Store automatique Somfy

THEME 2006

Note : Nom et Prénom supprimés à la diffusion de ce document sur le net.
Avec les remerciements à l'auteur.

SOMMAIRE

PARTIE ELECTRONIQUE

PRESENTATION DU SYSTEME ETUDIE
(séquence 1)

Lors de cette séquence nous avons pu prendre connaissance du sujet de l'étude, identifier les différentes structures. Afin de pouvoir expliquer le fonctionnement du store automatique.

1.1 Présentation de la société SOMFY :

Installé depuis son origine, en 1969, à Cluses en Haute-Savoie, Somfy s'appuie sur le rayonnement international de Somfy pour apporter à ses clients une prestation de proximité et de qualité.

Somfy conçoit des solutions de motorisation et d'automatisation aussi faciles d'installation que d'utilisation. Elles se déclinent aujourd'hui sur les volets roulants, les stores extérieurs et intérieurs, les portails, les portes de garage et les alarmes.

Somfy permet d'apporter encore plus de confort et de sécurité dans l'habitat. Ce sont au total 140 personnes à Cluses qui sont à l'écoute de leurs clients.

Site Internet SOMFY : http://www.somfy.com

1.2 Eléments composants le système :

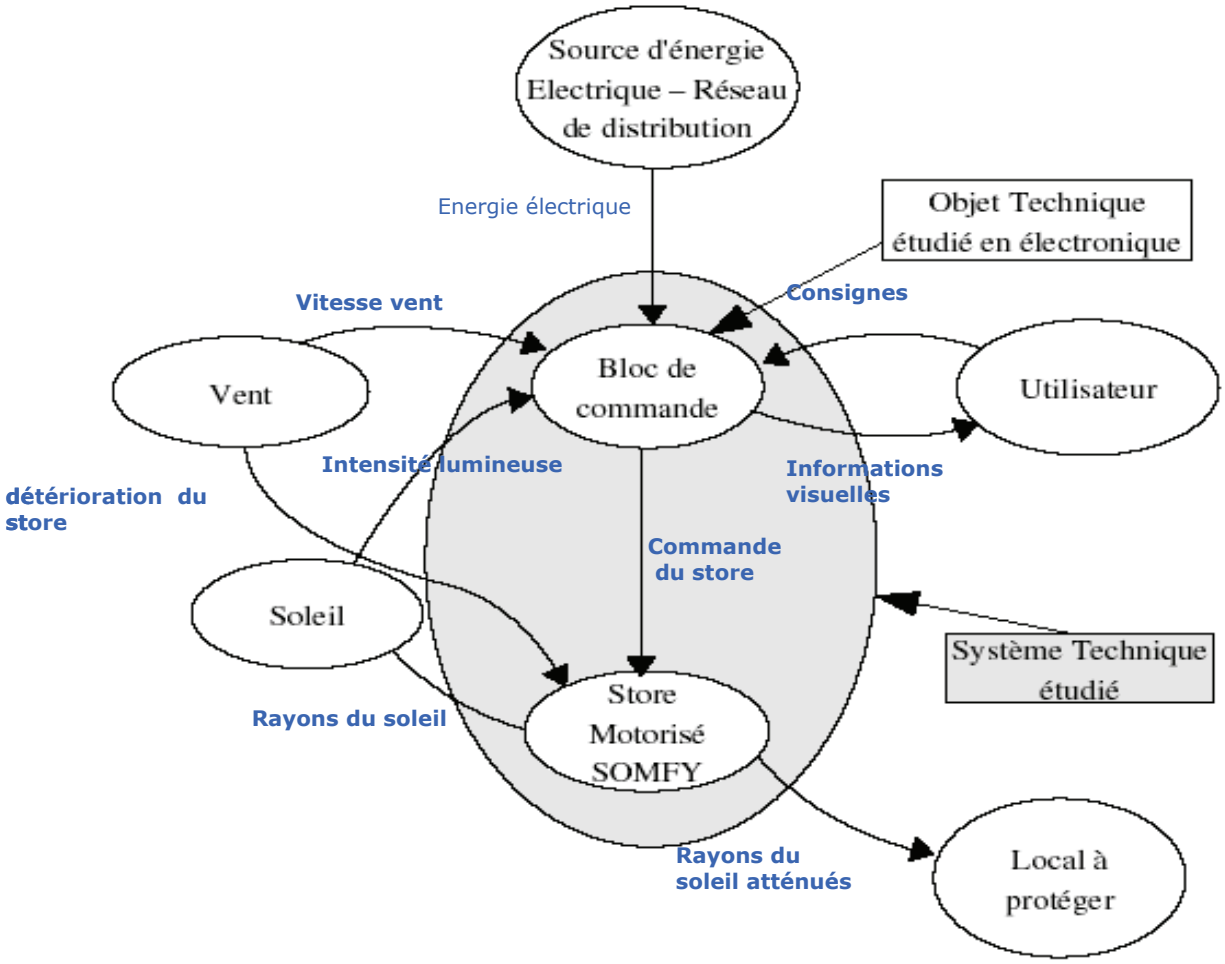
Le système étudié est composé d'un bloc de commande et d'un store motorisé.

→ Le bloc de commande permet d'actionner automatiquement (ou manuellement) le moteur du store. Ce bloc de commande constitue le cœur d'un automatisme qui permet d'agir sur la position du store et donc de le faire sortir ou rentrer par rapport aux conditions climatiques (vent et soleil) et par rapports aux souhaits de l'utilisateur.

→ Le store motorisé permet de protéger un local, une terrasse… contre l'excès d'ensoleillement.

Remarque : Cet automatisme a besoin aussi d'une source d'énergie pour fonctionner.

1.3 Diagramme sagittal :



1.4 Identification des fonctions principales

Une lecture de la description fonctionnelle permet d'identifier sur le schéma structurel les fonctions principales.

☞ Se reporter au découpage fonctionnel sur le schéma de la page suivante.

Description des fonctions :

FP1 : Cette fonction permet de mesurer l'ensoleillement extérieur reçu par le capteur. Celui-ci délivre une tension qui est proportionnelle à l'éclairement reçu à la fonction FP2.

FP2 : Cette fonction détecte le seuil où il faut faire descendre ou monter le store. Ce seuil de déclenchement est réglable.

FP3 : Cette fonction mesure la vitesse du vent et transmet à la fonction FP4, une tension proportionnelle à la vitesse du vent.

FP4 : Cette fonction délivre un signal logique indiquant la présence du vent.

FP5 : Cette fonction traite les informations en provenance des fonctions principales situées sur sa périphérie.
Elle contrôle directement le store à l'aide de la fonction FP7.
Elle informe aussi l'utilisateur à l'aide de la fonction FP8.

FP6 : Cette fonction met en forme des signaux de commandes pour que FP5 ait des signaux exploitables.

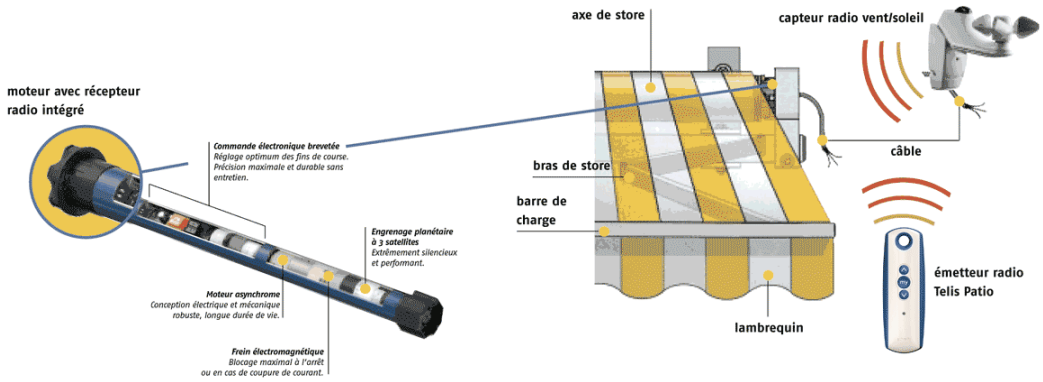
FP7 : La fonction adapte la puissance afin que la fonction FP5 puisse contrôler le store.

FP8 : Cette fonction transmet des informations à l'utilisateur sur le fonctionnement du store sous forme lumineuse.

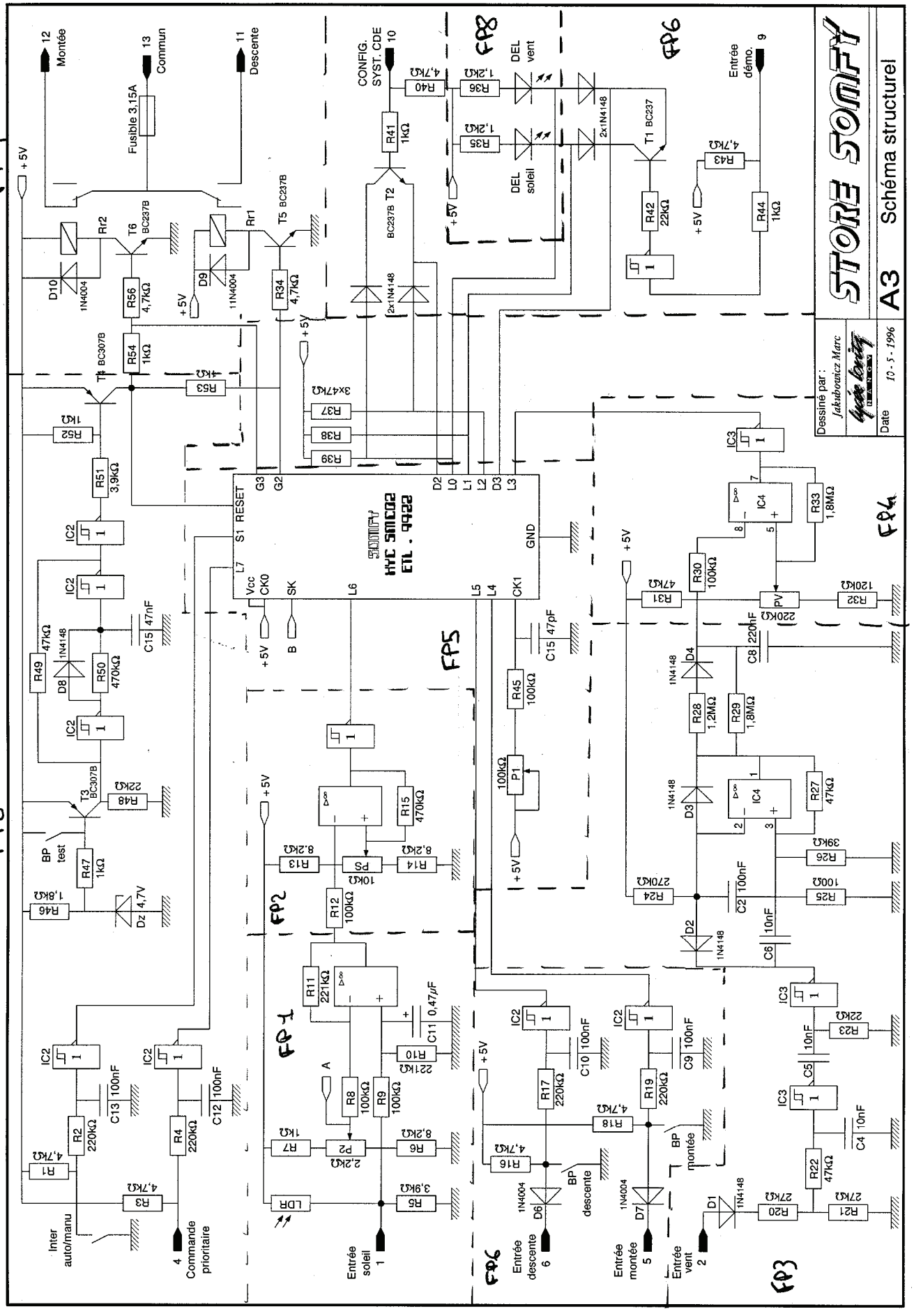
1.5 Les solutions techniques

En parcourant les informations constructeurs à notre disposition sur le site Internet du constructeur SOMFY, nous pouvons constater que le constructeur a choisi d'utiliser plusieurs techniques de transmission, pour communiquer entre le boîtier de commande et le capteur.

Par exemple nous pouvons constater que Somfy utilise plusieurs technologies pour transmettre des informations : la transmission radio, la transmission par fils électriques , la transmission infra rouge,…

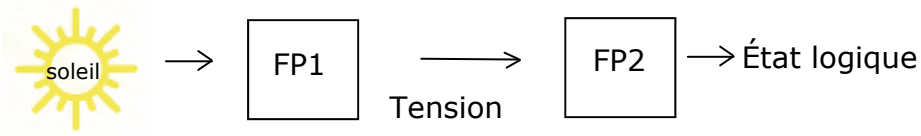


Identification des fonctions principales sur le schéma structurel



ETUDE DE CHAINE CAPTEUR SOLEIL
(séquence 2)

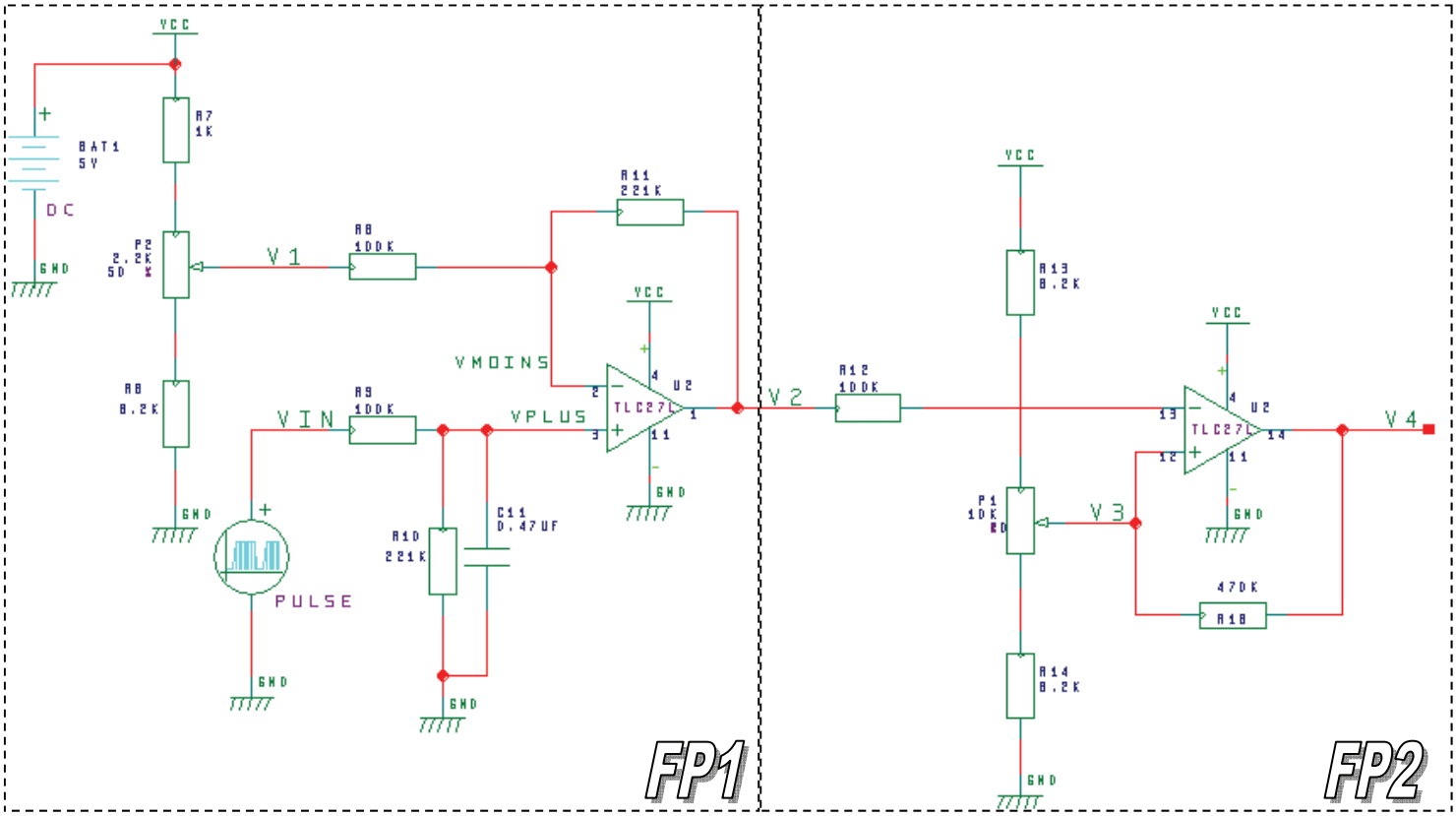
Lors de cette séquence nous avons pu étudier le fonctionnement de la chaîne capteur soleil. Nous avons vu que l'intensité lumineuse entrait dans FP1.
Il en ressort une tension image de l'intensité lumineuse, qui rentre dans FP2.
Cette tension est comparée à une consigne. Pour en ressortir un état logique. L'état 1 correspond à l'information « trop de soleil » et l'état 0 correspond à peu de soleil.



2.1 Identification des différentes structures

Nous pouvons voir sur le schéma ci-dessous que la fonction FP1 est composée d'un ALI branché en mode soustracteur.

La fonction FP2 est composée d'un ALI branché en mode comparateur.

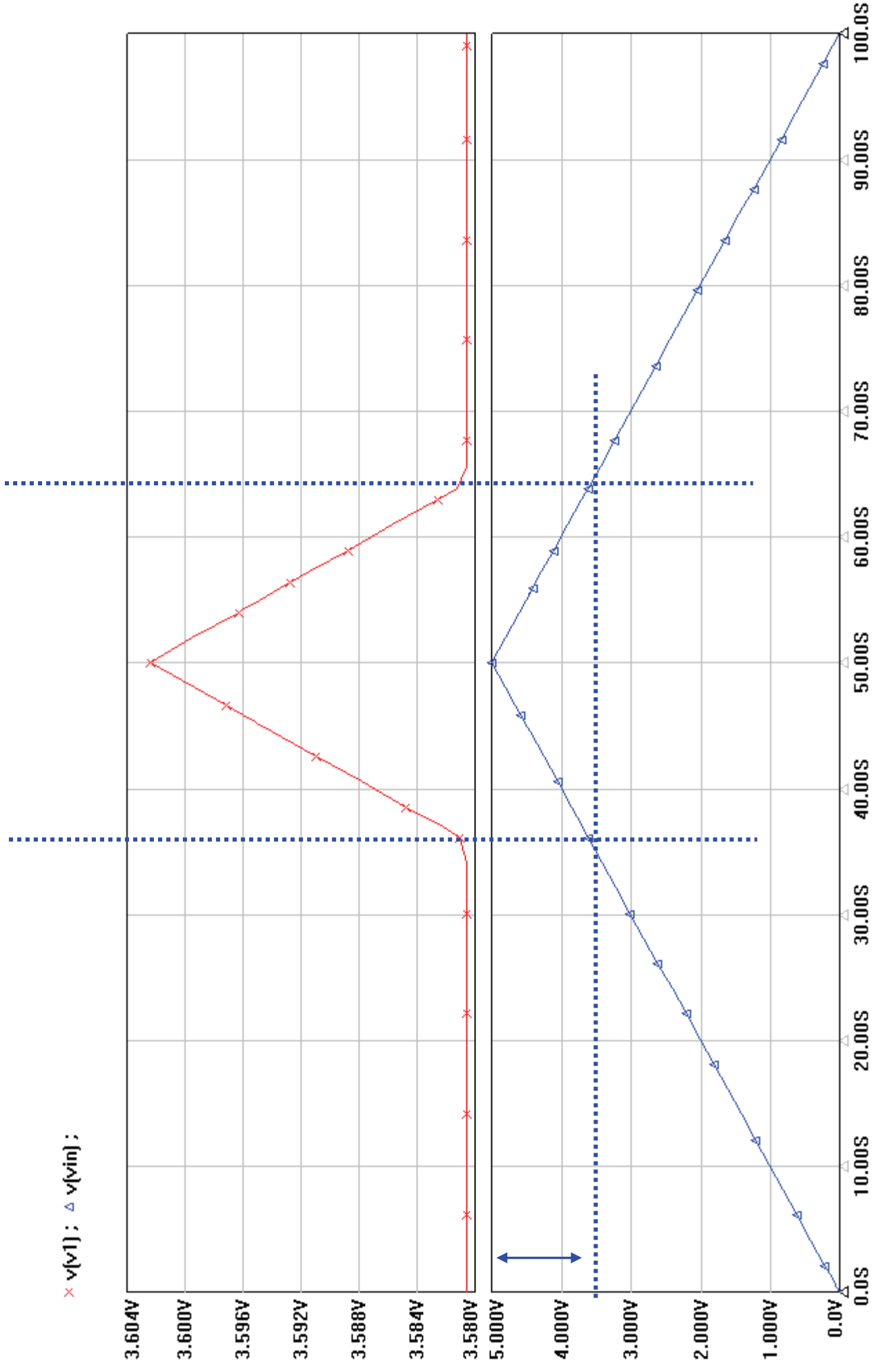


2.2 Choix du signal VIN.

Nous voulons que V2 soit représentatif de l'éclairement reçu. On observe que V2 est l'image de l'intensité lumineuse lorsque VIN est compris entre 3,5 et 5V.

Voir graphique page suivante.

