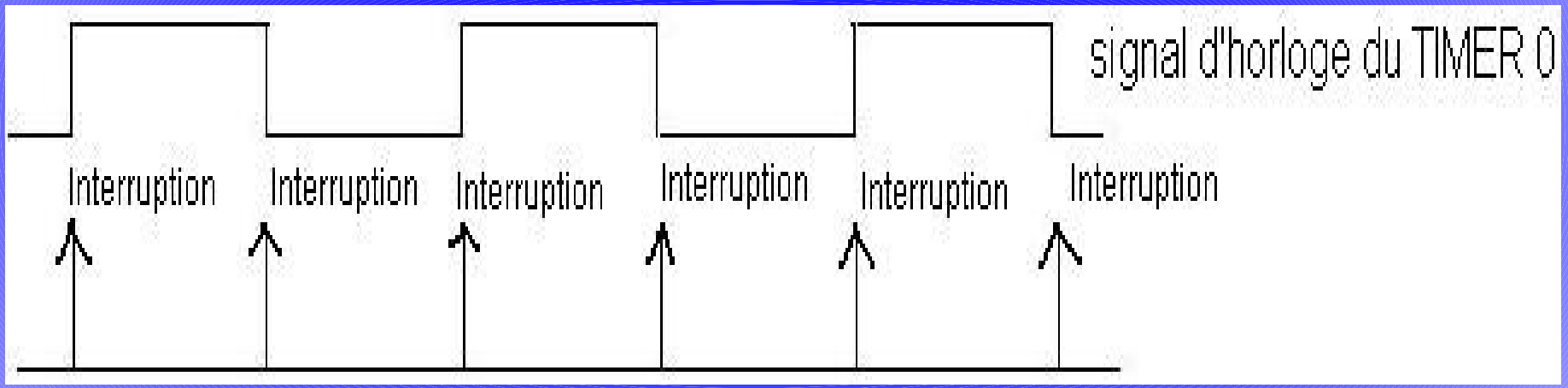
- Utilisation du TIMER 0
- Configuration du TIMER 0



Programmation.

Mise en place d'une routine d'interruption

Principe des interruptions = principe d'échantillonnage



Programmation.

Mise en place d'une routine d'interruption

➔Lignes de code relatives à la configuration non abordée dans notre soutenance.

➔Fonction fournie par Code Wizard :

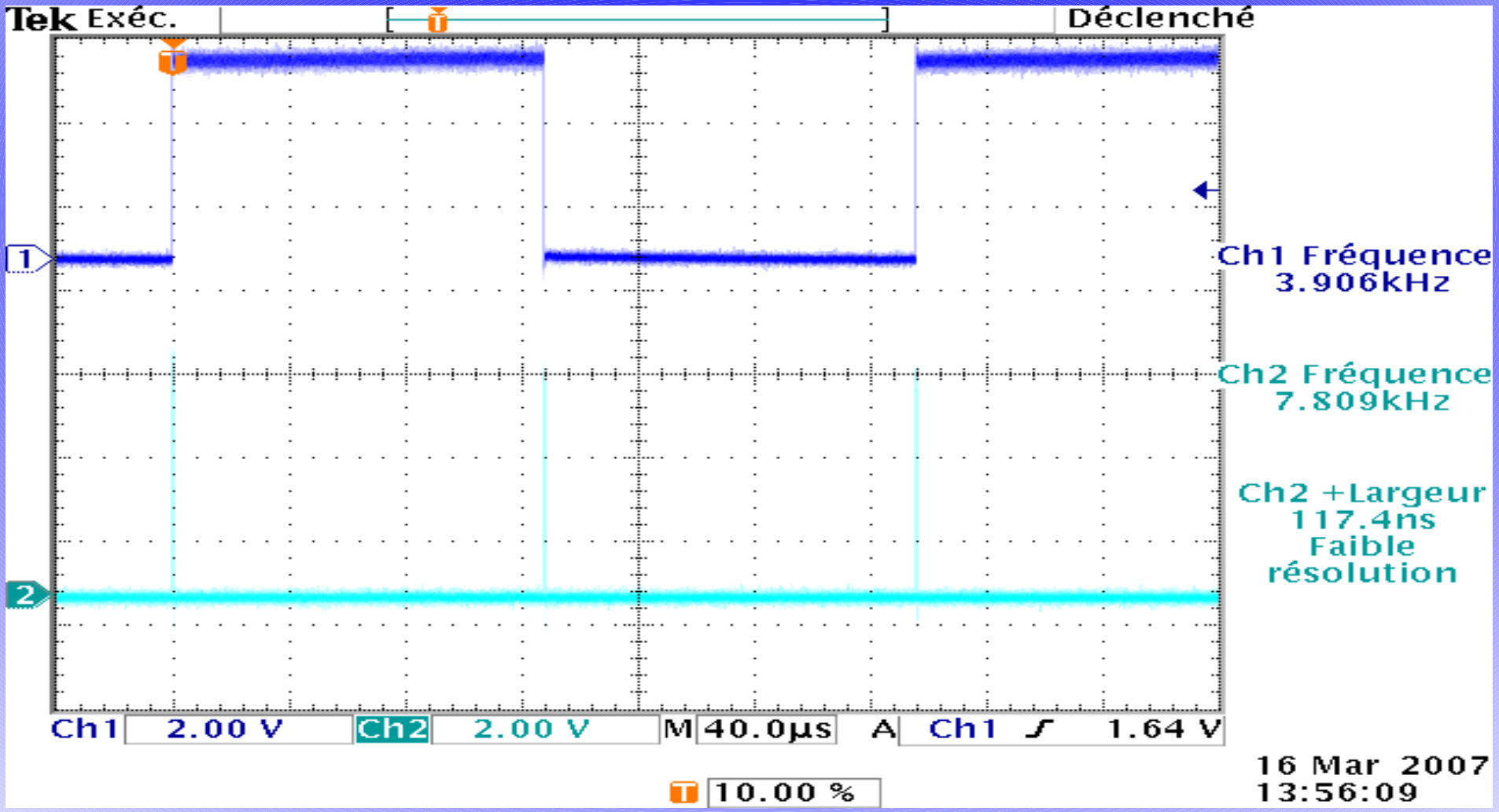
```
interrupt [TIM0_OVF] void timer0_ovf_isr(void);
```