Choix du signal VIN

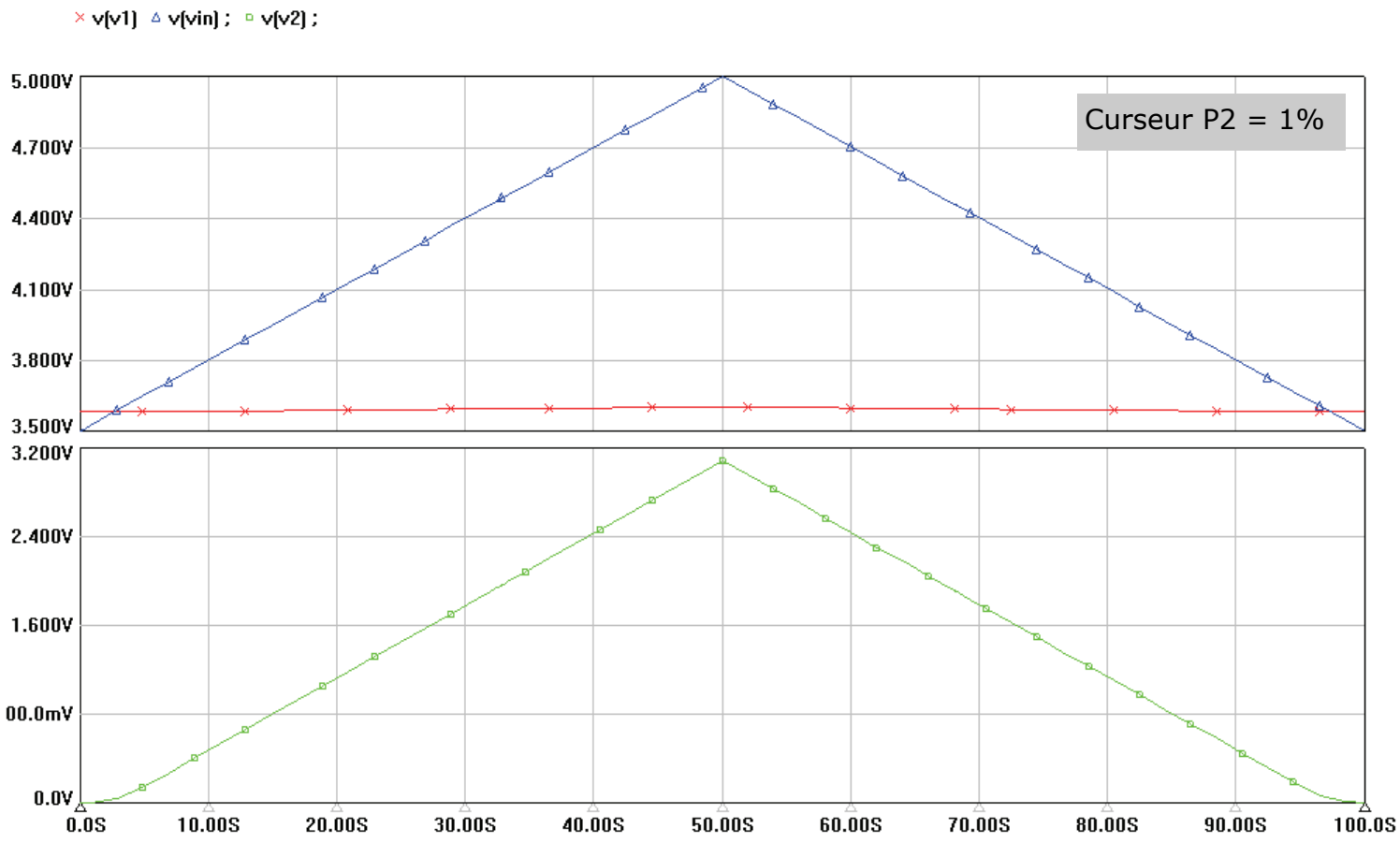


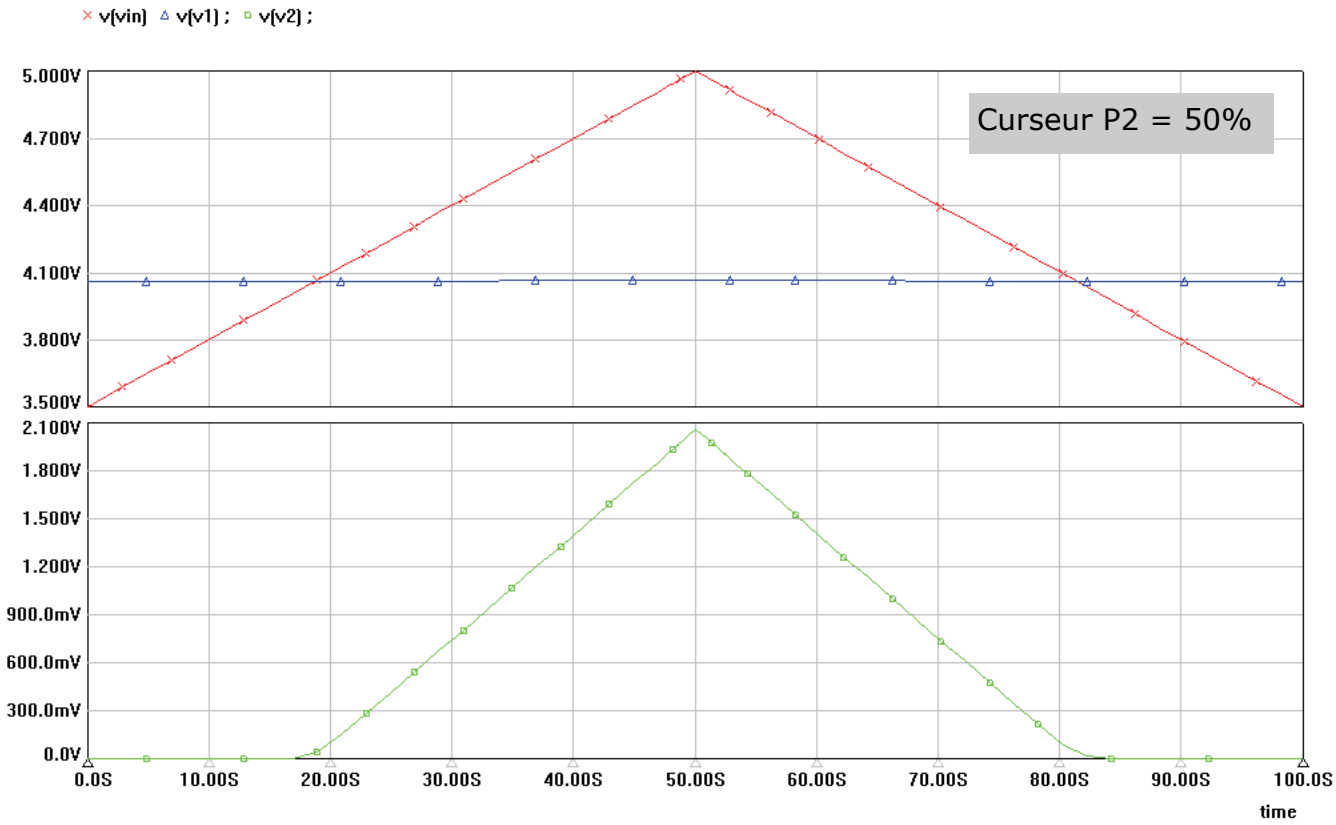
2.3 Relevé de la tension de la sortie U2 en fonction de la position du curseur P2.

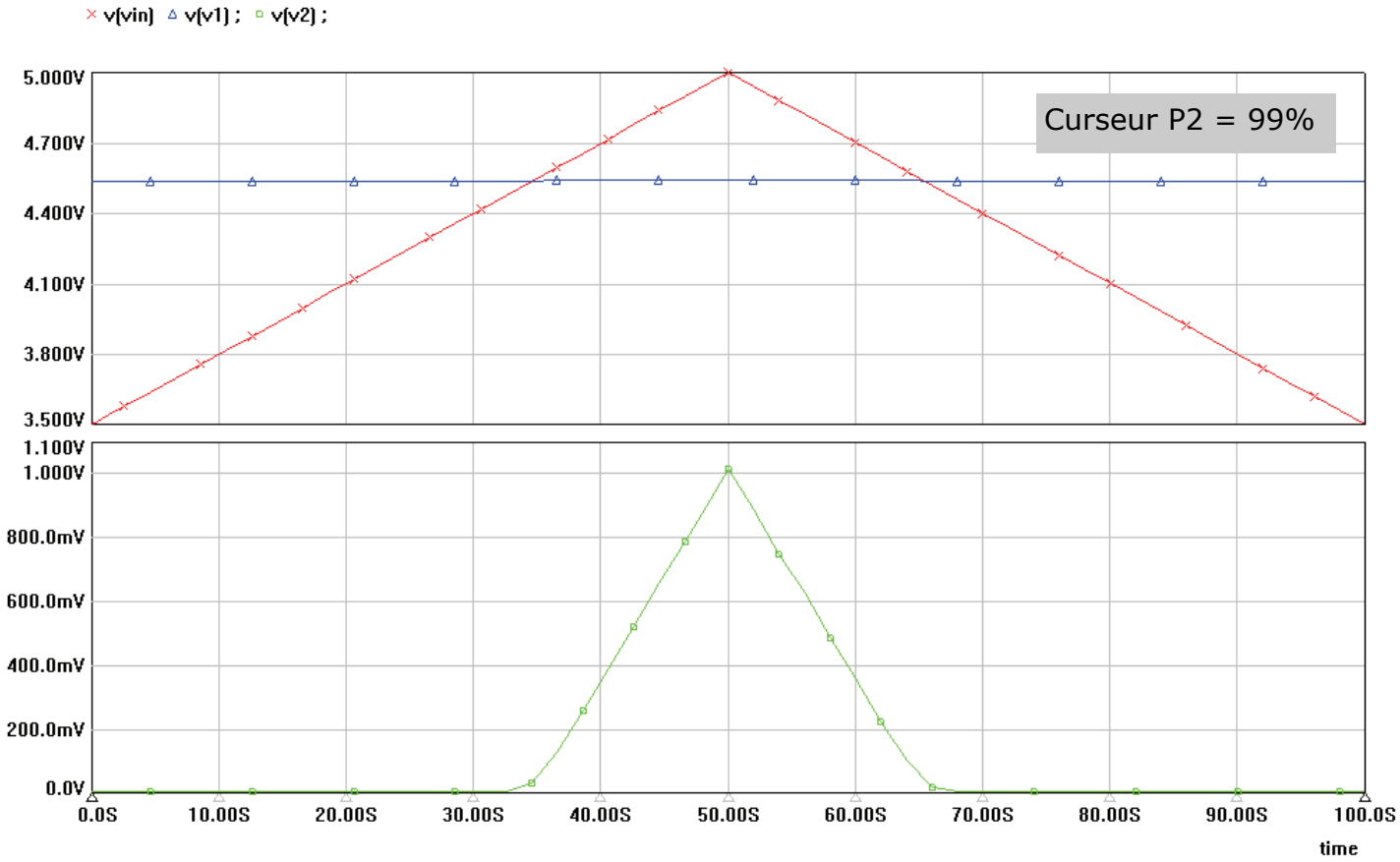
La position du curseur P2 influe sur la valeur de la tension V1.
V1 a une influence sur V2, quand V1 augmente V2 diminue.

En fait, P1 permet de changer la valeur moyenne de la tension présente en sortie de U2. Afin que cette tension reste dans le domaine linéaire (tension comprise entre 0V et 5V).

Exemple : on observe sur les chronogrammes suivants que pour P2 = 99%, la plage de non linéarité est plus importante que sur le chronogramme pour P2 = 1%.



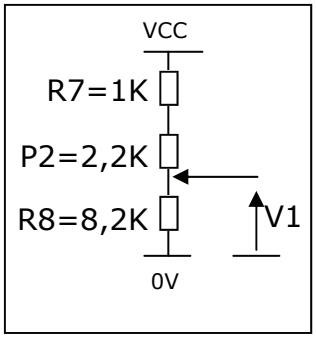

v(vin) v(v1) ; v(v2) ;
Curseur P2 = 50%
5.000V
4.700V
4.400V
4.100V
3.800V
3.500V
2.100V
1.800V
1.500V
1.200V
900.0mV
600.0mV
300.0mV
0.0V
0.0S
10.00S
20.00S
30.00S
40.00S
50.00S
60.00S
70.00S
80.00S
90.00S
100.0S
time


v(vin) v(v1) ; v(v2) ;
Curseur P2 = 99%
5.000V
4.700V
4.400V
4.100V
3.800V
3.500V
1.100V
1.000V
800.0mV
600.0mV
400.0mV
200.0mV
0.0V
0.0S
10.00S
20.00S
30.00S
40.00S
50.00S
60.00S
70.00S
80.00S
90.00S
100.0S
time

2.4 Expression théorique des différentes valeurs de V1 selon la position du curseur P2.

Premier cas : Curseur P2 au minimum (0%)

Schéma équivalent :



A l'aide du pont diviseur de tension on trouve :

V1min = VCC x R8 / (R7+R8+P2)

V1min = 3.596V

Deuxième cas : Curseur P2 au milieu (50%)

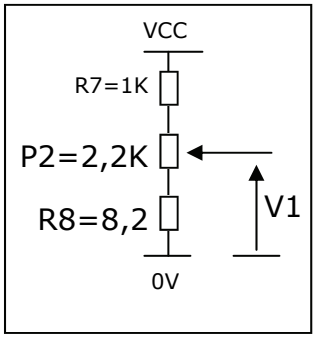


Schéma équivalent :

A l'aide du pont diviseur de tension on trouve :

V1 = VCC((R8+ 1/2P2)/(R8+R7+P2))

V1 = 4.08V

Troisième cas : Curseur P2 au maximum (100%)

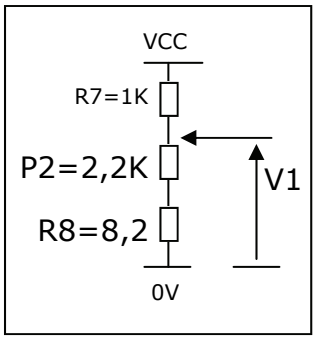


Schéma équivalent:

A l'aide du pont diviseur de tension on trouve :

V1 = VCC((R8+ P2)/(R8+R7+P2))

V1 = 4.56V

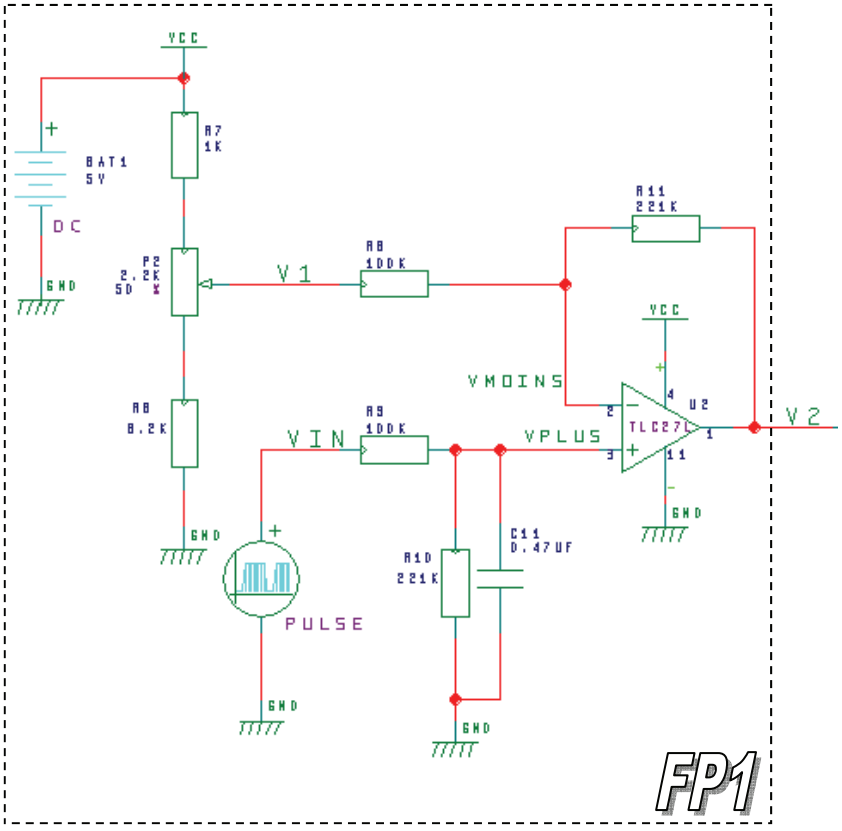
Tableau résumé de la plage de variation de la tension V1 selon la position du curseur de P2 :

Position de P2	Valeur de V1 (calcul théorique)	Valeur de V1 (Résultats de simulation)
100%	4.56 V	4.539 V
50%	4.08 V	4.057 V
0%	3.596 V	3.581 V

En simulation nous pouvons observer que les valeurs correspondent au calcul. Il y a une légère différence car l'intensité passant dans R8 et R11 n'est pas tout a fait nulle.

Mais aussi en simulation nous sommes obligés de mettre une valeur minimum de 1% au lieu de 0% pour faire fonctionner la simulation.

2.5 Calcul théorique de l'expression de la fonction de transfert de FP1

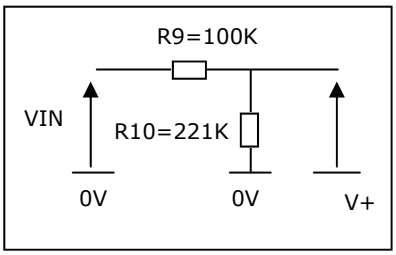


On va montrer que la fonction de transfert de FP1 peut s'écrire :
V2 = K (VIN-V1)

L'ALI fonctionne en régime linéaire car l'entrée moins est reliée à la sortie.
Donc on peut dire Vplus = Vmoins (ε = 0)

2.51 Expression de la tension V plus:

Schéma équivalent:

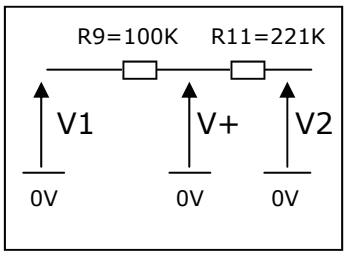


A l'aide du pont diviseur de tension on trouve :

Vplus = VIN x R10 / (R10+R9)

2.52 Expression de la tension V moins:

Schéma équivalent:



Pour trouver l'expression de Vmoins, on utilise le théorème de superposition :

Tout d'abord on inhibe V2 :
Vmoins = V1x R11 / (R8 + R11)

Puis on inhibe V1 :
Vmoins = V2 x R8 / (R8 + R11)

Et enfin on les additionnant, on obtient :

Vmoins = V1xR11 / (R8 + R11) + V2xR8 / (R8 + R11)

2.53 Expression de la tension V2 :

En partant de l'expression Vplus = Vmoins, on peut dire que :

VIN × R10 / (R10+R9) = V1×R11 / (R8 + R11) + V2×R8 / (R8 + R11)

On isole la tension V2 :

V2×R8 /(R8+R11) = (VIN×R10 / (R10+R9)) - (V1×R11/(R8 + R11))

V2=[(R8 + R11)/ R8]× [(VIN × R10 /(R10+R9)) - (V1×R11/(R8 + R11))]

Et en posant :
R = R11 = R10 = 221Kohms
R' = R8 = R9 = 100Kohms

V2 = ((VIN× R10 / (R10+R9)) - (R11×V1 / (R8+R11))) x ((R8+ R11)/R8)

V2 = ((VIN × R /(R+R')) - (R×V1/(R'+R))) x ((R' + R)/R')

En simplifiant on obtient :

V2 = (R/R') x (VIN-V1)

V2 = 2.21 (VIN-V1)

2.6 Seuils limites du comparateur:

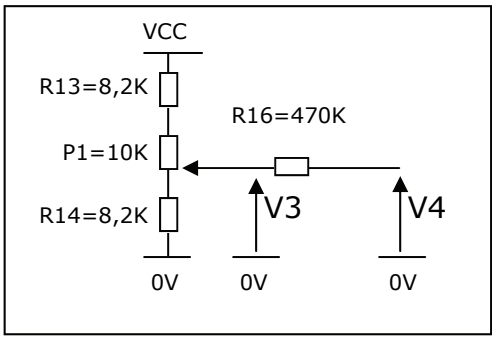
L'ALI fonctionne en régime non linéaire car l'entrée moins n'est pas reliée à la sortie.
Donc on peut dire que l'AIL fonctionne en mode comparateur :

Si Vplus > Vmoins alors V4 = VCC

Si Vplus < Vmoins alors V4 = GND

A. Détermination des seuils - Cas où le potentiomètre P1 est au minimum :

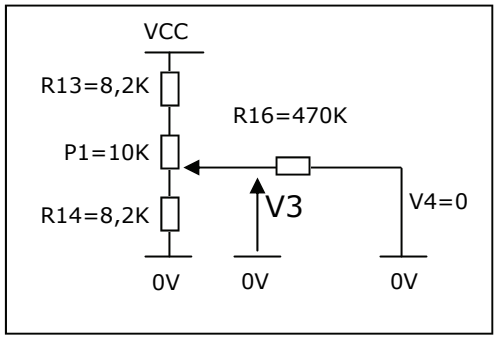
Schéma équivalent :



On utilise le théorème de superposition

Lorsque V4 = 0V :

Schéma équivalent :

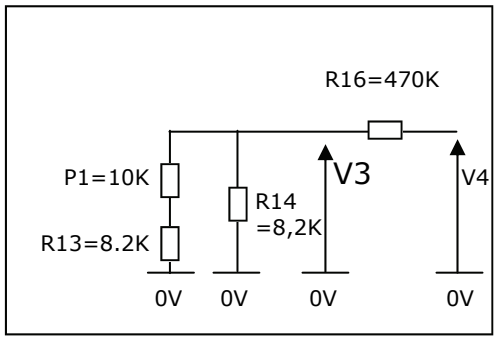


$$V3_1 = \frac{((R16\times R14)/(R14+R16))}{((R16\times R14)/(R14+R16))+R13+P1} \times VCC$$

$V3_1$ =1.53V (pour V4=OV)

Lorsque VCC = 0V :

Schéma équivalent:



$$V3_2 = \frac{(P1+R13)\times R14/(P1+R13+R14)}{(P1+R13)\times R14/(P1+R13+R14)+R16} X\ V4$$

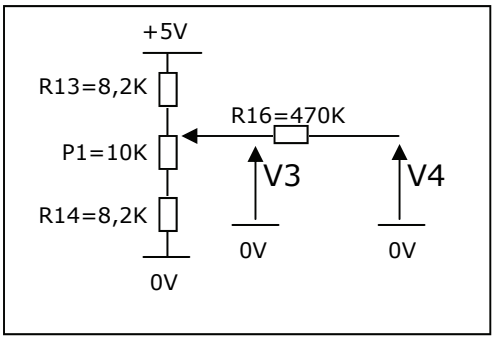
$V3_2$ =5. 4×10-2 V

D’après le théorème de superposition :

$V3=V3_1+V3_2$
d’où V3 = 1. 59 V

B. Détermination des seuils - Cas où le potentiomètre P1 est au maximum :

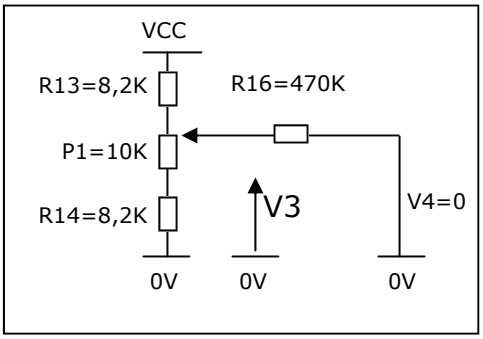
Schéma équivalent:



On utilise le théorème de superposition

Lorsque V4 = 0V :

Schéma équivalent:

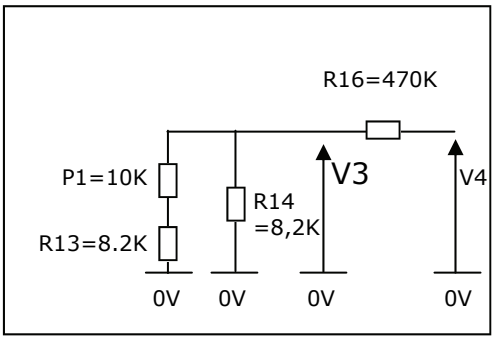


$$V3_1 = \frac{(P1+R14)\times R16 / (P1+R14+R16)}{[(P1+R14)\times R16/(P1+R14+ R14)]+R13} \text{ X VCC}$$

V31 =3.406V

Lorsque VCC = 0V :

Schéma équivalent : (même schéma que précédemment)



$$V3_2 = \frac{(P1+R13)\times R14/(P1+R13+R14)}{(P1+R13)\times R14/(P1+R13+R14)+R16)} \text{ X V4}$$

$V3_2$ =5. 4×10-2 V

D’après le théorème de superposition :

$V3=V3_1+V3_2$
V3 = 3. 465 V

En résumé :

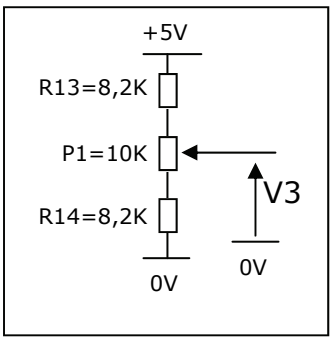
si P1 est à 100%
V3 = 3. 406 V si V4 = 0V
V3 = 3. 465 V si V4 = 5V

si P1 est à 0%
V3 = 1. 53 V si V4 = 0V
V3 = 1. 59 V si V4 = 5V

2.7 Justification de la présence de R15:

Calcul de U3 sans la résistance R15

Schéma équivalent:



Si P1 = 0%

$$V3 = \frac{R14}{(R14+R13+P1)} \times VCC$$

V3 =1. 553 V

Si P1 = 100%

$$V3 = \frac{R14+P1}{(R14+R13+P1)} \times VCC$$

V3=3.447 V

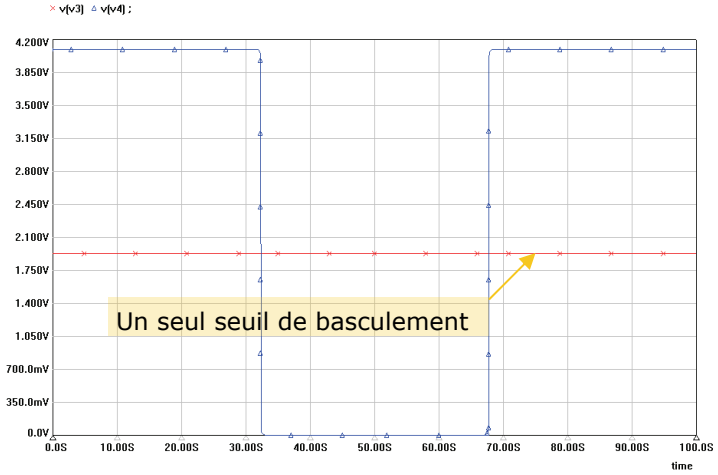
Nous pouvons constater que si l’on enlève la résistance R15, la simulation change. On voit que pour une valeur de P1 il y a qu’une seule valeur de V3.

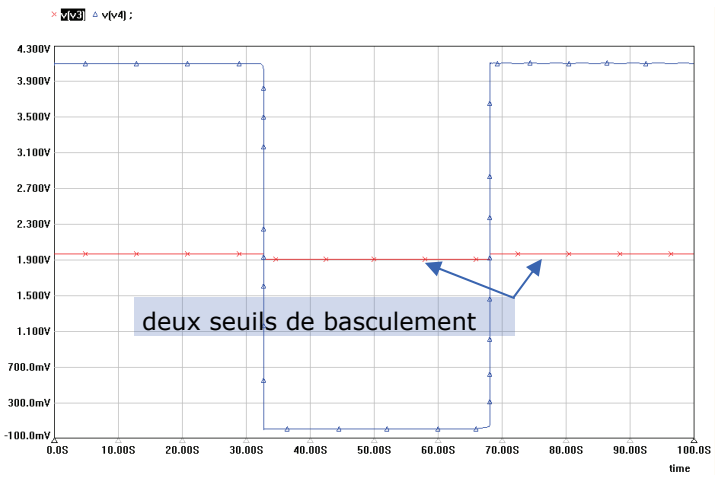
Alors qu’avec R15, la tension U3 dépend de la tension U4 ainsi on obtient deux seuils de tension pour une seule valeur de P1.

L’écart entres les deux seuils distants valant :

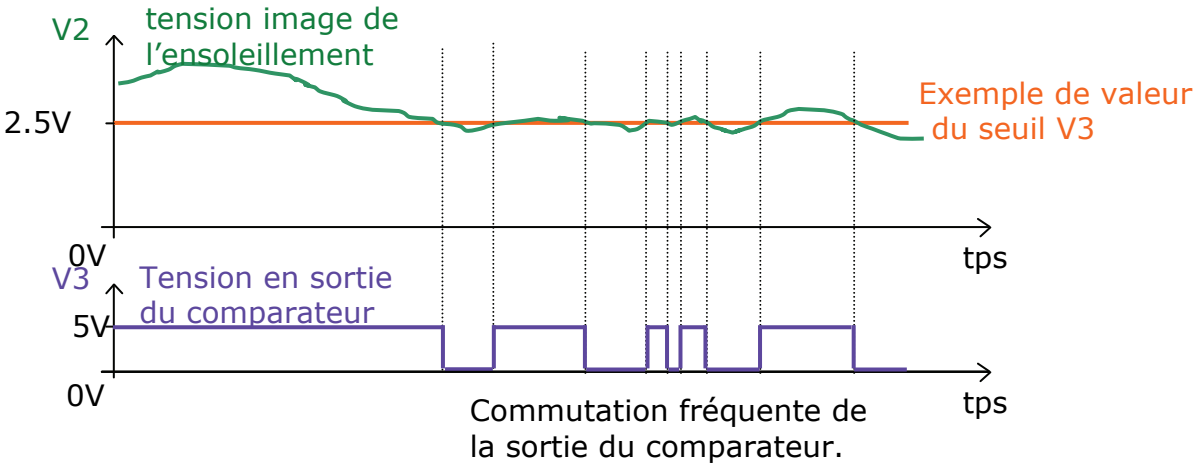
1.59-1.53 V = 0.61V.

Nous pouvons comprendre plus facilement à l’aide des graphiques suivants :



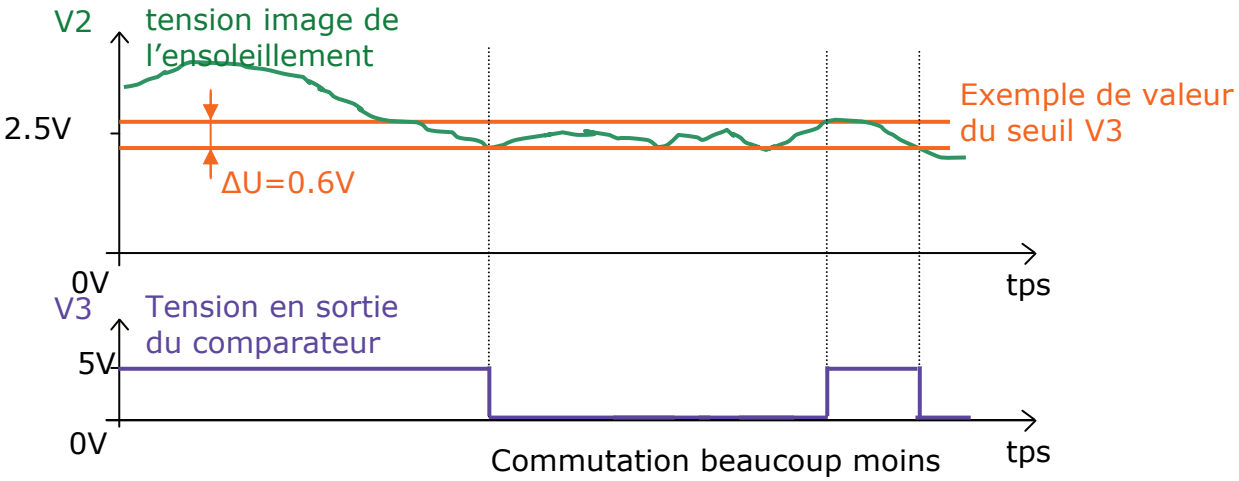


Cas où il n'y a pas de résistance R15 : (un seul seuil)



Commutation fréquente de la sortie du comparateur.

Cas où il y a la résistance R15 : (deux seuils)

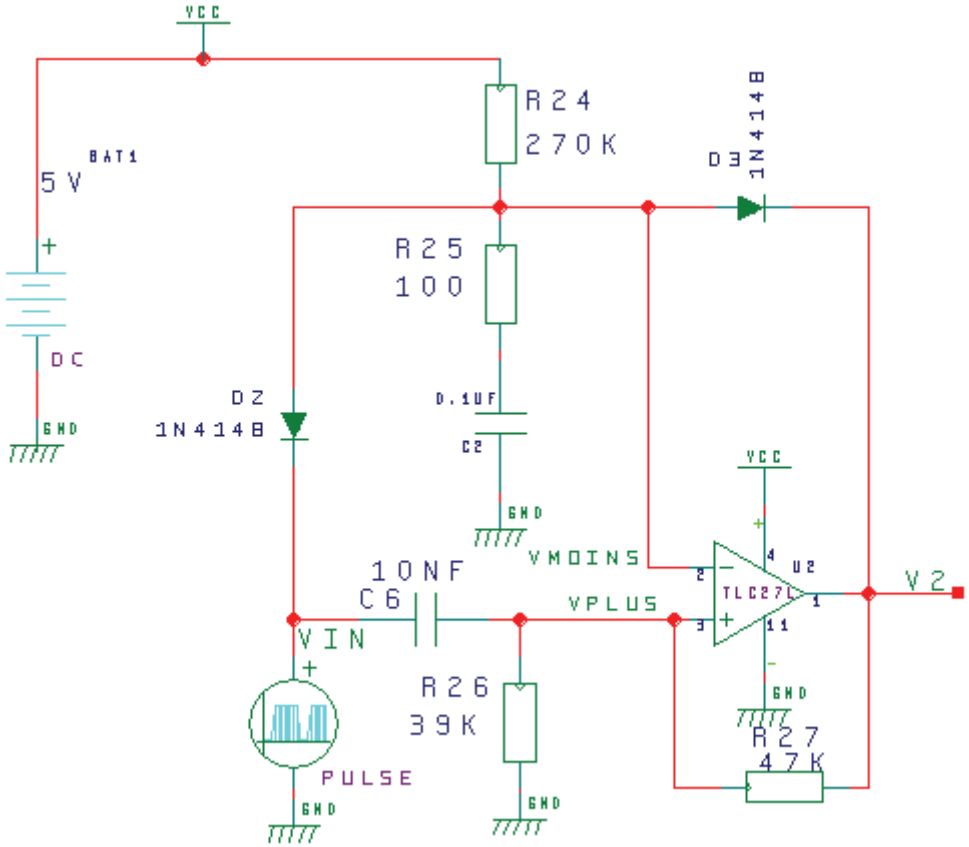


Commutation beaucoup moins fréquente de la sortie du comparateur. Le store est moins sollicité inutilement.

Conclusion :
La résistance R15 permet que les petites variations de l'éclairement ne fassent pas déclencher intempestivement la sortie du comparateur.

ETUDE DE LA CHAINE CAPTEUR VENT
(séquence 3)

Lors de cette séquence nous avons pu voir le fonctionnement de la chaîne d'acquisition vent. Nous avons vu qu'il y a en entrée de FP3 une tension avec des impulsions non calibrées en durée. La durée de ces impulsions dépend de la vitesse du vent. Cette tension est traitée pour qu'il en ressorte de FP3 une tension calibrée en durée (Tw);Ce qui permet d'avoir une tension dont la valeur moyenne image de la vitesse du vent.



3.1 Mode de fonctionnement :

Le mode de fonctionnement du montage construit autour de l'AIL U2 est le mode non linéaire.
Celui-ci est non linéaire car il n'y a pas de contre réaction entre l'entrée moins et la sortie de l'AIL.

Cet AIL fonctionne en comparateur.

3.2 Seuil de déclenchement :

- Cas où V2 = 0V :

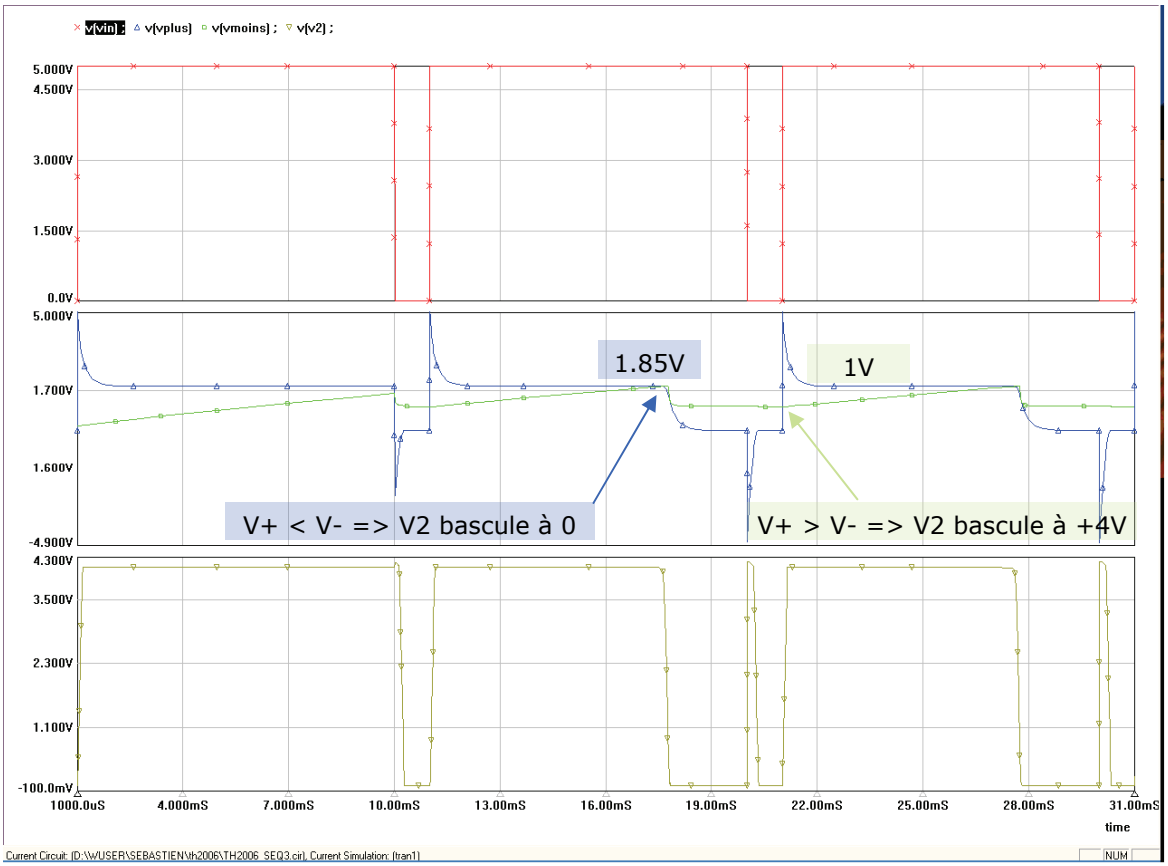
Si V2 = 0V alors Vplus < Vmoins

Lorsque V2= 0V alors la tension Vmoins = 1V.
Ce qui correspond à la tension de seuil de la diode D3.
D3 étant passante car nous avons la cathode qui est à 0V et l'anode à 1V.

Nous savons que la sortie du comparateur basculera lorsque Vplus > Vmoins.

Nous pouvons donc en déduire que le seuil de déclenchement est de 1V.

Nous pouvons justifier cela avec ces chronogrammes :



- Cas où V2 = 5V :

Si V2 = +5V alors Vplus > Vmoins

Déjà nous pouvons remarquer que la valeur pratique de la tension V2 vaut 4.09V.

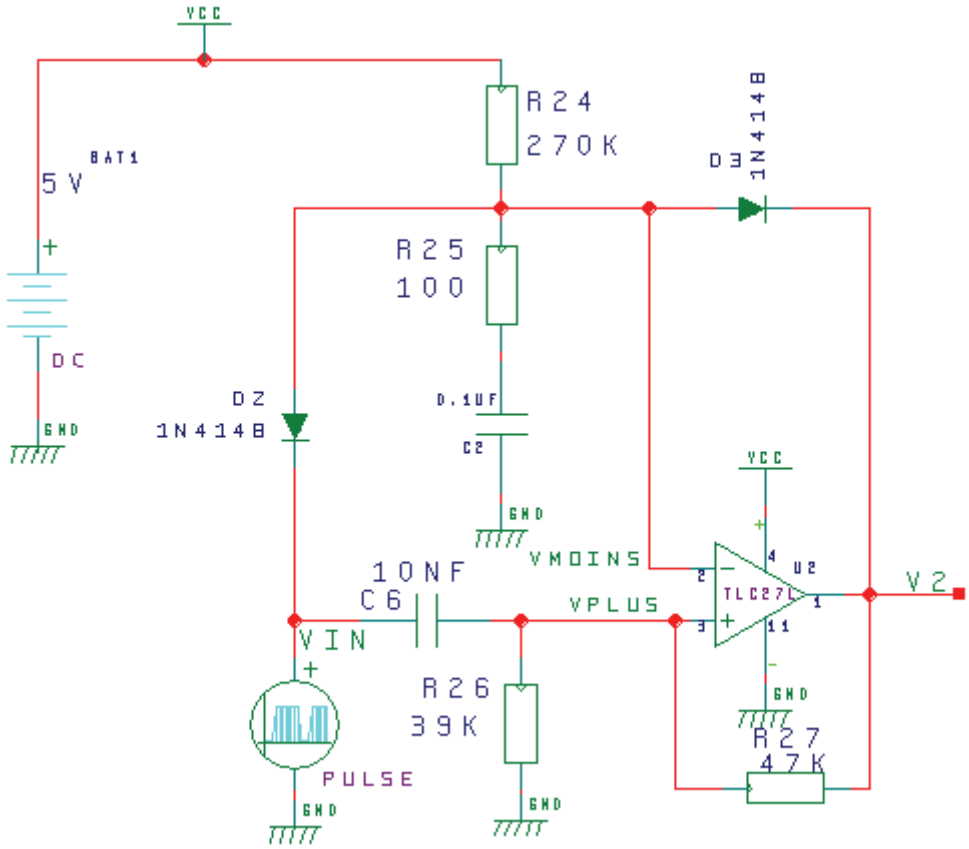
Cela provient du fait que le comparateur n'est pas parfait. En conséquence il y a une tension de déchet.

Nous savons que pour le déclenchement du comparateur il faut qu'il y est Vmoins > Vplus.
La seule chose pouvant faire varier Vplus étant alors la tension VIN, qui produit des impulsions de 5V . Puisque entre VIN et Vplus il y a un condensateur C6 et que celui-ci ne peut varier brusquement nous allons retrouver ces impulsions pour Vplus qui par la suite se stabilisera à 1.85V après la charge de C6.

Vplus = V2 (R26/(R26+R27))
Vplus =1.85V

Ces résultats se vérifient par simulation et sont observables sur les graphes précédents.

3.3 Evolution de Vmoins en fonction du temps :



Pour l'évolution de la tension Vmoins en fonction du temps nous pouvons remarquer que la tension du condensateur correspond à Vmoins car D3 est bloquée et la résistance R25 est très faible.

La tension Vmoins évolue exponentiellement en fonction du temps pour théoriquement tendre vers VCC .

La constante de temps se calcule en fonction de R24,R25 et C2

Tc = (R24+R25)C2
Tc =27ms

3.4 Durée de l'impulsion du monostable :

L'impulsion du monostable qui correspond au temps que met Vmoins pour passer de 1V à 1.85V se calcule à partir de l'équation suivante :

Formule de base: $Uc_{(t=t2)} = Uc_{(t->+\infty)} + (Uc_{(t=t1)} - Uc_{(t->+\infty)})\, e^{-t/t2}$

$Uc2_{(t=t2)} = 1{,}85V + (1V-4{,}09V)\, e^{-tw/tc}$

Avec $Uc_{(t=t2)} = 1.85V$.

On cherche t2 (ou tw : durée de l'impulsion):

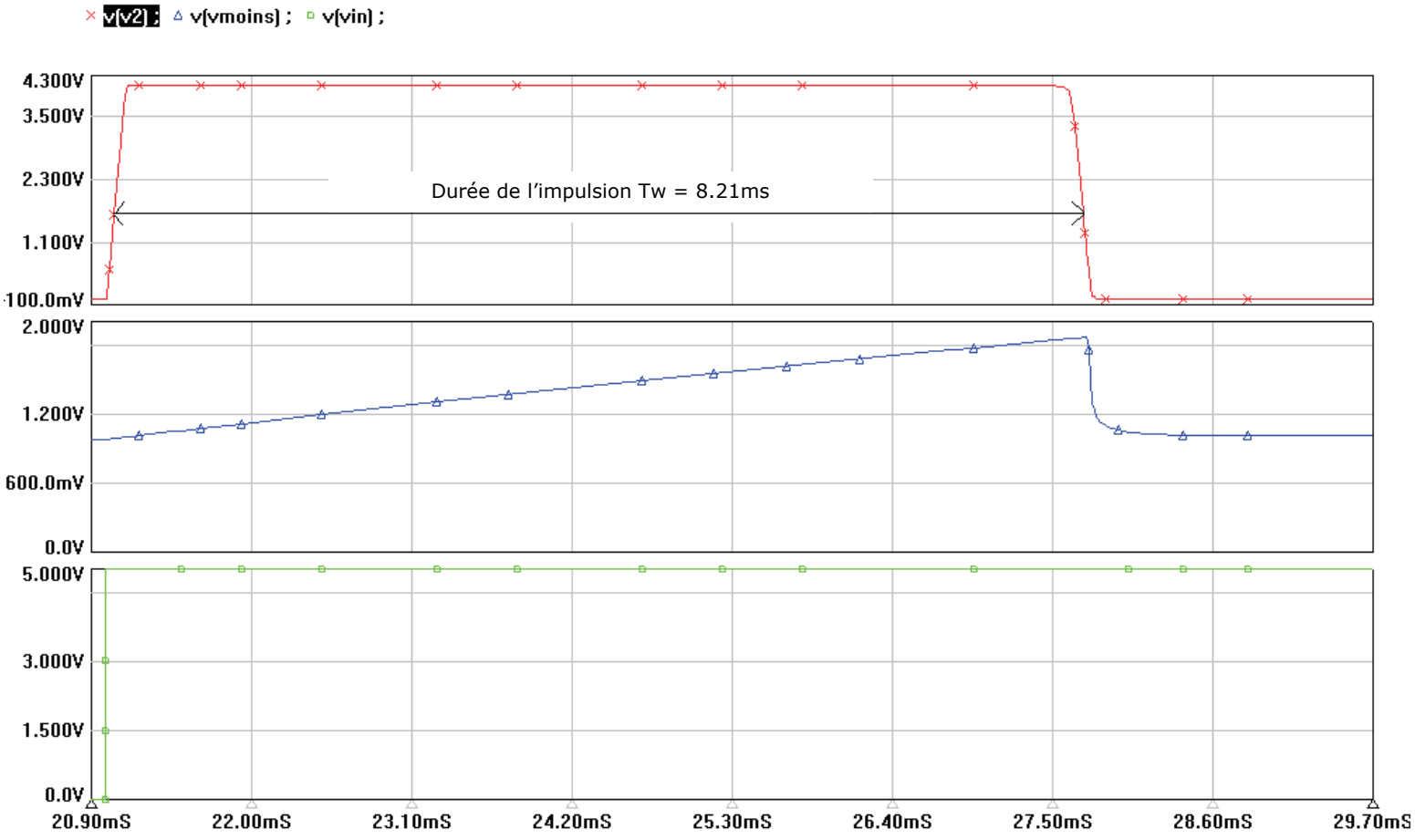
$(1.85-4{,}09)/(1-4{,}09) = e^{-tw/tc}$

$\ln[(1.85-4{,}09)/(1-4{,}09)] = \ln e^{-tw/tc}$

$-tw/tc = \ln 0.725$

$tw = -tc \ln 0.725$

tw = 8.68ms



3.5 Cas où la période du signal de commande est inférieure à la durée de l'impulsion générée :

Nous pouvons remarquer que si la période du signal de commande du monostable est inférieure à la durée de l'impulsion générée la tension V2 reste toujours à l'état haut.
On peut on déduire que le monostable est redéclenchable .

3.6 Justification du choix du constructeur :

En reprenant le schéma structurel complet de la chaîne d'acquisition de la vitesse du vent nous pouvons nous rendre compte qu'il y a déjà trois autre ALI utilisés. Ce qui permet de dire que pour des raisons économiques le constructeur a choisit ce montage ainsi il a pu utiliser tout ses l'ALI car il sont par quatre en général dans un circuit. .

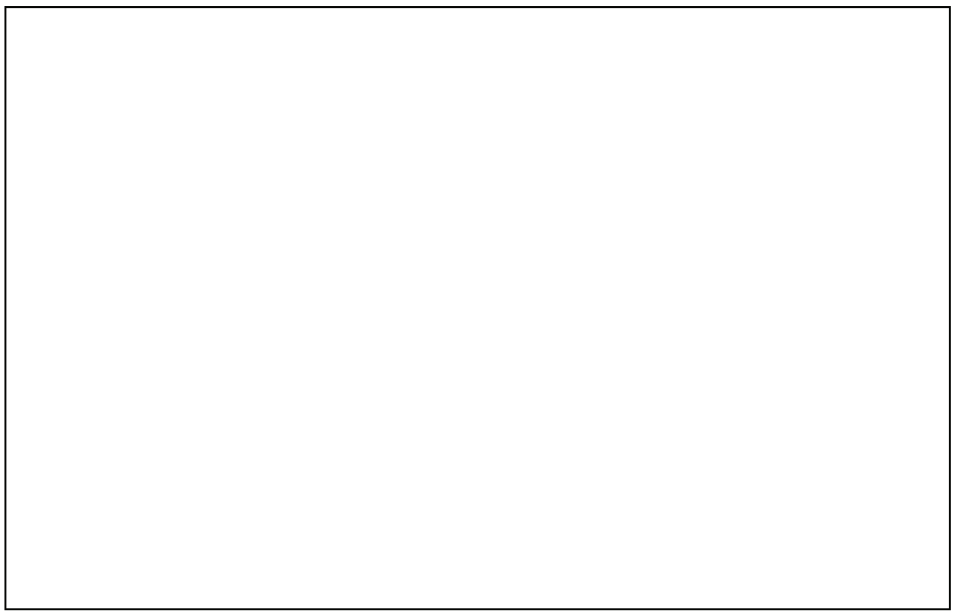
REALISATION DE LA CARTE
(séquence 4)

Lors de cette séquence nous avons conçu le typon. Ceci afin de pouvoir fabriquer la carte. Pour la conception du typon nous avons dû respecter des consignes (taille de la carte, distance d'isolement,…)
Nous avons ensuite testé la carte, pour voir si elle fonctionnait.