➔Ensemble des opérations placées dans la routine

Programmation.

Mise en place d'une routine d'interruption

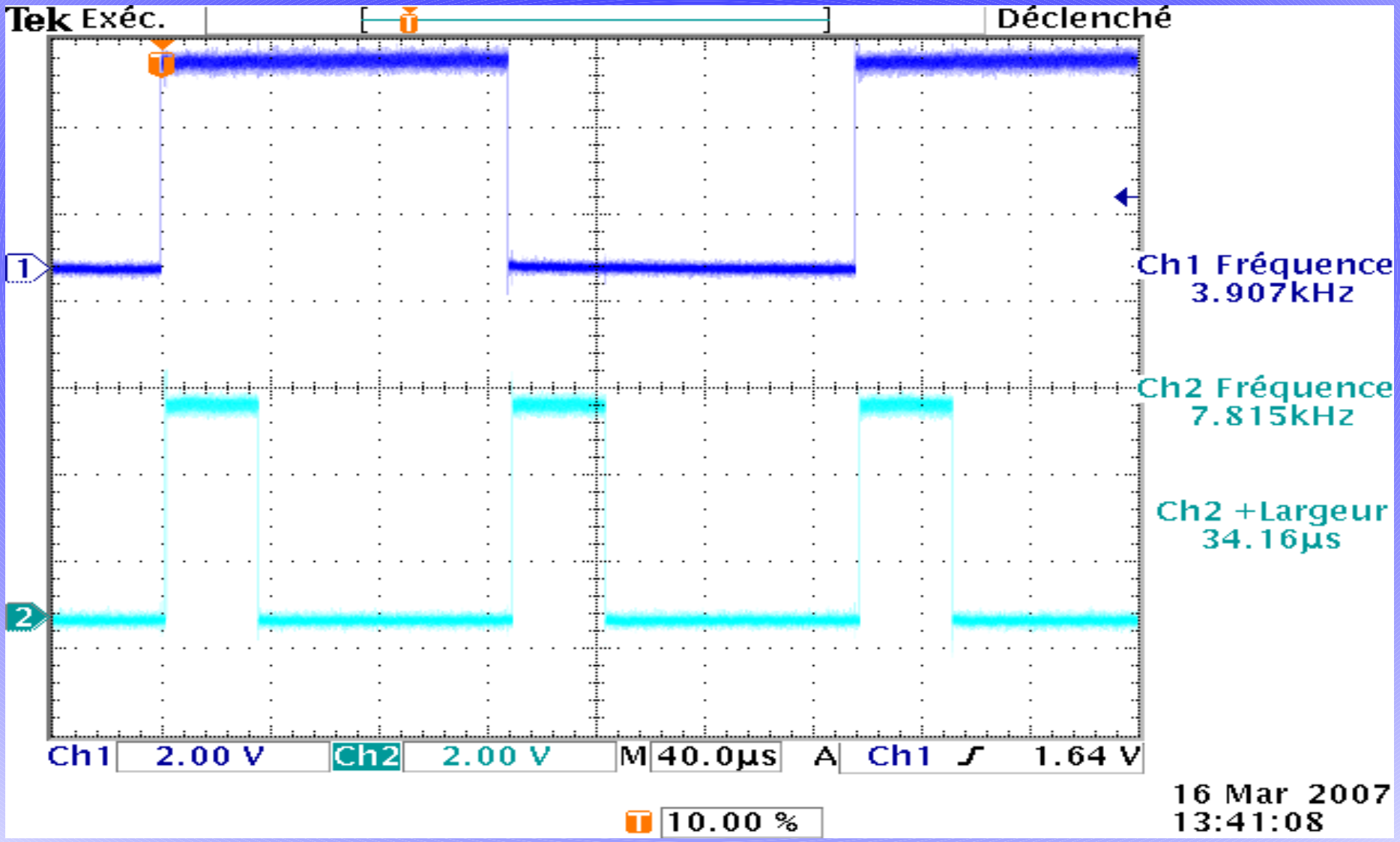
➔Test n°1 : mis à 1 et 0 alternativement du PIN 0 du PORTD