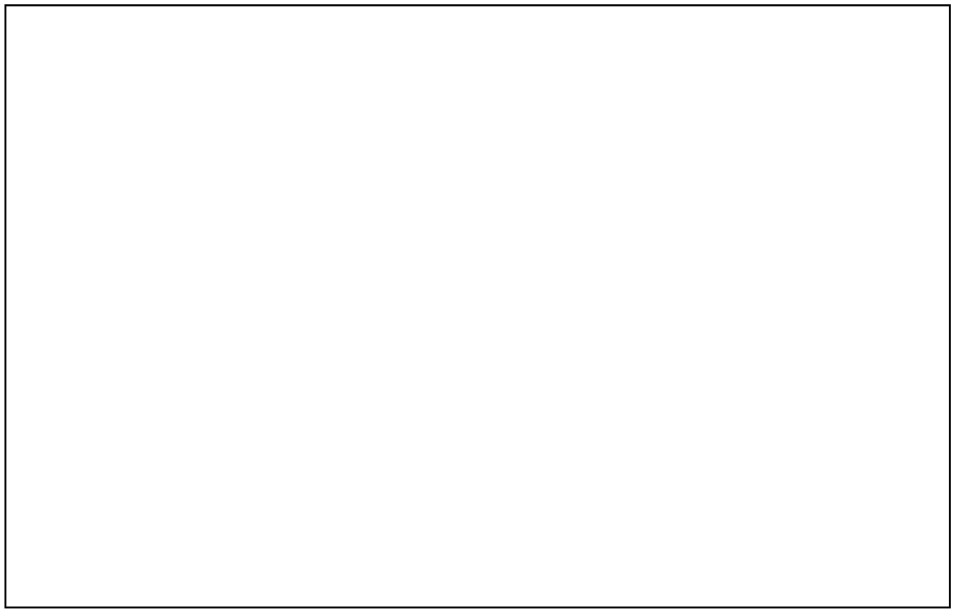
4.1 Conception du typon

Pour concevoir le typon nous avons effectuer le routage à partir du logiciel LayoPcb afin d'obtenir ces calques.

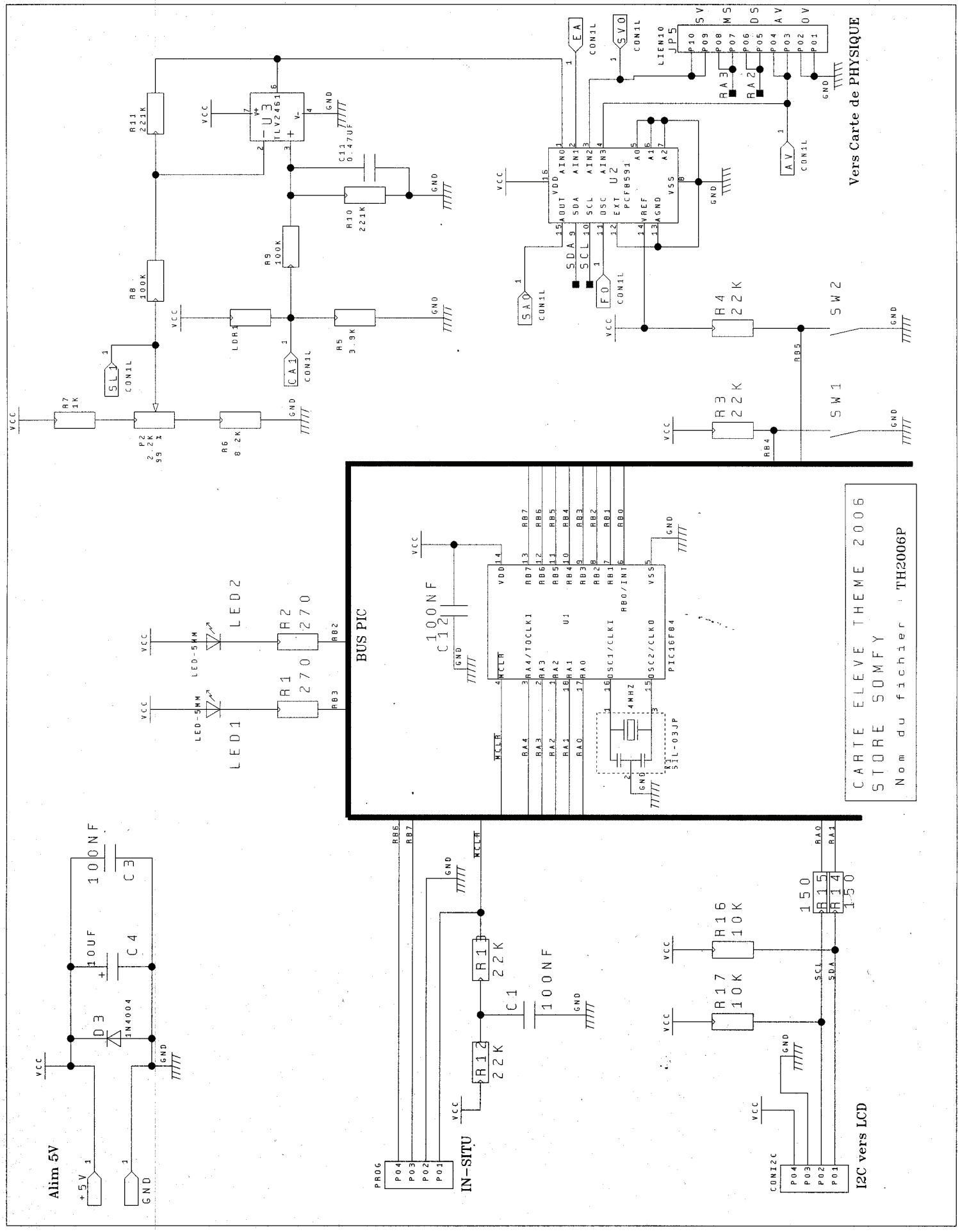
Typon coté Cuivre



Typon coté Composant



4.2 Schéma structurel utilisé pour la réalisation de la carte électronique



4.3 Nomenclature des composants de la carte

Désignation	Valeur	Repère	Nombre
Résistances 1/4 W	10K	R16, R17	2
Résistances 1/4 W	22K	R3,R4,R12,R13	4
Résistances 1/4 W	270	R1,R2	2
Résistances 1/4 W	221K	R10,R11	2
Résistances 1/4 W	150	R14,R15	2
Résistance 1/4 W	1K	R7	1
Résistance 1/4 W	8,2K	R6	1
Résistance 1/4 W	3,9K	R5	1
Condensateur chimique	10µF	C4	1
Condensateur film plastique	0,47µF	C11	1
Condensateur film plastique	100nF	C1,C3,C12	3
Diode	1N4007	D3	1
LED	D5 (verte)	LED2	1
LED	D5 (rouge)	LED1	1
LDR	/	LDR1	1
Potentiomètre	2,2K	P2	1
Résonateur céramique	4MHz	X1	1
Microcontrôleur	16F84	U1	1
CAN/CNA I2C	PCF8591	U2	1
ALI	TLV2461	U3	1
Switch	vert	SW1	1
Switch	rouge	SW2	1
Connecteur cosse pignard	/	CON1L	7
Connecteur carte physique	LIEN10	JP5	1
Connecteur I2C	LIEN4B	CONI2C	1
Connecteur programmation	LIEN4B	PRO	1
Douilles alimentation	2mm	+5V, GND	2

4.4 Vérification de la carte

Après avoir fabriquer la carte ,nous avons vérifié avec l'ohmmètre l'état des pistes, pour savoir si elle n'étaient pas coupées ou en mauvais état.

Nous avons contrôlé à l'ohmmètre la valeur des résistances pour si celles-ci correspondaient bien avec les valeurs données par la nomenclature.

Nous avons vérifié la polarité des composants polarisés(transistors; diodes; boutons ;……..)

Mise sous tension de la carte.

Vérification de tous les points d'alimentation et masse (VCC;GND) à l'aide d'un voltmètre.

PROGRAMMES DE TEST DE LA CARTE
(séquence 5)

Lors de cette séquence nous avons conçu des programmes. Ceci afin de pouvoir tester la cartes, voir si toutes les parties fonctionnent (bus,I2C,CAN, boutons poussoirs, LED…)

5.1 Programmes de test :

A/ Lecture des touches du clavier et allumage des DEL

Ci-dessous contenu du programme associé à ce programme de test avec explications : Sous programme **ihmboudi**

```
////////////////////////////////////////////////////////////////
// Programme de test : ihmboudi.c
////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////
//
// Objet du programme : Le programme ihmboudi a pour but de lire
// l'état des entrées du pic connectées aux 2 boutons poussoirs
// et d'allumer les deux leds correspondantes sur la carte.
//
// L'appui sur le BP Vert impose 0 volt sur l'entrée RB4 du PIC
// L'appui sur le BP Rouge impose 0 volt sur l'entrée RB5 du PIC
//
// L'écriture d'un zéro sur la broche RB2 du PIC allume la led verte
// L'écriture d'un zéro sur la broche RB3 du PIC allume la led rouge
//
////////////////////////////////////////////////////////////////

// Auteurs : --ANONYMAT--

#include   std84.h
#include   bit84.h
#define DMDS portb.4 // demande manuelle de descente du store (BP VERT)
#define DMMS portb.5 // demande manuelle de montée du store (BP ROUGE)
#define VDS portb.2 // à 0 pour visualiser la descente du store (LED VERTE)
#define VMS portb.3 // à 0 pour visualiser la montée du store (LED ROUGE)
#define MasquePortB 0x30 // Masque toutes les lignes du portb sauf RB4 et RB5

////////////////////////////////////////////////////////////////
// PROGRAMME PRINCIPAL
////////////////////////////////////////////////////////////////

void main()
{
      trisb.2=0; // led rouge en sortie
      trisb.3=0; // led verte en sortie
      trisb.4=1; // BP VERT en entrée
      trisb.5=1; // BP ROUGE en  entrée
      ihmboudi(); // APPEL DU SOUS-PROGRAMME Imhboudi
}

////////////////////////////////////////////////////////////////
// SOUS PROGRAMME IHMBOUDI
////////////////////////////////////////////////////////////////


void ihmboudi()
{


      while(1) // boucle infinie
        {
```

```
           if((portb & MasquePortB) == 0x10)  //  cas on l'on appuie uniquement sur bouton
poussoir 1(SW1)
              {
                VDS=0; // DEL rouge allumée
                delay(200);// temporisation
          VDS=1;// DEL rouge éteinte

                delay(200);// temporisation

              }
            else
              {
                VDS=1; // DEL rouge éteinte
              }



           if((portb & MasquePortB) == 0x20) //  cas on l'on appuie uniquement sur bouton
poussoir 2(SW2)
             {
               VMS=0; // DEL verte allumée

               delay(200);// temporisation

          VMS=1;// DEL verte éteinte

                delay(200);// temporisation

             }
            else
              {
              VMS=1;// DEL verte éteinte

              }


           if((portb & MasquePortB) == 0x00)// cas on l'on appuie sur les boutons poussoirs
              {
              VDS=0; // DEL rouge allumée

              delay(200);// temporisation

              VDS=1;// DEL rouge éteinte
              delay(100);// temporisation

            VMS=0;// DEL verte allumée

              delay(200);// temporisation

              VMS=1;// DEL verte éteinte

              delay(100);// temporisation



              }

      }
}

// FIN DU PROGRAMME
```

B/ TEST DU BUS I2C EXTERIEUR ET DES VOYANTS

Ci-dessous contenu du programme associé à ce programme de test avec explications : Sous programme **comcheni**

```
////////////////////////////////////////////////////////////////
// Programme de test : comcheni.c
////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////
//
// Objet du programme : Le programme comcheni a pour but d'établir
// un échange via le bus I2C avec le controleur de port // 8bits PCF8574
// Carte à leds.
//
// Ce programme réalise un chenillard à 8 leds.
//
////////////////////////////////////////////////////////////////

// Auteurs : --ANONYMAT--

#include   std84.h
#include   bit84.h
char indice;

////////////////////////////////////////////////////////////////
// PROGRAMME PRINCIPAL
////////////////////////////////////////////////////////////////

void main()
{
      comcheni(); // Appel du sous-programme comcheni
}


////////////////////////////////////////////////////////////////
// SOUS PROGRAMME COMCHENI
////////////////////////////////////////////////////////////////

void comcheni()
      {


      indice = 0xFE; // affectation de la valeur $FE (11111110) dans la variable indice.

      while(1)    // boucle infinie
      {

      i2c_out_1oct(0x20,indice); // Envoie de la donnée $FE à l'adresse $20 (PCF8574) via le
bus I2C
      delay(200); // attente de 200 ms
         if(indice == 0) // Si toutes les leds sont allumées
             indice=0xFE; // On éteint toutes les leds sauf la première
            else
             indice=indice<<1; // On effectue un décalage de 1 bit à gauche
          }
   }

// FIN DU PROGRAMME
```

C/ TEST DU BUS I2C INTERNE ET LA FONCTION DE GENERATION DE TENSION

Ci-dessous contenu du programme associé à ce programme de test avec explications : Sous programme **cnatrian**

```
////////////////////////////////////////////////////////////////
// Programme de test : cnatrian.c
////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////
//
// Objet du programme : Le programme cnatrian a pour but d'établir
// un échange via le bus I2C de notre maquette avec le
// CAN / CNA PCF8591
//
// Ce programme a pour but d'obtenir en sortie du CNA une tension
// qui évolue en fonction de la progression d'un nombre indice
//
////////////////////////////////////////////////////////////////

// Auteurs : --ANONYMAT--

#include   std84.h
#include   bit84.h
char indice;

////////////////////////////////////////////////////////////////
// PROGRAMME PRINCIPAL
////////////////////////////////////////////////////////////////

void main()
{
     cnatrian(); // Appel du sous-programme cnatrian
}

////////////////////////////////////////////////////////////////
// SOUS PROGRAMME CNATRIAN
////////////////////////////////////////////////////////////////

void cnatrian()

{
     while(1) // Boucle infinie
     {
     for(indice=0;indice<250;indice=indice+10) // executer 25 fois l'instruction suivante en
incrémentant indice de 10 en 10.

        i2c_out_2oct(0x48,0x40,indice); // Lancement d'une CNA à l'adresse $48, avec un octet
de contrôle=$40, et un nombre à convertir = indice

        for(indice=250;indice>0;indice=indice-10) // executer 25 fois l'instruction suivante
en décrémentant indice de 10 en 10.

        i2c_out_2oct(0x48,0x40,indice); // Lancement d'une CNA à l'adresse $48, avec un octet
de contrôle=$40, et un nombre à convertir = indice
        }
}

// FIN DU PROGRAMME
```

D/ TEST DU BUS I2C INTERNE ET DES FONCTIONS DE GENERATION ET DE LECTURE DE TENSION

Ci-dessous contenu du programme associé à ce programme de test avec explications : Sous programme **cnavercan**

```
////////////////////////////////////////////////////////////////
// Programme de test : CNAVERCAN.C
////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////
//
// Objet du programme : Le programme cnavercan a pour but de
// Convertir la tension image de l'intensité lumineuse reçue
// [entrée AIN0] en une valeur numérique (stockée dans resultatcan)
//
// Puis de reconvertir cette valeur numérique en une tension
// en sortie du CNA (sortie SAO)
// Ce programme réalise un chenillard à 8 leds.
//
////////////////////////////////////////////////////////////////

// Auteurs : --ANONYMAT--

#include   std84.h
#include   bit84.h
char resultatcan;

////////////////////////////////////////////////////////////////
// PROGRAMME PRINCIPAL
////////////////////////////////////////////////////////////////

void main()
{
      cnavercan(); // Appel du sous-programme cnavercan
   }

////////////////////////////////////////////////////////////////
// SOUS PROGRAMME CNAVERCAN
////////////////////////////////////////////////////////////////


void cnavercan()

   {
while(1) // Boucle infinie
   {
 resultatcan = i2c_can(0x48); // resultatcan récupère la valeur numérique issue de la CAN

 i2c_out_2oct(0x48,0x40,resultatcan); // Lancement d'une CNA à l'adresse $48, avec un octet de
contrôle=$40, et un nombre à convertir = resultacan

   }
}

// FIN DU PROGRAMME
```

E/ TEST DU BUS I2C EXTERIEUR ET L'AFFICHEUR A CRISTAUX LIQUIDES

Ci-dessous contenu du programme associé à ce programme de test avec explications : Sous programme **defillcd**

```
////////////////////////////////////////////////////////////////
// Programme de test : defillcd.c
////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////
//
// Objet du programme : Le programme defillcd a pour but d'établir
// un échange via le bus I2C de notre maquette avec la carte afficheur
//
// Ce programme a pour but d'afficher sur l'afficheur la série de nombres
// et de caractères suivants :
//
// 0 1 2 3 4 5 6 7 8 9 A B C D E F
//
////////////////////////////////////////////////////////////////

// Auteurs : --ANONYMAT--

#include std84.h
#include bit84.h

char b, b_ascii;

////////////////////////////////////////////////////////////////
// PROGRAMME PRINCIPAL
////////////////////////////////////////////////////////////////

void main()
{trisb=0xF3;     // ($F3=11110011) ->RB2 et RB3 en sortie pour les leds.
defillcd();      // Appel du sous-programme defillcd
}
////////////////////////////////////////////////////////////////
// SOUS PROGRAMME defillcd
////////////////////////////////////////////////////////////////
void defillcd()
{
 delays(1); // attente de 1 seconde
 i2c_lcd(0x70,2,1); // efface l'afficheur (adresse=$70 2=SelectrionModeCommande 1=N°Commande)
 delays(1); // attente de 1 seconde
 i2c_lcd(0x70,2,1); // efface l'afficheur (adresse=$70 2=SelectrionModeCommande 1=N°Commande)
 delays(1); // attente de 1 seconde
 while(1)
  {
   for( b = 0 ; b <= 0xF ; b++ ) // execute 16 fois les instructions suivantes en incrémentant
b de 1 en 1.
     {
       b_ascii=b;          // On affecte à la variable B_ascii le contenu de la variable B
       if (b_ascii>9)            // SI b_ascii est > 9 alors
         b_ascii=b_ascii+0x37;   // b_ascii = b_ascii+$37
       else                // sinon
        b_ascii=b_ascii+0x30;    // b_ascii = b_ascii+$30

       i2c_lcd(0x70,1,b_ascii); // Afficher un caractère (adresse=$70;
1=ModeAffichageCaractère ; Caractère à afficher=b_ascii)
         delay(250);         // attendre 250ms
     }
   i2c_lcd(0x70,2,1);           // efface l'afficheur (adresse=$70 2=SelectrionModeCommande
1=N°Commande)
   delays(1);                   // attendre 1 seconde
  }
}

// FIN DU PROGRAMME
```

E/ INTEGRATION DES PROGRAMMES DE TEST DES FONCTIONNALITES DE BASE DE LA CARTE

Ci-dessous contenu du programme associé à ce programme avec explications :
Programme **reufonct**

```
////////////////////////////////////////////////////////////////
// Programme : reufonct.c
////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////
//
// Objet du programme : Le programme reufonct a pour but
// de réunir l'ensemble des fonctions de test.
//
// Ce programme gére un menu pour choisir le programme de test
// à lancer.
//
////////////////////////////////////////////////////////////////

// Auteurs : --ANONYMAT--

#include std84.h
#include bit84.h
#define MS porta.2          // Commande Moteur : à 1 pour la montée du store
#define DS porta.3          // Commande Moteur : à 1 pour la descente du store
#define DMDS portb.4        // Demande Manuelle de Descente du Store (BP VERT)
#define DMMS portb.5        // demande Manuelle de Montée du Store (BP ROUGE)
#define VDS portb.2         // Visualisation la descente du store (LED VERTE) (actif à 0)
#define VMS portb.3         // Visualisation la montée du store (LED ROUGE) (actif à 0)
#define Visu_CMS Drapeaux.7     // LED de visualisation : Commande du moteur pour Montée du
Store
#define Visu_CDS Drapeaux.6     // LED de visualisation : Commande du moteur pour Descente du
Store
#define Visu_hau Drapeaux.5     // LED de visualisation : Store en position haute
#define Visu_bas Drapeaux.4     // LED de visualisation : Store en position basse
#define VisuDMMS Drapeaux.3     // LED de visualisation : Demande Manuelle de Montée du Store
#define VisuDMDS Drapeaux.2     // LED de visualisation : Demande Manuelle de Descente du Sto-
re
#define VisuDSLu Drapeaux.1     // LED de visualisation : Dépassement Seuil LUmière
#define VisuDSVe Drapeaux.0     // LED de visualisation : Dépassement Seuil Vent
#define FctManuel VariaLogi.3   // variable interne du microcontroleur : Fonctionnement Ma-
nuel/Auto
#define IniPosition VariaLogi.2 // variable interne du microcontroleur : Initialisation posi-
tion store
#define DMS VariaLogi.1    // variable interne du microcontroleur : DMS
#define DDS VariaLogi.0    // variable interne du microcontroleur : DDS
#define MasquePortB 0x30   // Masque toutes les lignes du portb sauf RB4 et RB5
#define cnavercan

char Mesurlum, Mesurven, Seuillum, Drapeaux, Temporaire, VariaLogi;

char indice,exindice;
char a ,b, b_ascii, c;
char resultatcan;

////////////////////////////////////////////////////////////////
// PROGRAMME PRINCIPAL
////////////////////////////////////////////////////////////////

void main()
{
 trisb=0xF3;          // Configuration des E/S du port B (RB2 et RB3 en sortie ;
RB0,RB1,RB4,RB5,RB6,RB7 en entrée <=11110011)
 trisa=0xF3;          // Configuration des E/S du port A (RA2 et RA3 en sortie ;
RA0,RA1,RA4,RA5,RA6,RA7 en entrée <=11110011)
 OPTION.7=0;          // Résistances de tirage interne du port B du PIC
 VMS=1;               // Eteindre la LED de Visualisation de la Montée du Store
```

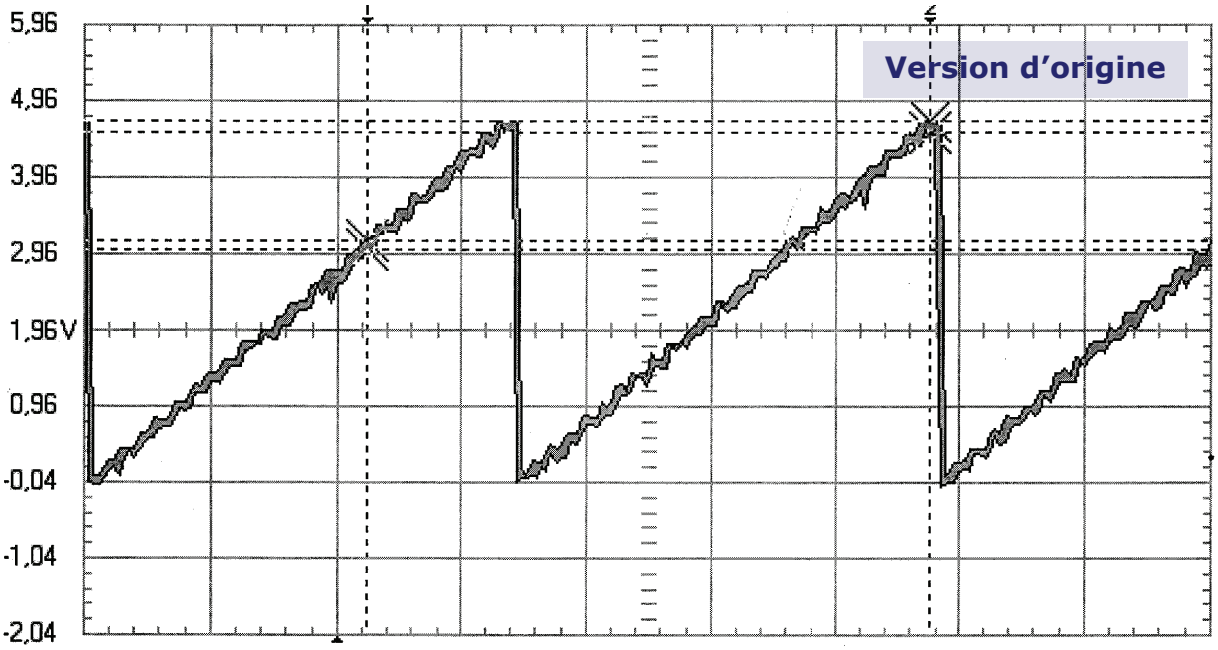
```
VDS=1;              // Eteindre la LED de Visualisation de la Descente du Store
resultatcan=0;
delays(2);          // TEMPORISATION DE 2 SECONDES
i2c_lcd(0x70,2,4);  // Affiche choix entre fonctionnement test et fonctionnement normal
delay(100);         // TEMPORISATION DE 100 ms
while(1)            // BOUCLE INFINIE
 {if (!DMDS)        // Si l'on appuie sur le BP de DMDS alors
  {i2c_lcd(0x70,2,19); // fonctionnement normal, affiche choix entre automatique et manuel
   delay(200);
   while(1)         // BOUCLE INFINIE
   {if (!DMDS)      // Si l'on appuie sur le BP de DMDS alors
    {FctManuel=1;
    fctnorma();// lance le sous programme fctnorma
    }
   }
   else        //sinon
    {
   if (!DMMS)       // Si l'on appuie sur le BP de DMMS alors
       {FctManuel=0;
        fctnorma();  // lance le sous programme fctnorma
    }
   }
  }
 }
 else if (!DMMS)    // SI l'on appuie sur le BP de DMMS alors
  {i2c_lcd(0x70,2,17);    // affiche store 2006 tests
   delay(100);      // temporisation 100ms
   i2c_lcd(0x70,2,5);     // affiche choix entre analogique et logique
   delay(100);      // temporisation 100ms
   while (1)        // BOUCLE INFINIE
    {if (!DMDS)// si l'on appuie sur le BP de DMDS alors
     {i2c_lcd(0x70,2,6); // affiche choix entre ihm local ou sur bus I2C
      delay(200);    //temporisation 200ms
      while (1)// BOUCLE INFINIE
       {if (!DMMS)   //SI l'on appuie sur le BP de DMMS alors
        {i2c_lcd(0x70,2,12); // affiche test touches DEL
         ihmboudi();   // lance le sous programme ihmboudi
        }
       else if (!DMDS)    // Sinon sI l'on appuie sur le BP de DMDS alors
        {i2c_lcd(0x70,2,7); // affiche choix entre caractères ou voyants
         delay(200); //temporisation 200ms
         while (1)   // BOUCLE INFINIE
         {if (!DMDS) //sI l'on appuie sur le BP de DMMS alors
          {i2c_lcd(0x70,2,13); // affiche test voyants i2c
           comcheni();    // lance le sous programme comcheni
          }
          else if (!DMMS) //Sinon sI l'on appuie sur le BP de DMMS alors
           {i2c_lcd(0x70,2,14); // affiche test caracte i2c
            defillcd(); // lance le sous programme defillcd
           }
         }
     }
    }
  }
  else if (!DMMS)    //Sinon sI l'on appuie sur le BP de DMMS alors
   {i2c_lcd(0x70,2,8);    // affiche choix entre génération et recopie d'une tension
    delay(200);//temporisation 200ms
    while (1)        // BOUCLE INFINIE
     {if (!DMMS)     //Sinon sI l'on appuie sur le BP de DMMS alors
      {i2c_lcd(0x70,2,15); // affiche test gene triang
       cnatrian();   // lance le sous programme cnatrian
      }
     else if (!DMDS)  //si l'on appuie sur le BP de DMMS alors
      {i2c_lcd(0x70,2,16); // affiche store 2006 tests
       cnavercan();  // lance le sous programme cnavercan
      }
     }
   }
  }
 }
}
```

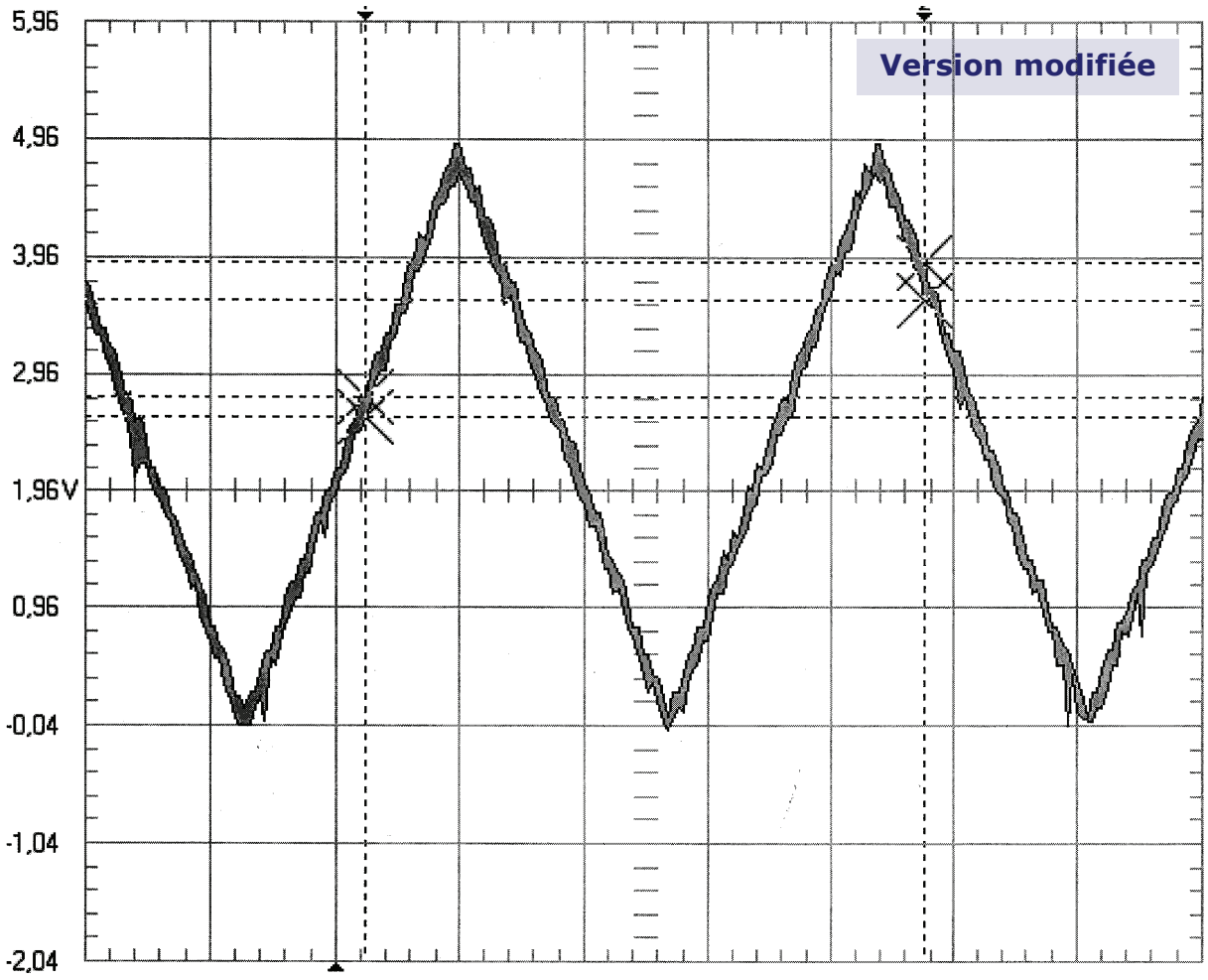
```
}

////////////////////////////////////////////////////////////////
// <FIN> SOUS PROGRAMME PRINCIPAL
////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////
// <DEBUT> SOUS PROGRAMME FCTNORMA
////////////////////////////////////////////////////////////////

void fctnorma()
{
      delays(1); // ATTENTE 1 SECONDE - CE SOUS PROGRAMME N'EST PAS
               // PROGRAMME DANS LA SEQUENCE 5.
}

////////////////////////////////////////////////////////////////
// <FIN> SOUS - PROGRAMME FCTNORMA
////////////////////////////////////////////////////////////////


////////////////////////////////////////////////////////////////
// <DEBUT> SOUS PROGRAMME IHMBOUDI
////////////////////////////////////////////////////////////////

void ihmboudi()
{


      while(1) // boucle infinie
        {

           if((portb & MasquePortB) == 0x10)  //  cas on l'on appuye uniquement sur bouton
poussoir 1(SW1)
              {
                VDS=0; // DEL rouge allumée
                delay(200);// temporisation
          VDS=1;// DEL rouge éteinte

                delay(200);// temporisation

              }
            else
              {
                VDS=1; // DEL rouge éteinte
              }



           if((portb & MasquePortB) == 0x20) //  cas on l'on appuye uniquement sur bouton
poussoir 2(SW2)
             {
               VMS=0; // DEL verte allumée

               delay(200);// temporisation

          VMS=1;// DEL verte éteinte

                delay(200);// temporisation

             }
            else
              {
              VMS=1;// DEL verte éteinte

              }


           if((portb & MasquePortB) == 0x00)// cas on l'on appuye sur les boutons poussoirs
              {
              VDS=0; // DEL rouge allumée
```

```
              delay(200);// temporisation

              VDS=1;// DEL rouge éteinte
              delay(100);// temporisation

             VMS=0;// DEL verte allumée

              delay(200);// temporisation

              VMS=1;// DEL verte éteinte

              delay(100);// temporisation



              }

       }
}

////////////////////////////////////////////////////////////////
// <FIN> SOUS PROGRAMME IHMBOUDI
////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////
// <DEBUT> SOUS PROGRAMME COMCHENI
////////////////////////////////////////////////////////////////

void comcheni()
      {


      indice = 0xFE; // affectation de la valeur $FE (11111110) dans la variable indice.

      while(1)   // boucle infinie
      {

      i2c_out_1oct(0x20,indice); // Envoie de la donnée $FE à l'adresse $20 (PCF8574) via le
bus I2C
      delay(200); // attente de 200 ms
         if(indice == 0) // Si toutes les leds sont allumées
             indice=0xFE; // On éteint toutes les leds sauf la première
            else
             indice=indice<<1; // On effectue un décalage de 1 bit à gauche
          }
   }

////////////////////////////////////////////////////////////////
// <FIN> SOUS PROGRAMME COMCHENI
////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////
// <DEBUT> SOUS PROGRAMME defillcd
////////////////////////////////////////////////////////////////

void defillcd()
{
 delays(1); // attente de 1 seconde
 i2c_lcd(0x70,2,1); // efface l'afficheur (adresse=$70 2=SelectrionModeCommande 1=N°Commande)
 delays(1); // attente de 1 seconde
 i2c_lcd(0x70,2,1); // efface l'afficheur (adresse=$70 2=SelectrionModeCommande 1=N°Commande)
 delays(1); // attente de 1 seconde
 while(1)
  {
   for( b = 0 ; b <= 0xF ; b++ ) // execute 16 fois les instructions suivantes en incrémentant
b de 1 en 1.
     {
       b_ascii=b;           // On affecte à la variable B_ascii le contenu de la variable B
       if (b_ascii>9)            // SI b_ascii est > 9 alors
         b_ascii=b_ascii+0x37;   // b_ascii = b_ascii+$37
```

```
       else                // sinon
        b_ascii=b_ascii+0x30;    // b_ascii = b_ascii+$30

       i2c_lcd(0x70,1,b_ascii); // Afficher un caractère (adresse=$70;
1=ModeAffichageCaractère ; Caractère à afficher=b_ascii)
         delay(250);          // attendre 250ms
     }
   i2c_lcd(0x70,2,1);           // efface l'afficheur (adresse=$70 2=SelectrionModeCommande
1=N°Commande)
   delays(1);                   // attendre 1 seconde
  }
}

////////////////////////////////////////////////////////////////
// <FIN> SOUS PROGRAMME defillcd
////////////////////////////////////////////////////////////////


////////////////////////////////////////////////////////////////
// <DEBUT> SOUS PROGRAMME CNATRIAN
////////////////////////////////////////////////////////////////

void cnatrian()

{
     while(1) // Boucle infinie
     {
     for(indice=0;indice<250;indice=indice+10) // executer 25 fois l'instruction suivante en
incrémentant indice de 10 en 10.

        i2c_out_2oct(0x48,0x40,indice); // Lancement d'une CNA à l'adresse $48, avec un octet
de contrôle=$40, et un nombre à convertir = indice

        for(indice=250;indice>0;indice=indice-10) // executer 25 fois l'instruction suivante
en décrémentant indice de 10 en 10.

        i2c_out_2oct(0x48,0x40,indice); // Lancement d'une CNA à l'adresse $48, avec un octet
de contrôle=$40, et un nombre à convertir = indice
        }
}

////////////////////////////////////////////////////////////////
// <FIN> SOUS PROGRAMME CNATRIAN
////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////
// <DEBUT> SOUS PROGRAMME CNAVERCAN
////////////////////////////////////////////////////////////////

void cnavercan()

   {
while(1) // Boucle infinie
   {
 resultatcan = i2c_can(0x48); // resultatcan récupère la valeur numérique issue de la CAN

 i2c_out_2oct(0x48,0x40,resultatcan); // Lancement d'une CNA à l'adresse $48, avec un octet de
contrôle=$40, et un nombre à convertir = resultacan

   }
   }

////////////////////////////////////////////////////////////////
// <FIN> SOUS PROGRAMME CNAVERCAN
////////////////////////////////////////////////////////////////

// FIN DU PROGRAMME
```

Pour le programme cnatrian, nous avons modifié le programme d'origine pour générer un signal triangulaire pour N variant de 0 à 240, puis de 240 à 0. Comme l'illustre les relevés d'oscillogrammes suivants :





ETUDE DU BUS I2C
(séquence 6)

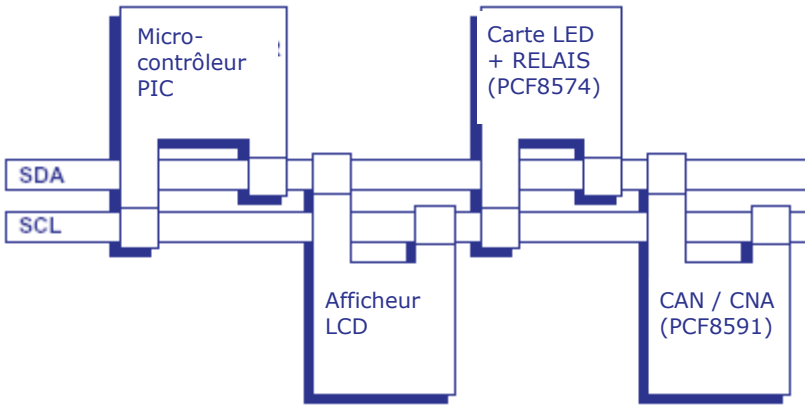
Lors de cette séquence nous avons pu étudier le bus I2C ainsi qu'un CAN/CNA .
Nous avons vus les différentes caractéristiques du bus I2C ainsi que celles du Contrôleur de port // 8bits [PCF8574] et du CAN/CNA [PCF8591]
(plus précisément le PFF8591).
Puis nous avons pu interpréter et conçu différents chronogrammes)

6.1 Caractéristiques générales

Il existe différentes générations du bus I2C. Nous pouvons retrouver différentes générations dans ce tableau :

	Standard-Mode	Fast-Mode	High-Speed-Mode	
Bit Rate (kbits/s)	0 to 100	0 to 400	0 to 1700	0 to 3400
Max Cap Load (pF)	400	400	400	100
Rise time (ns)	1000	300	160	80
Spike Filtered (ns)	N/A	50	10	
Address Bits	7 and 10	7 and 10	7 and 10	

Un bus I2C permet de faire communiquer différents périphériques esclaves à un microcontrôleur (le maitre). Voici la configuration de notre carte et des cartes périphériques.



Pour pouvoir communiquer « correctement », chaque périphérique sur le bus I2C doit posséder une adresse. Dans notre application chaque adresse est composée de 7 bits.

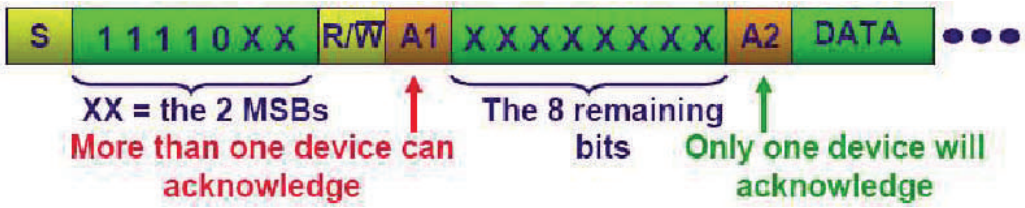
Composant	Adresse
Afficheur LCD	$70
Carte LED+Relais	$20
CAN/CNA	$48

Plus généralement, il est possible de faire communiquer les périphériques entres eux en utilisant deux grands types d'adressage sur le bus I2C.

Il existe l'adressage 7 bits :

L'adresse est composé de 7 bits. Ces bits sont transmis juste après le bit de start (poids fort en premier).

Il existe l'adressage 10 bits :

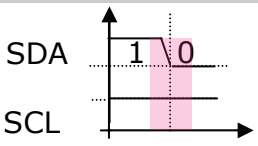
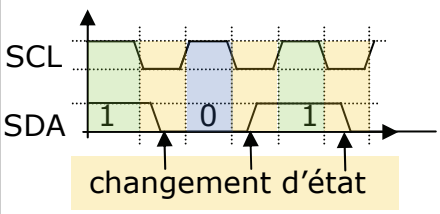


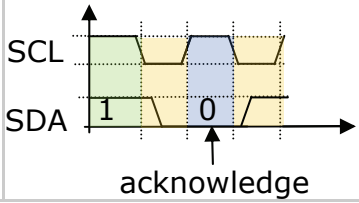
L'adresse est composé de 5 premiers bits fixes (11110) caractérisant le mode d'adressage 10 bits suivis des 2 bits de poids fort. Ces bits sont transmis juste après le bit de start (poids fort en premier). Les 8 bits suivants de l'adresse sont transmis dans un deuxième octet (poids faible en dernier).

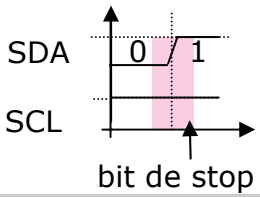
Sur un bus I2C il y a deux signaux SDA et SCL.
SDA est un fil bidirectionnel de transmission de données.
SCL est un fil qui permet le cadencement des transferts. Il est émis par le maître.

REMARQUE : le signal SCL ne peut pas être considéré comme une horloge traditionnelle.

Quatre informations élémentaires caractérisent l'échange d'informations sur ces deux signaux.

Bit de start Qui correspond à un basculement du signal SDA de 1 à 0 quand SCL est à 1.	
Echange des données (ou de l'adresse) Ce qui caractérise le bus I2C c'est que les changement de SDA se font sur l'état bas de SCL. Ils sont validés sur l'état haut de SCL.	

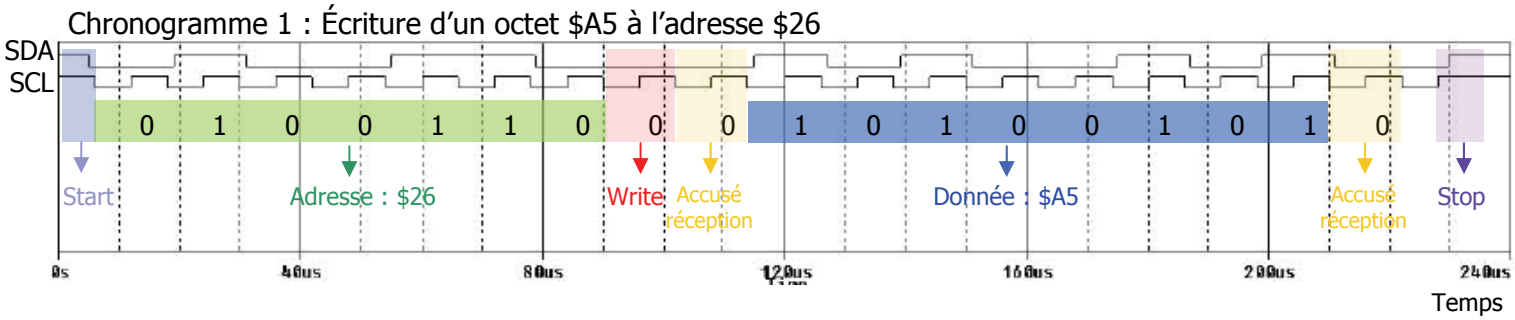
Bit d'acknowledge (accusé de réception) Ce qui caractérise le bus I2C c'est que les changement de SDA se font sur l'état bas de SCL. Ils sont validés sur l'état haut de SCL.	

Bit de stop Qui correspond à un basculement du signal SDA de 0 à 1 quand SCL est à 1.	

REMARQUE : le signal SCL ne peut pas être considéré comme une horloge traditionnelle d'un système numérique à base de microprocesseur, car le signal n'est pas périodique. Ce signal est commandé par le maître, et si l'esclave ne répond pas le signal reste sur le même état.

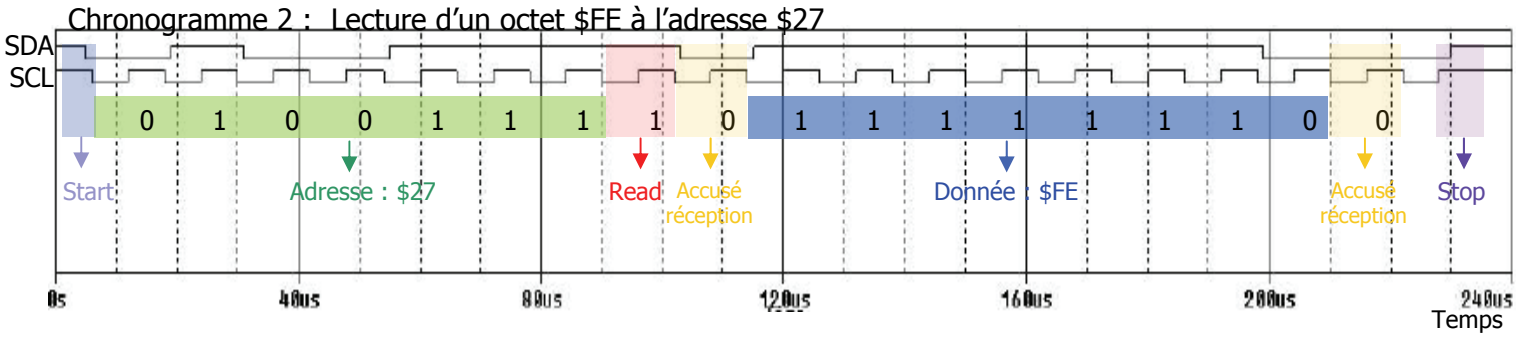
6.2 Interprétation des chronogrammes

Chronogramme 1 :



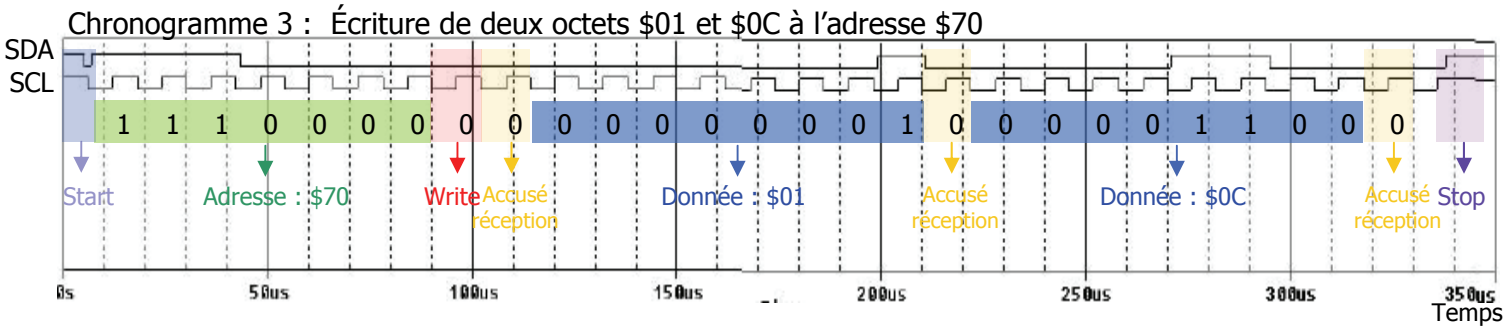
Sur ce chronogramme nous pouvons voir le bit de **start** ; **l'adresse $26** en hexadécimal ; le bit **d'écriture** ; **l'acknowledge (accusé de réception)** envoyé par l'esclave pour confirmer la liaison; la **donnée $A5** en hexadécimal envoyée par le maître ; **l'acknowledge** envoyé par l'esclave pour confirmer la réception de la donnée et enfin le bit de **stop** pour arrêter l'échange de données.