Programmation.

Mise en place d'une routine d'interruption

➔Test n°2 : Convertion insérée dans la routine



Mise en place de la fonction PWM

Programmation.

Mise en place de la fonction PWM

- Objectif : Création d'un signal en créneau
  - signal de commande Va
- Utilisation d'un nouveau TIMER : TIMER 1
  - Configuration : Code Wizard

```
TCCR1A=0x81;
TCCR1B=0x81;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x40; // registre important. Permet de changer α
OCR1BH=0x00;
OCR1BL=0x00;
```

Programmation.

Mise en place de la fonction PWM

Traitement des tensions converties:

- on note εc l'écart corrigé entre Vc et Vm
  - εc= Va

Test n°1 : correction à action P : Kr=2

La grandeur numérique Va sera rentrée dans le registre OCR1AL

Programmation.

Mise en place de la fonction PWM

Nouvelle définition de la routine d'interruption

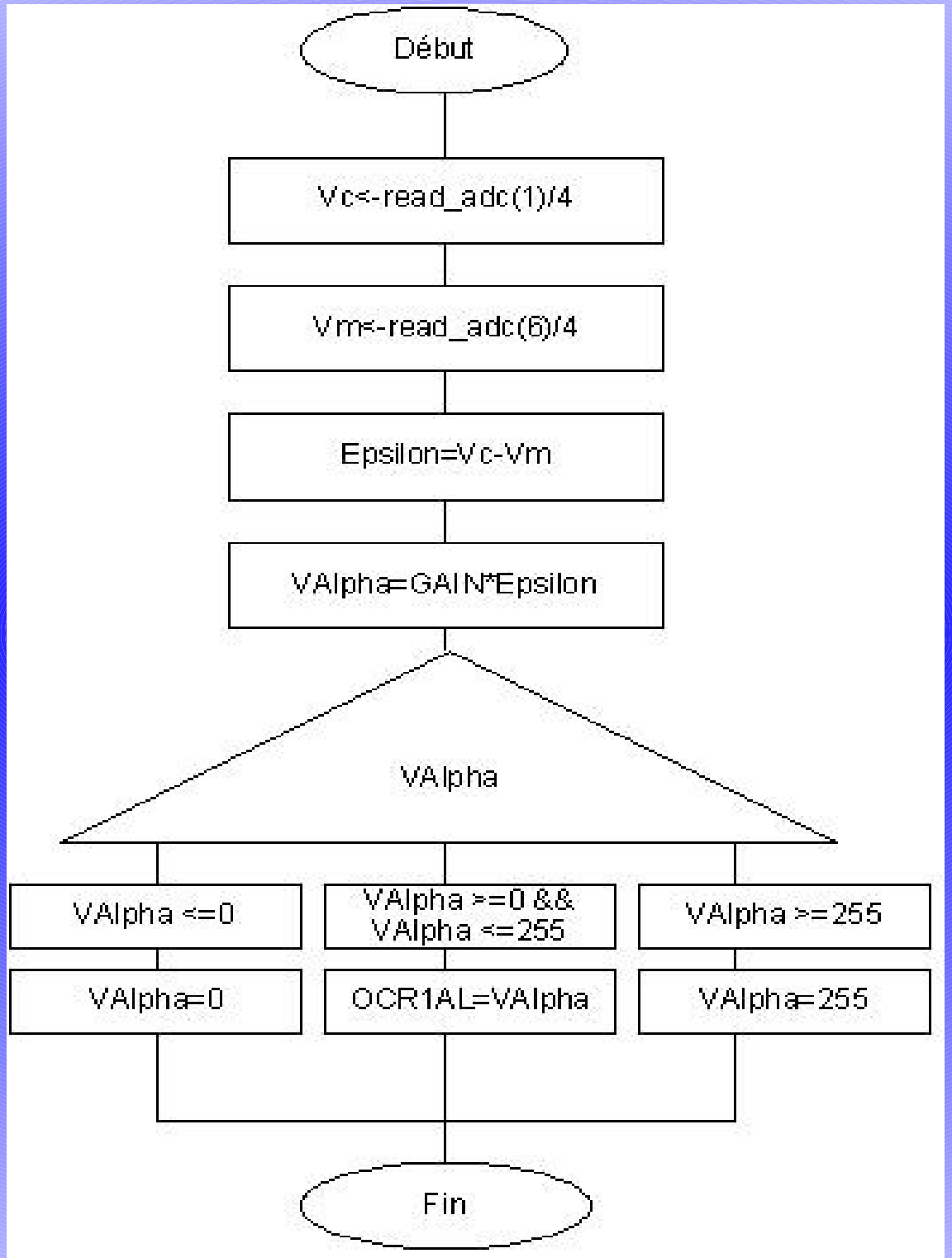
```
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
    // Place your code here
    Vc=read_adc(1)/4;
    Vm=read_adc (6)/4;
    Epsilon=Vc-Vm;
    VAlpha=GAIN*Epsilon;
    if (VAlpha<=0)
    VAlpha=0;
    if(VAlpha>=255)
    VAlpha=255;
    if (VAlpha>=0 && VAlpha<=255 )
    {OCR1AL=VAlpha;}
}
```