Chronogramme 2 :



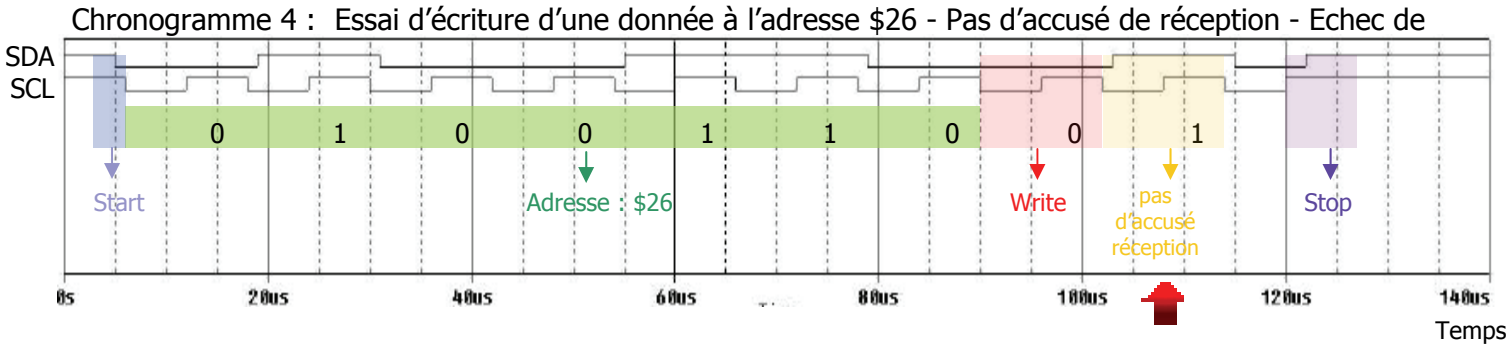
Sur ce chronogramme nous pouvons voir le bit de **start** ; **l'adresse $27** en hexadécimal ; le bit **de lecture** ; **l'acknowledge (accusé de réception)** envoyé par l'esclave pour confirmer la liaison; la **donnée $FE** en hexadécimal envoyée par l'esclave ; **l'acknowledge** envoyé par le maître pour confirmer la réception de la donnée et enfin le bit de **stop** pour arrêter l'échange de données.

Chronogramme 3 :



Sur ce chronogramme nous pouvons voir le bit de **start** ; **l'adresse $70** en hexadécimal ; le bit **d'écriture** ; **l'acknowledge (accusé de réception)** envoyé par l'esclave pour confirmer la liaison; la **donnée $01** en hexadécimal envoyée par le maître ; **l'acknowledge** envoyé par l'esclave pour confirmer la réception de la donnée, puis la **donnée $0C** en hexadécimal envoyée par le maître ; **l'acknowledge** envoyé par l'esclave pour confirmer la réception de la donnée et enfin le bit de **stop** pour arrêter l'échange de données.

Chronogramme 4 :



Sur ce chronogramme nous pouvons voir le bit de **start** ; **l'adresse $26** en hexadécimal ; le bit **de lecture** ; **l'acknowledge (accusé de réception) n'est pas envoyé** par l'esclave pour dire qu'il y a eu un problème lors de la transmission de données. Cela peut provenir du fait que l'esclave n'est pas connecté ou bien que l'esclave est endommagé ou tout simplement que l'adresse n'existe pas.

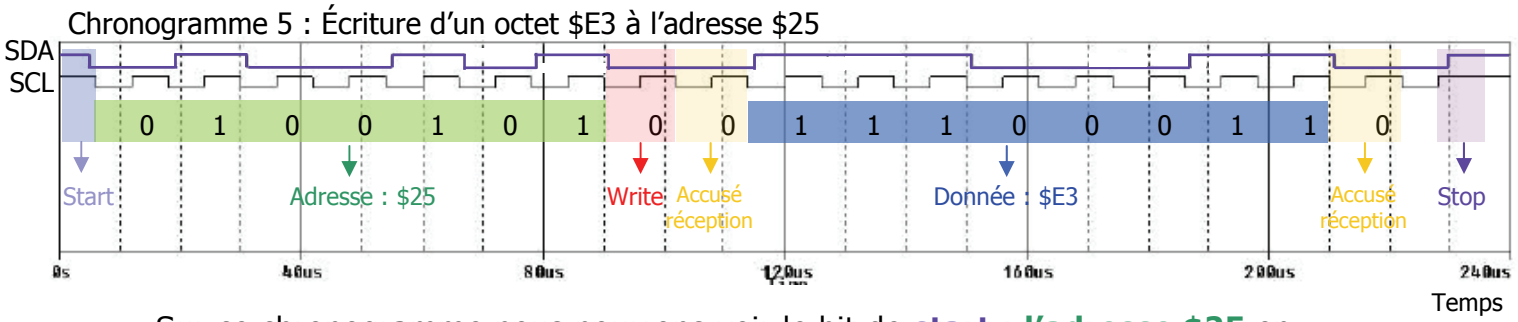
6.3 Détermination des chronogrammes d'un PCF8574

Dans la documentation constructeur nous pouvons voir que l'adresse est décomposables en deux parties :
-> les bits A6 à A3 sont fixes. C'est d'ailleurs de cette façon que nous pouvons savoir que les informations échangées sont destinées à un PCF8574.
-> Les bits A2 à A0 sont eux variables. Chaque bit est configurable par l'application d'un « 1 » ou d'un « 0 » sur une entrée d'adresse correspondante.

Pour résumer nous pouvons dire que l'adresse d'un PCF8574 peut varier de $20 à $27 en hexadécimal (soit 7 périphériques identiques sur le même bus).

Chronogramme 5:

Sachant que l'adresse de base est $20 en hexadécimal. Si nous rajoutons la valeur 5 en décimal, nous obtenons **l'adresse $25** en hexadécimal.

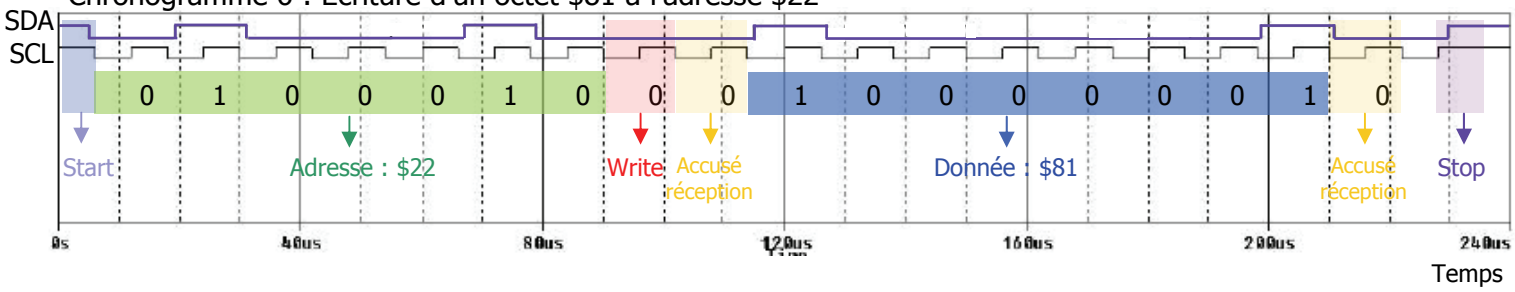


Sur ce chronogramme nous pourrons voir le bit de **start** ; **l'adresse $25** en hexadécimal ; le bit **d'écriture** ; **l'acknowledge (accusé de réception)** pour confirmer la liaison; la **donnée $E3** en hexadécimal ; **l'acknowledge** pour confirmer la réception de la donnée et le bit de **stop** pour arrêter l'échange de données.

Chronogramme 6:

Sachant que l'adresse de base est $20 en hexadécimal. Si nous rajoutons la valeur 2 en décimal, nous obtenons **l'adresse $22** en hexadécimal.

Chronogramme 6 : Écriture d'un octet $81 à l'adresse $22



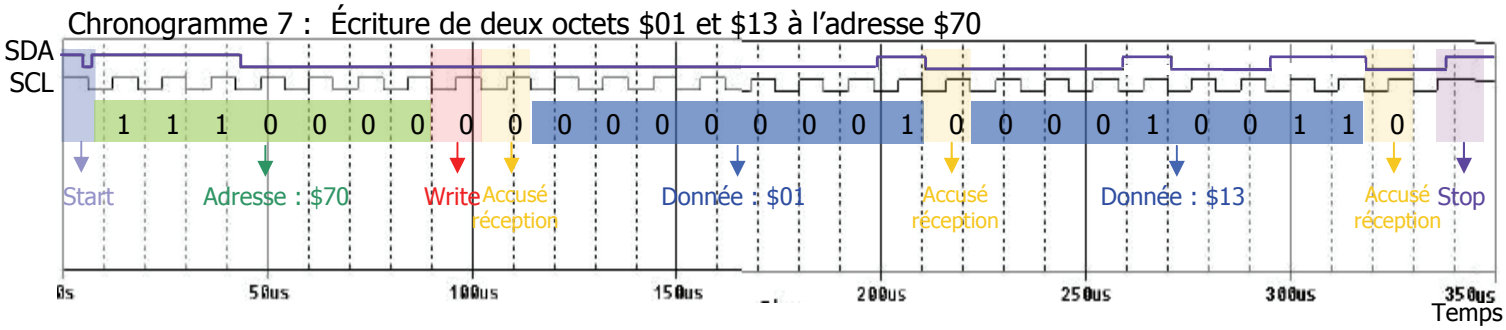
Sur ce chronogramme nous pourrons voir le bit de **start** ; **l'adresse $22** en hexadécimal ; le bit **d'écriture** ; **l'acknowledge (accusé de réception)** pour confirmer la liaison; la **donnée $81** en hexadécimal ; **l'acknowledge** pour confirmer la réception de la donnée et le bit de **stop** pour arrêter l'échange de données.

6.4 Détermination des chronogrammes d'un afficheur LCD.

Tout d'abord nous savons pour commencer que l'adresse de l'afficheur LCD est en hexadécimal $70. Nous savons aussi que l'afficheur doit à chaque fois recevoir deux octets.

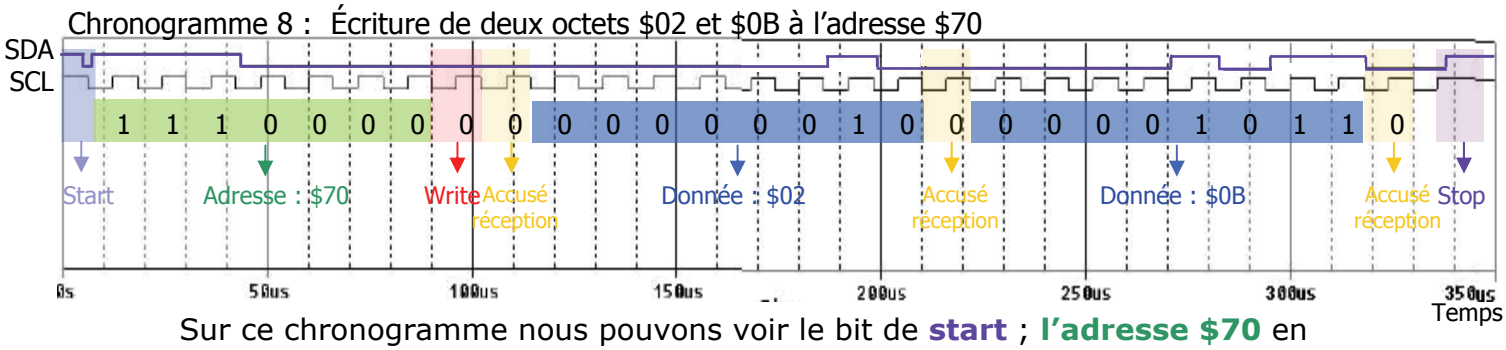
Chronogramme 7:

Chronogramme 7 : Écriture de deux octets $01 et $13 à l'adresse $70



Sur ce chronogramme nous pouvons voir le bit de **start** ; **l'adresse $70** en hexadécimal ; le bit **d'écriture** ; **l'acknowledge (accusé de réception)** envoyé par l'esclave pour confirmer la liaison; la **donnée $01** en hexadécimal envoyée par le maître ; **l'acknowledge** envoyé par l'esclave pour confirmer la réception de la donnée, puis la **donnée $13** en hexadécimal (19 en décimal) envoyée par le maître ; **l'acknowledge** envoyé par l'esclave pour confirmer la réception de la donnée et enfin le bit de **stop** pour arrêter l'échange de données.

Chronogramme 8:



Sur ce chronogramme nous pouvons voir le bit de **start** ; **l'adresse $70** en hexadécimal ; le bit **d'écriture** ; **l'acknowledge (accusé de réception)** envoyé par l'esclave pour confirmer la liaison; la **donnée $02** en hexadécimal envoyée par le maître ; **l'acknowledge** envoyé par l'esclave pour confirmer la réception de la donnée, puis la **donnée $0B** en hexadécimal (11 en décimal) envoyée par le maître ; **l'acknowledge** envoyé par l'esclave pour confirmer la réception de la donnée et enfin le bit de **stop** pour arrêter l'échange de données.

6.5 Caractéristiques et commande d'un PCF8591

Il existe pour le mode de Conversion Analogique Numérique (CAN) quatre entrées analogiques (AIN0 à AIN3). Et pour le mode de Conversion Numérique Analogique (CNA) une sortie Analogique AOUT.

Conversion Numérique Analogique		
Caractéristique	MIN	MAX
Tension de sortie	VSS=0V	VDD=5V
Erreur d'offset		50mV
Erreur de gain		1%
Erreur de linéarité		+/-1.5 LSB
Fréquence de conversion		11.5 KHz

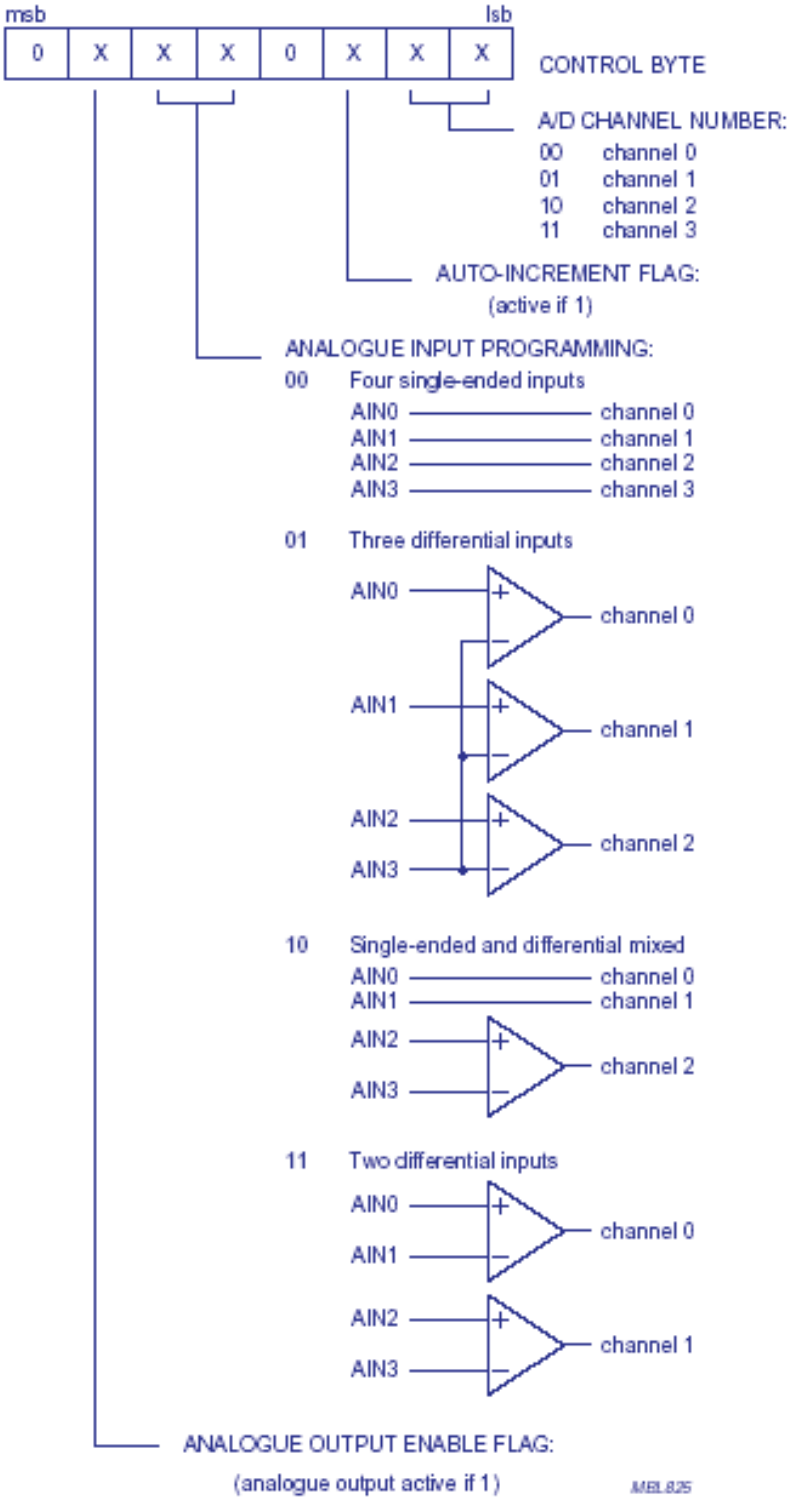
Conversion Analogique Numérique		
Caractéristique	MIN	MAX
Tension d'entrée	VAGND=0V	VREF=VCC=5V
Erreur d'offset		20mV
Erreur de gain		1%
Erreur de gain si ΔV<16LSB		5%
Erreur de linéarité		+/-1.5 LSB
Fréquence de conversion		11.5 KHz

Ci dessous voici les différents mode de conversion :

-> 3 modes différentiel : pour lesquels les entrées fonctionnent en mode différentielles (deux à deux).

->1 mode indépendant : Pour lequel les 4 entrées sont séparées et sont référencées par rapport à V_{AGND}.

On peut choisir ces différents modes en configurant l'octet de contrôle.

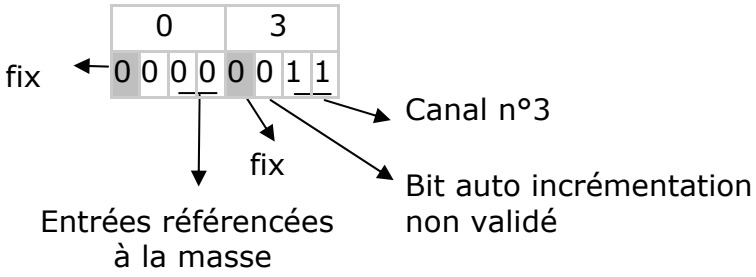


Exemple d'échange pour réaliser une conversion ANALOGIQUE/ NUMERIQUE :

Conversion analogique numérique d'une tension présente sur le canal 3.

IL faut envoyer une trame de configuration du CAN :
Pour cela il faut envoyer une trame comprennant un bit de start, l'adresse du CAN $48 puisque c'est un PCF8591 ; le bit d'écriture ; l'acknowledge ; l'octet de contrôle ;l'acknowledge et le bit de stop.

L'octet de contrôle aura pour valeur :



Puis il faut envoyer une trame de lecture du résultat de la conversion :

Cette trame est composée d'un bit de start ; l'adresse $48 ; le bit de lecture ; l'acknowledge ; un octet de donnée ; un acknowledge ;…….... et le bit de stop.

Les octets de conversion se suivent. Ils arrivent régulièrement, en fonction de la cadence imposée par le maître.

Dans la documentation la valeur de la tension V_{AOUT} générée par le CNA en fonction des signaux V_{REF} et V_{AGND} est donnée par la formule :

MODE C N A :

$$V_{AOUT} = V_{AGND} + \frac{(V_{REF} - V_{AGND})}{256} \sum_{i=0}^{i=7} D_i \times 2^i$$

Nous pouvons en déduire la plus petite tension (quantum) qui peut être générée au dessus de $V_{AGND.}$

V_{REF} = 5V et V_{AGND}= 0V d'une part:

V_{AOUT}=0+ ((5-0)/256) x 1
V_{AOUT} = quantum = 19.5mV

V_{REF} = 3.5V et V_{AGND}=1.2V d'autre part:

V_{AOUT}=1.2+((3.5-1.2)/256) x 1
V_{AOUT} =1.209V

Calcul de V_{AOUT}:

Si le nombre N vaut 135, la sortie V_{AOUT} vaut :
avec V_{REF} = 5V et V_{AGND} = 0V

V_{AOUT} = 0+((5-0)/256)x135
V_{AOUT}=2.637V

Si le nombre N vaut 203, la sortie V_{AOUT} vaut :
avec V_{REF} = 3.5V et V_{AGND} = 1.2V

V_{AOUT} = 1.2+((3.5-1.2)/203) x 1
V_{AOUT}=3.04V

Dans la documentation la valeur numérique N générée par le CAN en fonction du signal V_{AIN} et des signaux V_{REF} et V_{AGND} est donnée par la formule :

MODE C A N :

$$\textbf{Valeur numérique N} = \frac{V_{AIN}-V_{AGND}}{V_{REF}-V_{AGND}} \times 256$$

Exemple :

Si V_{AIN}=2.9V ; V_{REF} =5V et V_{AGND}=0V

alors N = (2.9-O)/(5-0) x 256
N =148

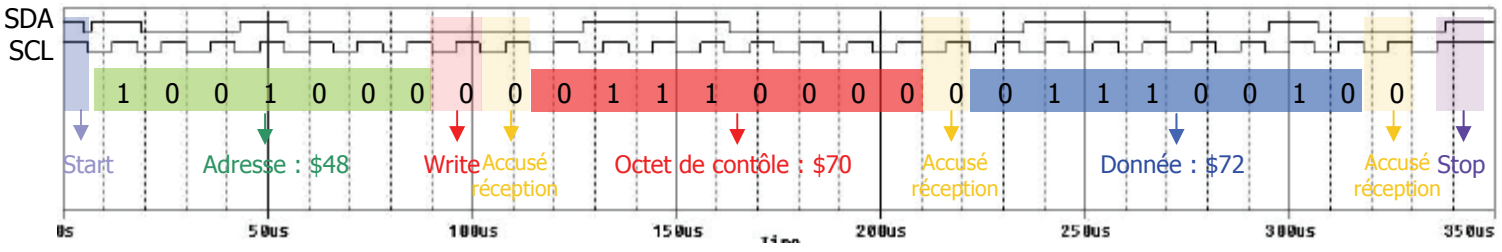
Si V_{AIN}=2.9V ; V_{REF} =3.5V et V_{AGND}=1.2V

alors N = (2.9-1.2)/(3.5-1.2) X 256
N **=189**

6.6 Interprétation des chronogrammes d'un PCF8591

Sur ce chronogramme nous pouvons voir le bit de **start** ; **l'adresse $48** en

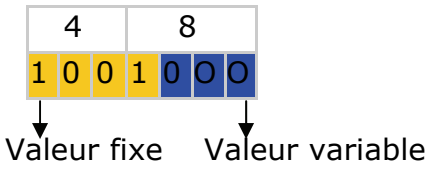
Chronogramme 9 : Écriture de deux octets $70 et $72 à l'adresse $48



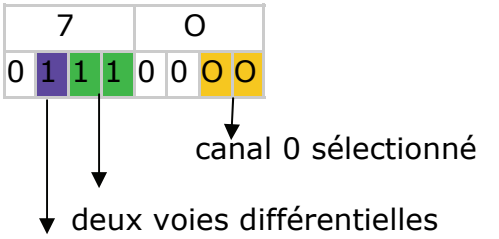
hexadécimal ; le bit **d'écriture** ; **l'acknowledge (accusé de réception)** envoyé par l'esclave pour confirmer la liaison; l'octet de contrôle $70 en hexadécimal envoyée par le maître ; **l'acknowledge** envoyé par l'esclave pour confirmer la réception de la donnée, puis la **donnée $72** en hexadécimal envoyée

par le maître ; **l'acknowledge** envoyé par l'esclave pour confirmer la réception de la donnée et enfin le bit de **stop** pour arrêter l'échange de données.