Routine d'affichage sur LCD effectuée dans la boucle infinie

Programmation.

Mise en place de la fonction PWM

Ordinnogramme de la fonction routine d'interruption : Traitement des signaux numériques


Début
Vc<-read_adc(1)/4
Vm<-read_adc(6)/4
Epsilon=Vc-Vm
VAlpha=GAIN*Epsilon
VAlpha
VAlpha <=0
VAlpha >=0 &&
VAlpha <=255
VAlpha >=255
VAlpha=0
OCR1AL=VAlpha
VAlpha=255
Fin

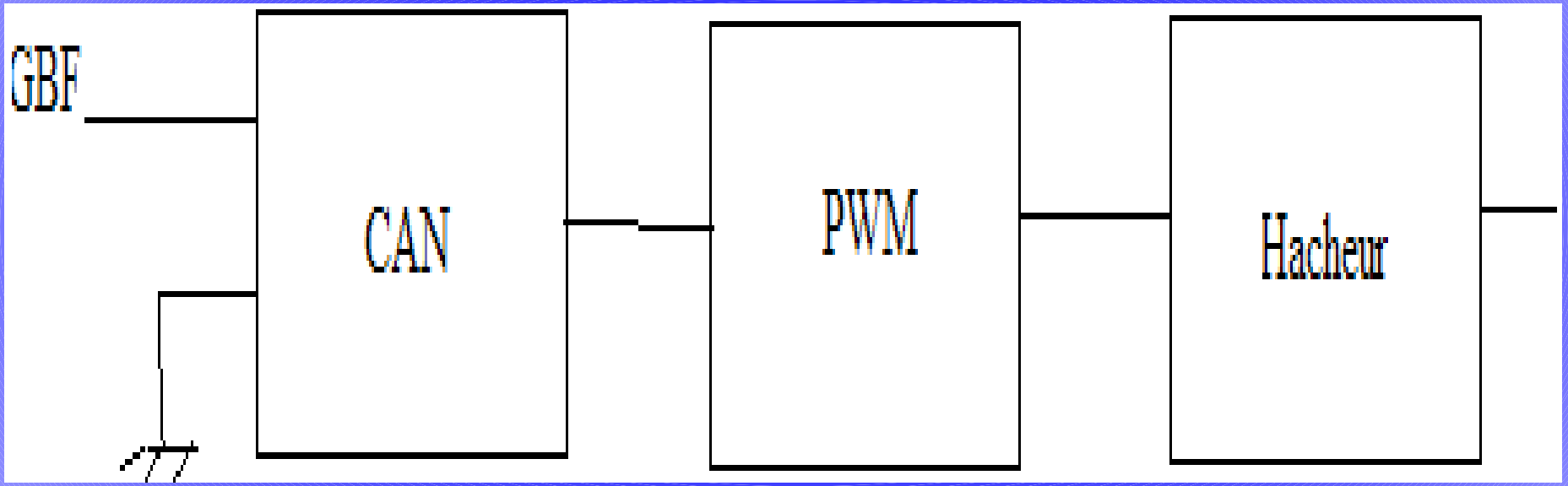
Programmation.