Pour l'adresse on a:



Pour les paramètres qui sont appliqués à travers le registre de contrôle on a :



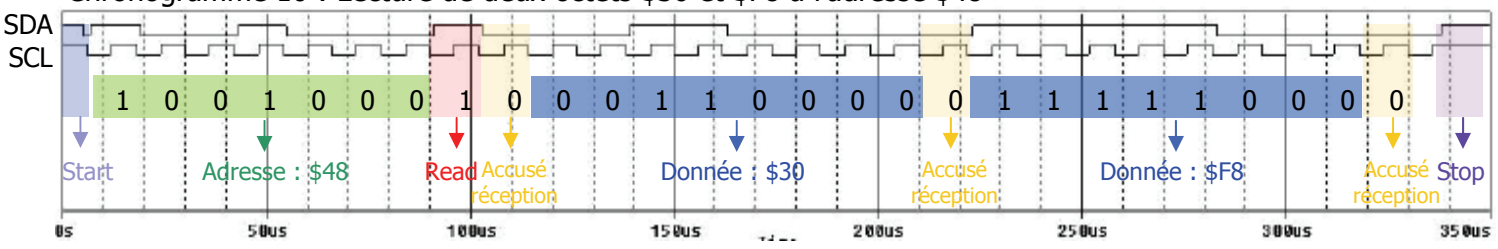
Puisque nous sommes en écriture c'est une conversion N/A
Nous avons la donnée $71 en hexadécimal qui donne en décimal113

Calcul de Vaout:

Si le nombre N vaut 114, la sortie V_{AOUT} vaut :
avec V_{REF} = 5V et V_{AGND} = 0V

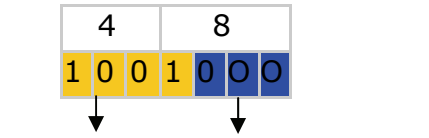
V_{AOUT} = 0+((5-0)/256)x114
V_{AOUT}=2.227V

Chronogramme 10 : Lecture de deux octets $30 et $F8 à l'adresse $48



Sur ce chronogramme nous pouvons voir le bit de **start** ; **l'adresse $48** en hexadécimal ; le bit **lecture** ; **l'acknowledge (accusé de réception)** envoyé par l'esclave pour confirmer la liaison; la **donnée $30** en hexadécimal envoyée par l'esclave ; **l'acknowledge** envoyé par le maître pour confirmer la réception de la donnée, puis la **donnée $F8** en hexadécimal envoyée par l'esclave ; **l'acknowledge** envoyé par le maître pour confirmer la réception de la donnée et enfin le bit de **stop** pour arrêter l'échange de données.

Pour l'adresse on a:



Puisque nous sommes en lecture c'est une conversion A/N
Nous avons la donnée $30 en hexadécimal qui donne en décimal 48.
Puis nous avons la donnée $F8 en hexadécimal qui donne en décimal 248

Calcul de Vaout:

Si le nombre N vaut 48, la sortie V_{AOUT} vaut :
avec V_{REF} = 5V et V_{AGND} = 0V

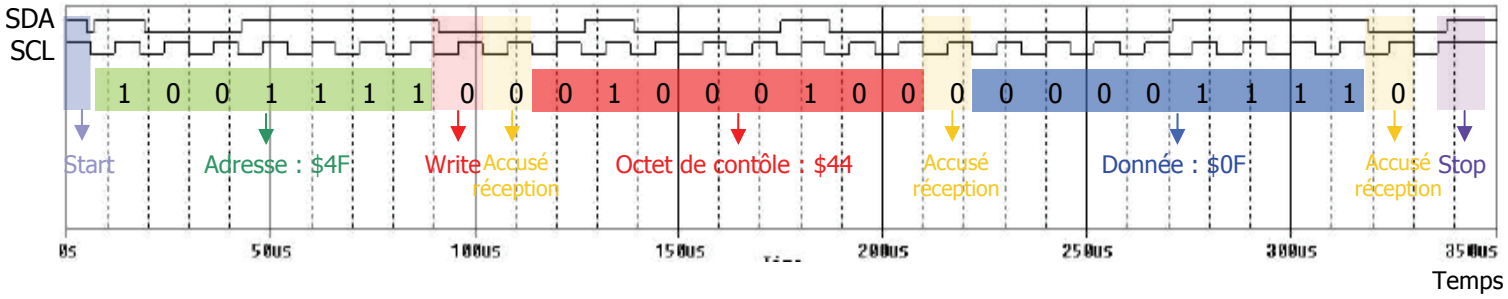
V_{AOUT} = 0+((5-0)/256)x48
V_{AOUT}=O.9375V

Calcul de Vaout:

Si le nombre N vaut 248, la sortie V_{AOUT} vaut :
avec V_{REF} = 5V et V_{AGND} = 0V

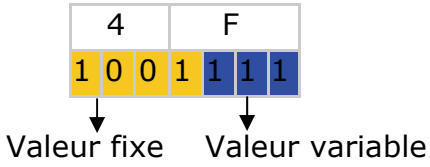
V_{AOUT} = 0+((5-0)/256)x248
V_{AOUT}=4.844V

Chronogramme 13 : Écriture de deux octets $44 et $0F à l'adresse $4F

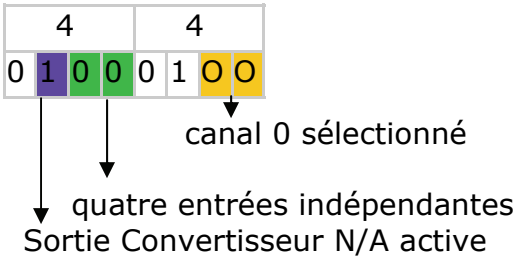


Sur ce chronogramme nous pouvons voir le bit de **start** ; **l'adresse $4F** en hexadécimal ; le bit **d'écriture** ; **l'acknowledge (accusé de réception)** envoyé par l'esclave pour confirmer la liaison; l'octet de contrôle $44 en hexadécimal envoyée par le maître ; **l'acknowledge** envoyé par l'esclave pour confirmer la réception de la donnée, puis la **donnée $0F** en hexadécimal envoyée par le maître ; **l'acknowledge** envoyé par l'esclave pour confirmer la réception de la donnée et enfin le bit de **stop** pour arrêter l'échange de données.

Pour l'adresse on a:



Pour les paramètres qui sont appliqués à travers le registre de contrôle on :



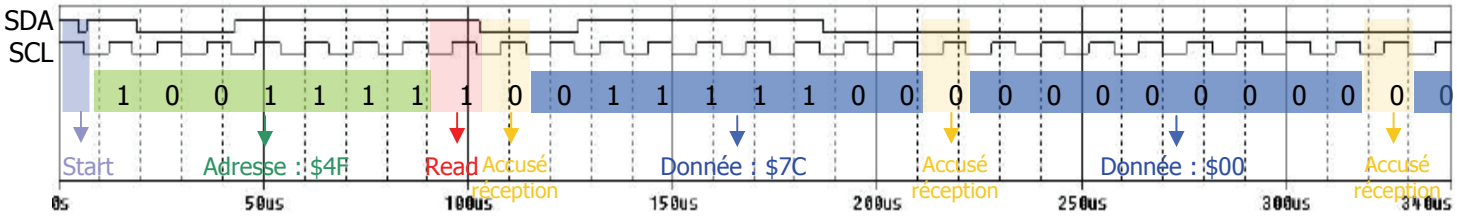
Puisque nous sommes en écriture c'est une conversion N/A
Nous avons la donnée $OF en hexadécimal qui donne en décimal15

Calcul de Vaout:

Si le nombre N vaut 15, la sortie V_{AOUT} vaut :
avec V_{REF} = 5V et V_{AGND} = 0V

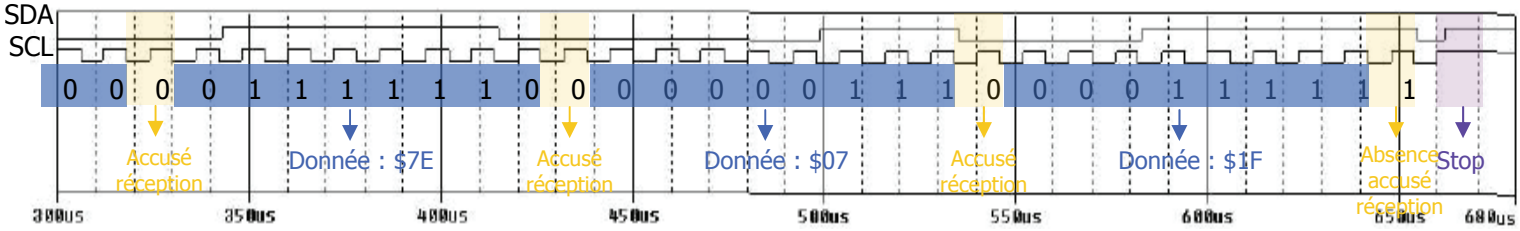
V_{AOUT} = 0+((5-0)/256)x15
V_{AOUT}=0.293V

Chronogramme 14 : Première partie de la trame de lecture des octets $7C, $00, $7E, $07 et $1F à l'adresse



Temps

Chronogramme 15 : Deuxième partie de la trame de lecture des octets $7C, $00, $7E, $07 et $1F à l'adresse

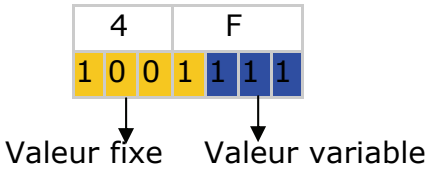


Temps

Sur ce chronogramme nous pouvons voir le bit de **start** ; **l'adresse $4F** en hexadécimal ; le bit **lecture** ; **l'acknowledge (accusé de réception)** envoyé par l'esclave pour confirmer la liaison; la **donnée $7C** en hexadécimal envoyée par l'esclave ; **l'acknowledge** envoyé par le maître pour confirmer la réception

de la donnée, puis la **donnée $OO** en hexadécimal envoyée par l'esclave ; **l'acknowledge** envoyé par le maître pour confirmer la réception de la donnée, puis la **donnée $7E** en hexadécimal envoyée par l'esclave ; **l'acknowledge** envoyé par le maître pour confirmer la réception de la donnée, puis la **donnée $O7** en hexadécimal envoyée par l'esclave ; **l'acknowledge** envoyé par le maître pour confirmer la réception de la donnée, enfin la **donnée $1F** en hexadécimal envoyée par l'esclave ; **pas d'acknowledge** envoyé par le maître pour confirmer la réception de la donnée ; mais il y a le bit de **stop** pour arrêter l'échange de données.

Pour l'adresse on a:



Puisque nous sommes en lecture c'est une conversion A/N
Nous avons la donnée $7C en hexadécimal qui donne en décimal 124.
Nous avons la donnée $OO en hexadécimal qui donne en décimal 00
Nous avons la donnée $7E en hexadécimal qui donne en décimal 126.
Nous avons la donnée $07 en hexadécimal qui donne en décimal 7
Nous avons la donnée $1F en hexadécimal qui donne en décimal 31

Calcul de Vaout:

Si le nombre N vaut 124, la sortie V_{AOUT} vaut :
avec V_{REF} = 5V et V_{AGND} = 0V

V_{AOUT} = 0+((5-0)/256)x124
V_{AOUT}=2.42V

Calcul de Vaout:

Si le nombre N vaut 0, la sortie V_{AOUT} vaut :
avec V_{REF} = 5V et V_{AGND} = 0V

V_{AOUT} = 0+((5-0)/256)xO
V_{AOUT}=0V

Calcul de Vaout:

Si le nombre N vaut 126, la sortie V_{AOUT} vaut :
avec V_{REF} = 5V et V_{AGND} = 0V

V_{AOUT} = 0+((5-0)/256)x126
V_{AOUT}=2.461V

Calcul de Vaout:

Si le nombre N vaut 7, la sortie V_{AOUT} vaut :
avec V_{REF} = 5V et V_{AGND} = 0V

V_{AOUT} = 0+((5-0)/256)x7
V_{AOUT}=0.1367V

Calcul de Vaout:

Si le nombre N vaut 31, la sortie V_{AOUT} vaut :
avec V_{REF} = 5V et V_{AGND} = 0V

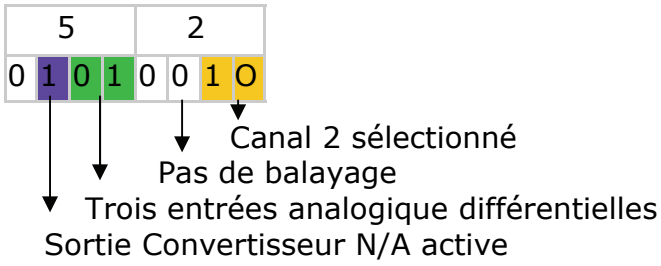
V_{AOUT} = 0+((5-0)/256)x31
V_{AOUT}=0.605V

6.7 Détermination des chronogrammes d'un PCF8591

Chronogramme 16

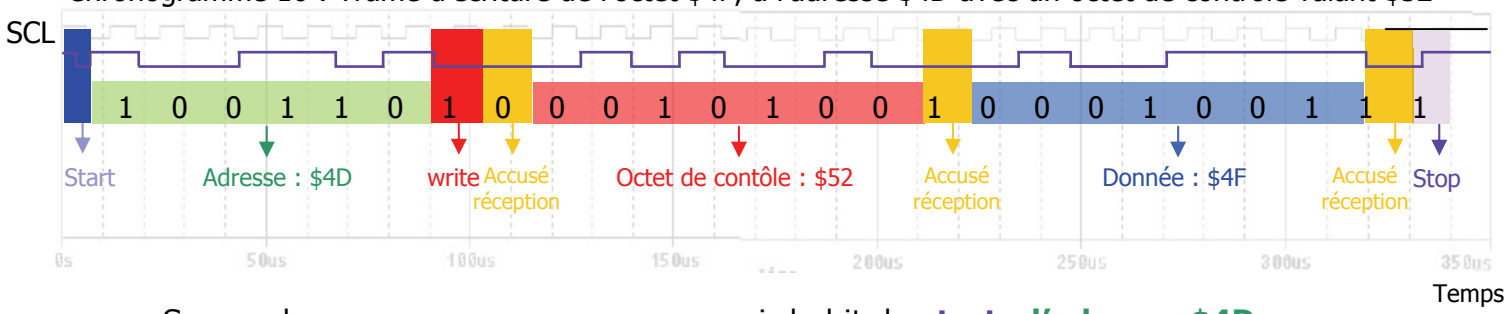
Sachant que l'adresse de base est $48 en hexadécimal. Si nous rajoutons la valeur 5 en décimal, nous obtenons **l'adresse $4D** en hexadécimal.

Pour les paramètres qui sont appliqués à travers le registre de contrôle on :



La valeur en décimal 79 correspond à $4F en hexadécimal.

Chronogramme 16 : Trame d'écriture de l'octet $4F, à l'adresse $4D avec un octet de contrôle valant $52

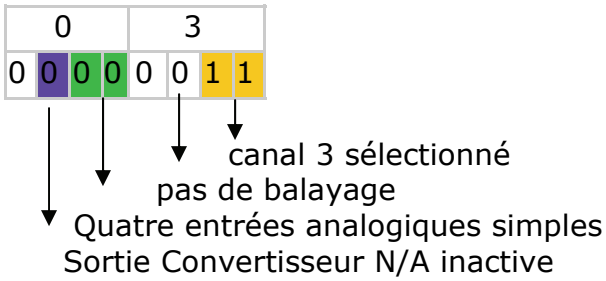


Sur ce chronogramme nous pourrons voir le bit de **start** ; **l'adresse $4D** en hexadécimal ; le bit **d'écriture** ; **l'acknowledge (accusé de réception)** pour confirmer la liaison; l'octet de contrôle $52 en hexadécimal ; **l'acknowledge** pour confirmer la réception de la donnée;la **donnée $4F** en hexadécimal et le bit de **stop** pour arrêter l'échange de données.

Chronogramme 11

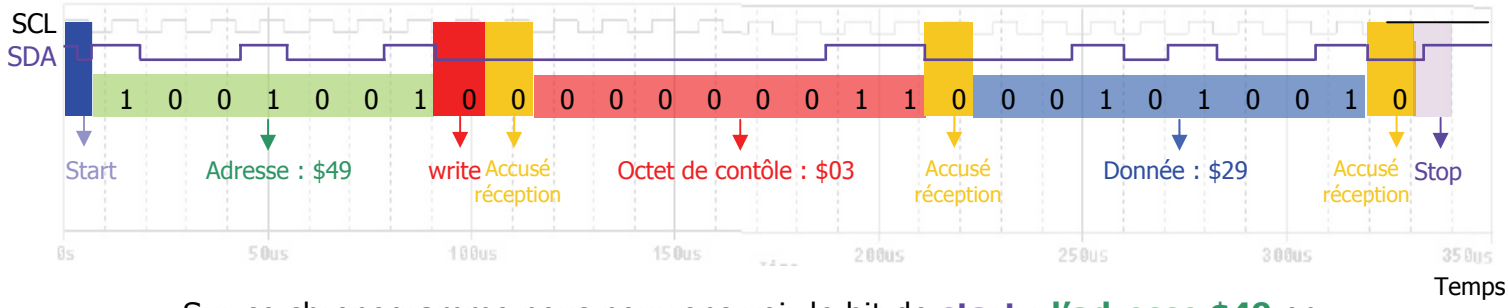
Sachant que l'adresse de base est $48 en hexadécimal. Si nous rajoutons la valeur 1 en décimal, nous obtenons **l'adresse $49** en hexadécimal.

Pour les paramètres qui sont appliqués à travers le registre de contrôle on a:



La valeur en décimal 41 correspond à $29 en hexadécimal.

Chronogramme 11 : Trame d'écriture de l'octet $29, à l'adresse $49 avec un octet de contrôle valant $03

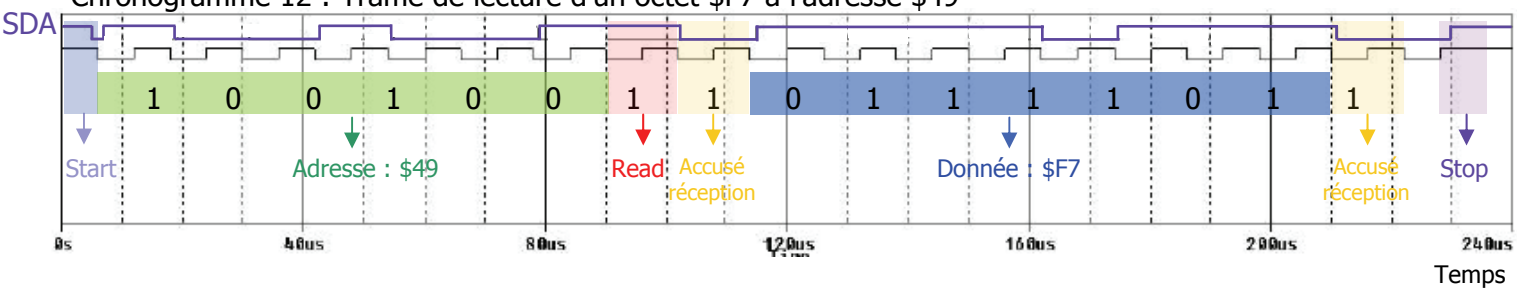


Sur ce chronogramme nous pourrons voir le bit de **start** ; **l'adresse $49** en hexadécimal ; le bit **d'écriture** ; **l'acknowledge (accusé de réception)** pour confirmer la liaison; l'octet de contrôle $03 en hexadécimal ; **l'acknowledge** pour confirmer la réception de la donnée;la **donnée $29** en hexadécimal et le bit de **stop** pour arrêter l'échange de données.

Chronogramme 12

La valeur en décimal 247 correspond à $F7 en hexadécimal.

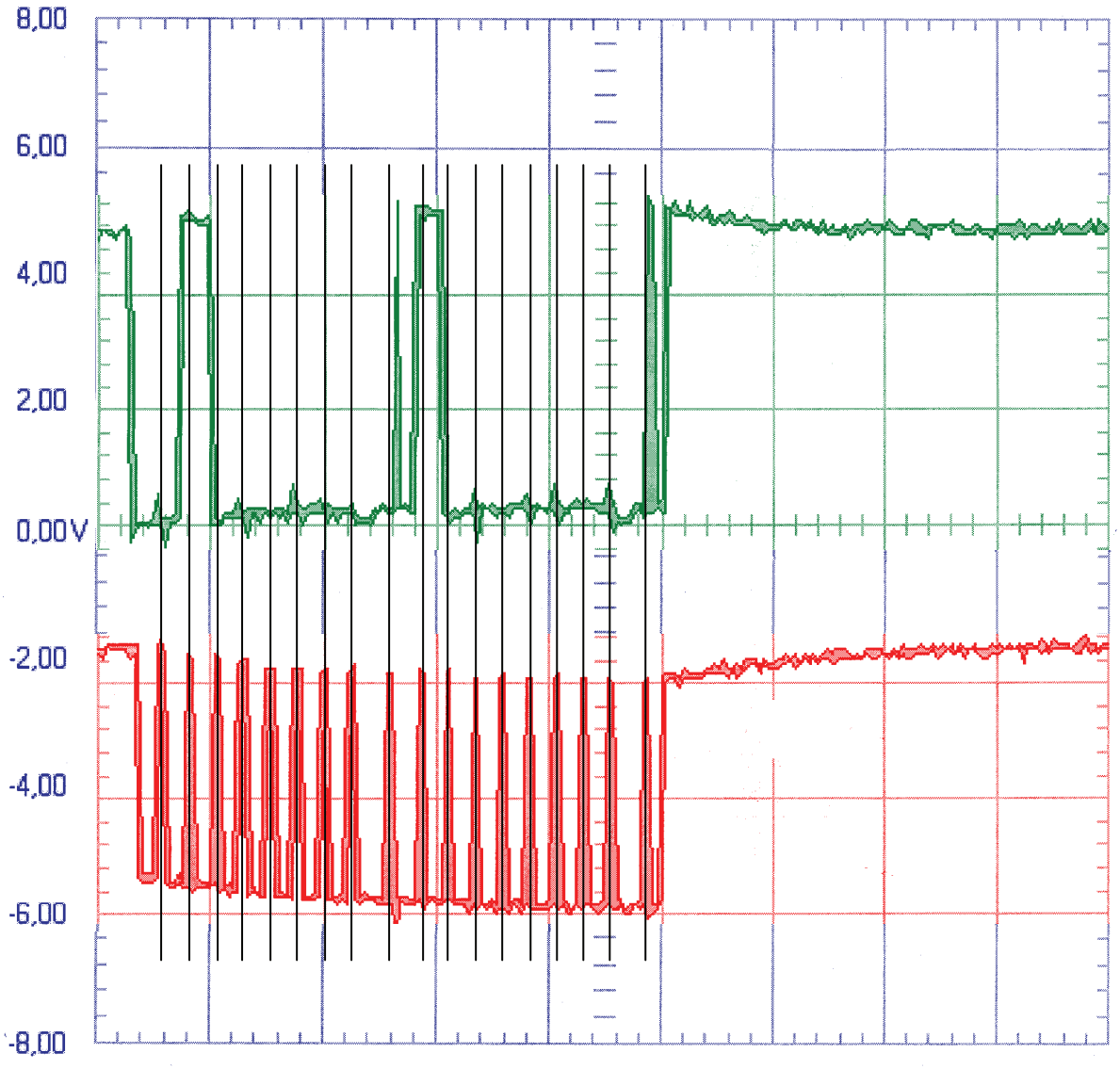
Chronogramme 12 : Trame de lecture d'un octet $F7 à l'adresse $49



Sur ce chronogramme nous pourrons voir le bit de **start** ; **l'adresse $49** en hexadécimal ; le bit **lecture** ; **l'acknowledge (accusé de réception)** pour confirmer la liaison; la **donnée $F7** en hexadécimal ; **l'acknowledge** pour confirmer la réception de la donnée;et le bit de **stop** pour arrêter l'échange de données.