Relation PC Moteur

Programmation.

Relation PC-Moteur

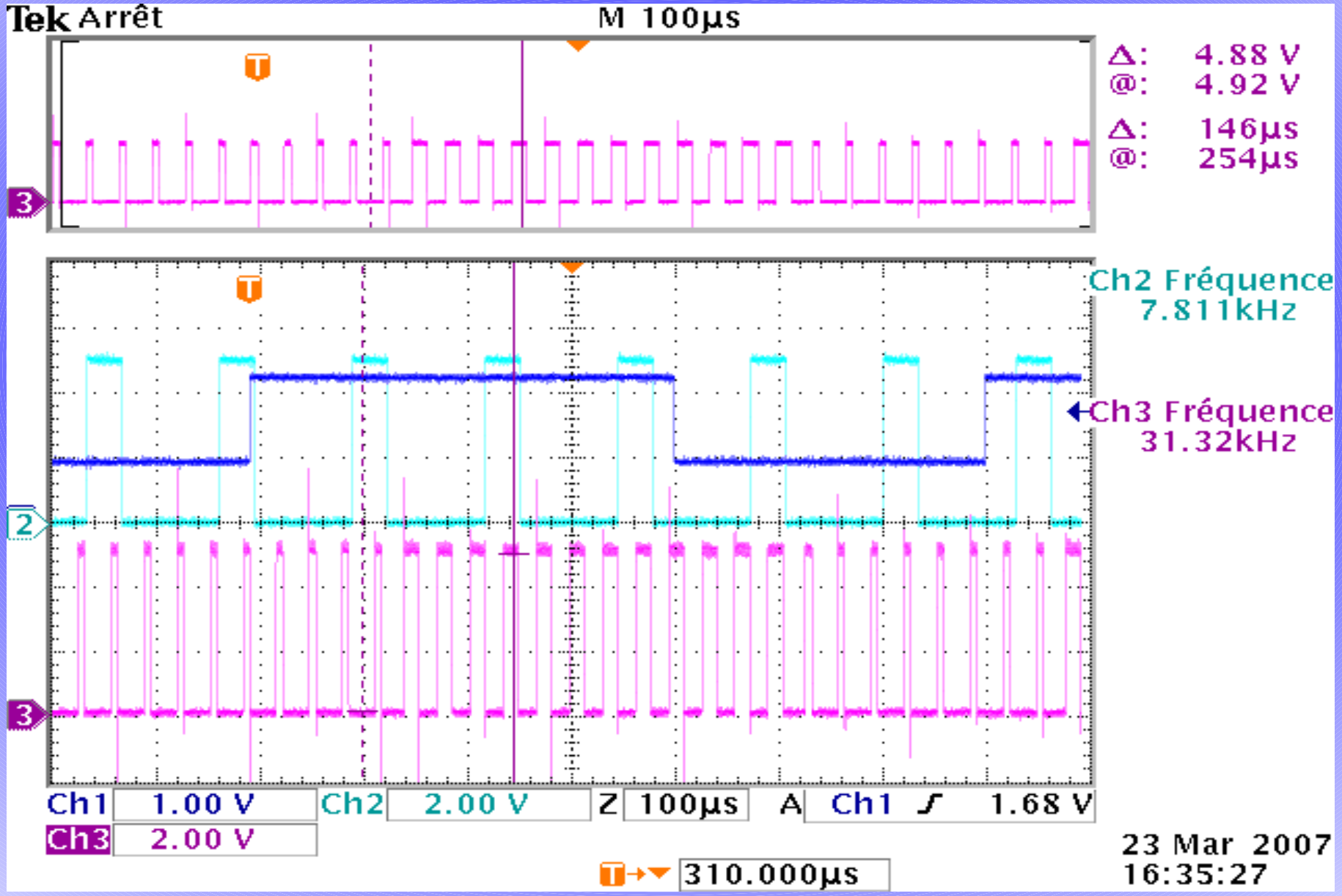
Test n°1 : Synoptique du test



Prochain test : séance suivante.

Programmation.

Relation PC-Moteur



Conclusion