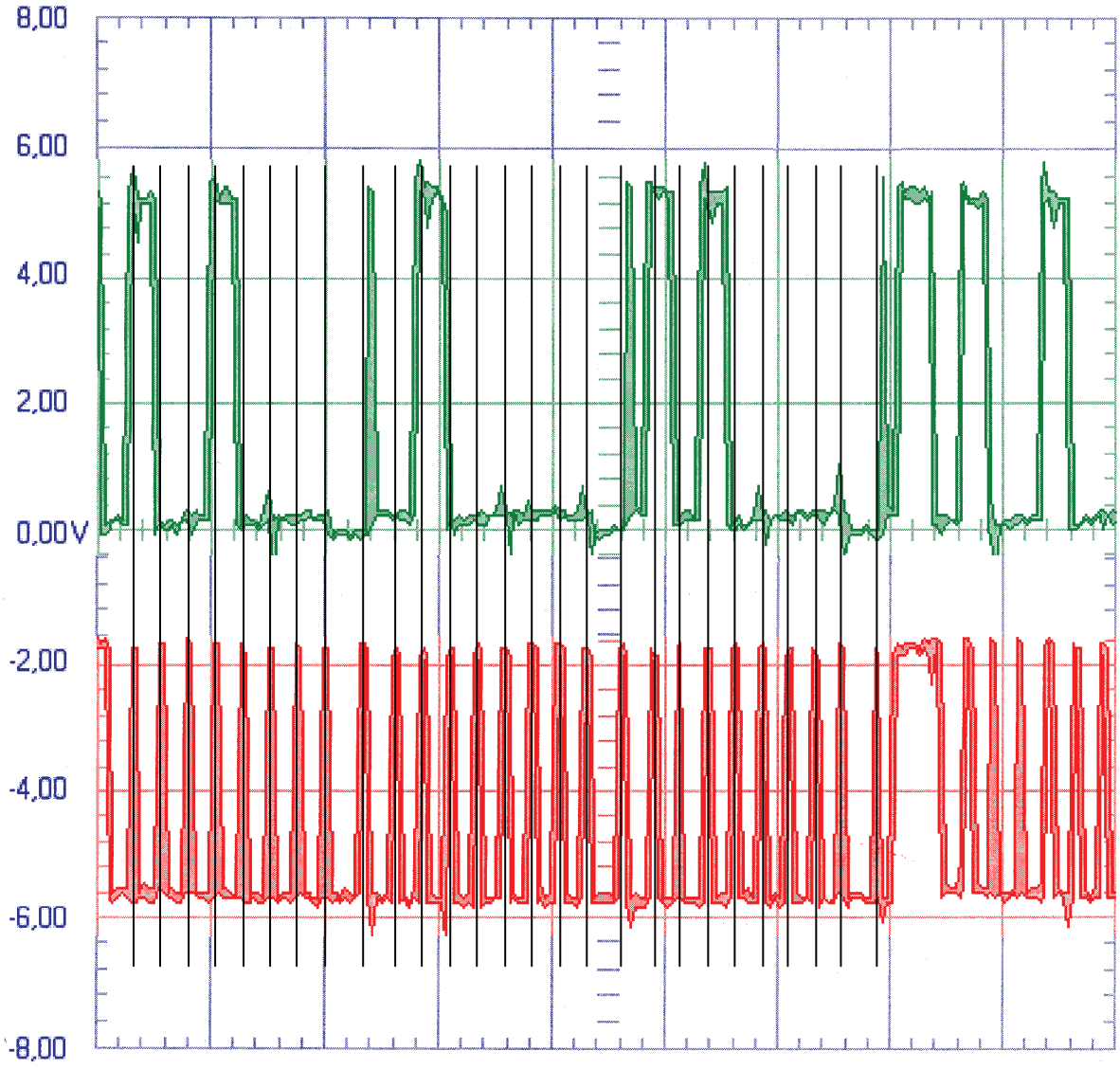
6.8 Relevés - Message simple d'un PCF8574

Message simple d'un contrôleur de port // 8bits PCF8574 - Relevé d'un oscillogramme pour l'envoi d'une trame pour le chenillard



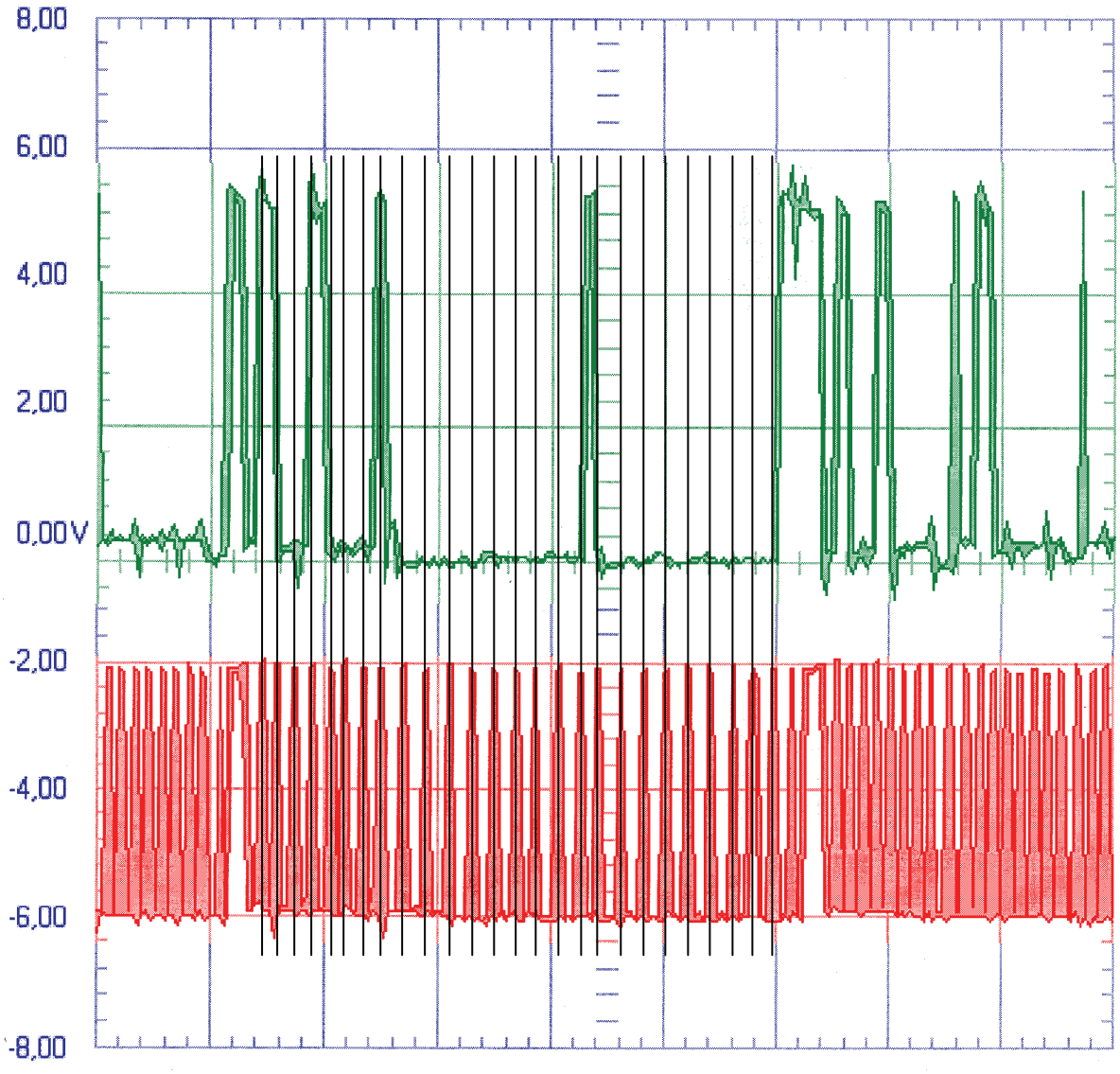
Caractéristique	VOIE 1	VOIE 2
Nom du signal	SDA	SCL
Echelle Tension	2V/DIV	2V/DIV
Echelle Temps	100µS/DIV	100µS/DIV

Message simple d'un CAN/CNA PCF8591 - Relevé d'un oscillogramme pour l'opération de conversion Numérique Analogique (génération d'un triangle)



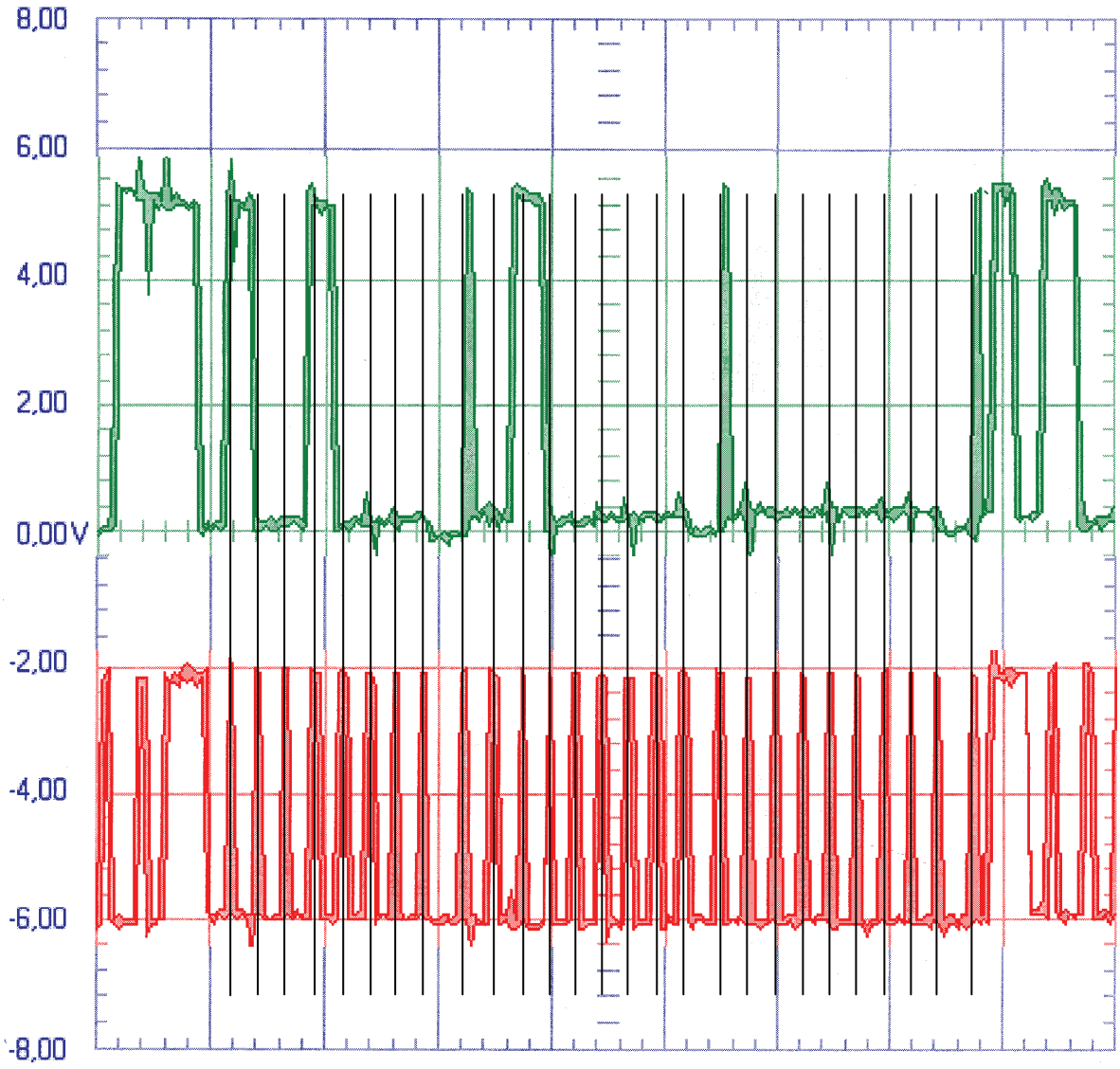
Caractéristique	VOIE 1	VOIE 2
Nom du signal	SDA	SCL
Echelle Tension	2V/DIV	2V/DIV
Echelle Temps	100µS/DIV	100µS/DIV

Message double d'un CAN/CNA PCF8591 - Relevé d'un oscillogramme pour l'opération de conversion Analogique Numérique.



Caractéristique	VOIE 1	VOIE 2
Nom du signal	SDA	SCL
Echelle Tension	2V/DIV	2V/DIV
Echelle Temps	200µS/DIV	200µS/DIV

Message double d'un CAN/CNA PCF8591 - Relevé d'un oscillogramme pour l'opération de conversion Numérique Analogique.



Caractéristique	VOIE 1	VOIE 2
Nom du signal	SDA	SCL
Echelle Tension	2V/DIV	2V/DIV
Echelle Temps	100µS/DIV	100µS/DIV

PROGRAMMATION DU FONCTIONNEMENT NORMAL DU STORE
(séquence 7)

Lors de cette séquence nous avons conçu le programme final. Dans ce programme se trouve le programme de la séquence 5. Et une autre partie de programme permettant de faire fonctionner le store soit en automatique, soit en manuel.

7.1 Programme de gestion du store

Ci-dessous contenu du programme associé à ce programme avec explications :
Programme **Programme_Complet.c**

```
////////////////////////////////////////////////////////////////
// Programme : Programme_complet.c
////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////
//
// Objet du programme : Le programme Programme_complet a pour but
// de programmer la partie du fonctionnement normal en plus
// de la gestion des programmes de test.
//
// En fonction normal, ce programme gére le mode automatique
// et le mode manuel
//
////////////////////////////////////////////////////////////////

// Auteurs : --ANONYMAT--

#include std84.h
#include bit84.h
#define MS porta.2          // Commande Moteur : à 1 pour la montée du store
#define DS porta.3          // Commande Moteur : à 1 pour la descente du store
#define DMDS portb.4        // Demande Manuelle de Descente du Store (BP VERT)
#define DMMS portb.5        // demande Manuelle de Montée du Store (BP ROUGE)
#define VDS portb.2         // Visualisation la descente du store (LED VERTE) (actif à 0)
#define VMS portb.3         // Visualisation la montée du store (LED ROUGE) (actif à 0)
#define Visu_CMS Drapeaux.7     // LED de visualisation : Commande du moteur pour Montée du
Store
#define Visu_CDS Drapeaux.6     // LED de visualisation : Commande du moteur pour Descente du
Store
#define Visu_hau Drapeaux.5     // LED de visualisation : Store en position haute
#define Visu_bas Drapeaux.4     // LED de visualisation : Store en position basse
#define VisuDMMS Drapeaux.3     // LED de visualisation : Demande Manuelle de Montée du Store
#define VisuDMDS Drapeaux.2     // LED de visualisation : Demande Manuelle de Descente du Sto-
re
#define VisuDSLu Drapeaux.1     // LED de visualisation : Dépassement Seuil LUmière
#define VisuDSVe Drapeaux.0     // LED de visualisation : Dépassement Seuil Vent
#define FctManuel VariaLogi.3   // variable interne du microcontroleur : Fonctionnement Ma-
nuel/Auto
#define IniPosition VariaLogi.2 // variable interne du microcontroleur : Initialisation posi-
tion store
#define DMS VariaLogi.1    // variable interne du microcontroleur : DMS
#define DDS VariaLogi.0    // variable interne du microcontroleur : DDS
#define MasquePortB 0x30   // Masque toutes les lignes du portb sauf RB4 et RB5
#define cnavercan

char Mesurlum, Mesurven, Seuillum, Drapeaux, Temporaire, VariaLogi;

char indice,exindice;
char a ,b, b_ascii, c;
char resultatcan;

////////////////////////////////////////////////////////////////
// PROGRAMME PRINCIPAL
```

```
//////////////////////////////////////////////////////////////

void main()
{
 trisb=0xF3;          // Configuration des E/S du port B (RB2 et RB3 en sortie ;
RB0,RB1,RB4,RB5,RB6,RB7 en entrée <=11110011)
 trisa=0xF3;          // Configuration des E/S du port A (RA2 et RA3 en sortie ;
RA0,RA1,RA4,RA5,RA6,RA7 en entrée <=11110011)
 OPTION.7=0;          // Résistances de tirage interne du port B du PIC
 VMS=1;               // Eteindre la LED de Visualisation de la Montée du Store
 VDS=1;               // Eteindre la LED de Visualisation de la Descente du Store
 resultatcan=0;
 DS=0;                // Commande du moteur à l'arret
 MS=0;                // Commande du moteur à l'arret
 delays(2);           // TEMPORISATION DE 2 SECONDES
 i2c_lcd(0x70,2,4);   // Affiche choix entre fonctionnement test et fonctionnement normal
 delay(100);          // TEMPORISATION DE 100 ms
 while(1)             // BOUCLE INFINIE
  {if (!DMDS)         // Si l'on appuie sur le BP de DMDS alors
   {i2c_lcd(0x70,2,19); // fonctionnement normal, affiche choix entre automatique et manuel
    delay(200);
    while(1)          // BOUCLE INFINIE
    {if (!DMDS)       // Si l'on appuie sur le BP de DMDS alors
     {FctManuel=1;
     fctnorma();// lance le sous programme fctnorma
     }
    }
    else        //sinon
     {
    if (!DMMS)        // Si l'on appuie sur le BP de DMMS alors
        {FctManuel=0;
         fctnorma();  // lance le sous programme fctnorma
     }
    }
   }
  }
  else if (!DMMS)     // SI l'on appuie sur le BP de DMMS alors
   {i2c_lcd(0x70,2,17);    // affiche store 2006 tests
    delay(100);       // temporisation 100ms
    i2c_lcd(0x70,2,5);     // affiche choix entre analogique et logique
    delay(100);       // temporisation 100ms
    while (1)         // BOUCLE INFINIE
     {if (!DMDS)// si l'on appuie sur le BP de DMDS alors
      {i2c_lcd(0x70,2,6); // affiche choix entre ihm local ou sur bus I2C
       delay(200);    //temporisation 200ms
       while (1)// BOUCLE INFINIE
        {if (!DMMS)   //SI l'on appuie sur le BP de DMMS alors
         {i2c_lcd(0x70,2,12); // affiche test touches DEL
          ihmboudi();   // lance le sous programme ihmboudi
         }
        else if (!DMDS)    // Sinon sI l'on appuie sur le BP de DMDS alors
         {i2c_lcd(0x70,2,7); // affiche choix entre caractères ou voyants
          delay(200); //temporisation 200ms
          while (1)   // BOUCLE INFINIE
          {if (!DMDS) //sI l'on appuie sur le BP de DMMS alors
           {i2c_lcd(0x70,2,13); // affiche test voyants i2c
            comcheni();    // lance le sous programme comcheni
           }
           else if (!DMMS) //Sinon sI l'on appuie sur le BP de DMMS alors
            {i2c_lcd(0x70,2,14); // affiche test caracte i2c
             defillcd(); // lance le sous programme defillcd
            }
         }
      }
     }
   }
   else if (!DMMS)    //Sinon sI l'on appuie sur le BP de DMMS alors
    {i2c_lcd(0x70,2,8);    // affiche choix entre génération et recopie d'une tension
     delay(200);//temporisation 200ms
     while (1)        // BOUCLE INFINIE
      {if (!DMMS)     //Sinon sI l'on appuie sur le BP de DMMS alors
```

```
       {i2c_lcd(0x70,2,15); // affiche test gene triang
        cnatrian();  // lance le sous programme cnatrian
       }
      else if (!DMDS)  //si l'on appuie sur le BP de DMMS alors
       {i2c_lcd(0x70,2,16); // affiche store 2006 tests
        cnavercan();  // lance le sous programme cnavercan
       }
      }
    }
   }
  }
 }
}

////////////////////////////////////////////////////////////////
// <FIN> SOUS PROGRAMME PRINCIPAL
////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////
// <DEBUT> SOUS PROGRAMME FCTNORMA
////////////////////////////////////////////////////////////////

void fctnorma()
{
 VMS = 1;        // DEL verte éteinte
 VDS = 1;        // DEL rouge éteinte
 Seuillum = 0x70;     // Réglage du seuil lumière
 Drapeaux = 0;
 Visu_hau = 1;        // store en position haute - led jaune (D5) allumée
 IniPosition = 0;
 if (FctManuel)
  {
   i2c_lcd (0x70,2,21); // affiche store 2066 manu. lum xxx vent xxx
   delay (200); // temporisation de 200 ms
  }
  else
   {
  i2c_lcd (0x70,2,20);     // affiche store 2006 auto. lum xxx vent xxx
  delay (200);        // temporisation de 200 ms
   }
  while (1)           // Boucle infinie
   {
    DMS = 0;          // demande montée store
    DDS = 0;          // demande descente store
    Affilumi(); // LANCE SOUS PROGRAMME AFFILUMI
    Affivent();       // LANCE SOUS PROGRAMME AFFIVENT
    if (!IniPosition)
     {
     IniPosition = 1;
     DMS = 1;         // demande montée store
     }
    if (FctManuel)
     {
     if (!DMDS)
      {
       if (Visu_hau)  // si store position haute
        {
        DDS = 1;;     // demande descente store
        VisuDMDS = 1;      // demande manuelle descente store
        }
       else
       {
       VisuDMDS = 0;
        if (!DMMS)
         {
          if (Visu_bas) // si store position basse
          DMS = 1;;   // demande montée store
          VisuDMMS = 1; // demande manuelle montée store
         }
        else
         {
```

```
          VisuDMMS = 0;    // demande manuelle montée store
         }
       }
     }
     else
     {
      if (Mesurlum > Seuillum) // si intensité lumineuse supérieure au seuil
       {
        VisuDSLu = 1;      // seuil de lumière dépassé - led jaune (D1) allumée
        if (Visu_hau)      // si store en position haute - led jaune (D5) allumée
        DDS = 1;     // demande descente store
       }
      else
       {
        VisuDSLu = 0;      // seuil de lumière non dépassé - led jaune (D1) éteinte
        if (Visu_bas)      // si store en position basse - led rouge (D4) allumée
         DMS = 1;     // demande montée store
       }
     }

     Traitmot ();     // LANCE SOUS PROGRAMME TRAITMOT
  }
}
////////////////////////////////////////////////////////////////////
// <FIN> SOUS - PROGRAMME FCTNORMA
////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////
// <DEBUT> SOUS PROGRAMME AFFILUMI
////////////////////////////////////////////////////////////////////

 void Affilumi()
 {
     i2c_out_1oct(0x48,0x40);  //Sélection du canal0 du can.
     Mesurlum = i2c_can(0x48); //acquisition de la l'intensité lumineuse
     delay(200);
     i2c_lcd(0x70,2,9);
     delay(200);
     b_ascii=Mesurlum;
     b_ascii=b_ascii>>4;
     Envoiasc();
     b_ascii=Mesurlum;
     Envoiasc();
 }

////////////////////////////////////////////////////////////////////
// <FIN> SOUS PROGRAMME AFFILUMI
////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////
// <DEBUT> SOUS PROGRAMME AFFIVENT
////////////////////////////////////////////////////////////////////

void Affivent()
 {
     i2c_out_1oct(0x48,0x43);  //Sélection du canal3 du can.
     Mesurven = i2c_can(0x48);
     delay(200);
     i2c_lcd(0x70,2,11);
     delay(200);
     i2c_lcd(0x70,2,11);
     delay(200);
     b_ascii=Mesurven;
     b_ascii=b_ascii>>4;
     Envoiasc();
     b_ascii=Mesurven;
     Envoiasc();
 }
////////////////////////////////////////////////////////////////////
// <FIN> SOUS PROGRAMME AFFIVENT
////////////////////////////////////////////////////////////////////
```

```
////////////////////////////////////////////////////////////////
// <DEBUT> SOUS PROGRAMME ENVOIASC
////////////////////////////////////////////////////////////////


void Envoiasc()
 {
     b_ascii=b_ascii&0x0f;
     if (b_ascii<0x0a)
          b_ascii=b_ascii+0x30;
     else
          b_ascii=b_ascii+0x37;
     i2c_lcd(0x70,1,b_ascii);
     delay(250);
 }

////////////////////////////////////////////////////////////////
// <FIN> SOUS PROGRAMME ENVOIASC
////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////
// <DEBUT> SOUS PROGRAMME TRAITMOT
////////////////////////////////////////////////////////////////

void Traitmot()
 {   RafraichiVoyants();
     if (DMS)
       {
        VMS = 0;
      MS = 1;//porta = 0x08;
        Visu_CMS = 1;
      DDS=0;
     }
     else
     {
      if (DDS)
          {
           VDS = 0;
     DS =1 ;//porta = 0x04;
     Visu_CDS = 1;
          }
     }
     RafraichiVoyants();
     if (DMS)
       {
         delays(3);
         Visu_hau = 1;
       Visu_bas = 0;
       MS = 0;
       VMS = 1;
       Visu_CMS = 0;
       RafraichiVoyants();
     }
   else
       {
         if (DDS)
           {
             delays(3);
       Visu_hau = 0;
       Visu_bas = 1;
       DS = 0;
           VDS = 1;
       Visu_CDS = 0;
       RafraichiVoyants();
           }
     }
 }

////////////////////////////////////////////////////////////////
// <FIN> SOUS PROGRAMME TRAITMOT
```

```
////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////
// <DEBUT> SOUS PROGRAMME RAFRAICHIVOYANTS
////////////////////////////////////////////////////////////////

void RafraichiVoyants()
 {
   Temporaire=~Drapeaux;
     i2c_out_1oct(0x20,Temporaire);
 }

////////////////////////////////////////////////////////////////
// <FIN> SOUS PROGRAMME RAFRAICHIVOYANTS
////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////
// <DEBUT> SOUS PROGRAMME IHMBOUDI
////////////////////////////////////////////////////////////////

void ihmboudi()
{


      while(1) // boucle infinie
        {

           if((portb & MasquePortB) == 0x10)  //  cas on l'on appuye uniquement sur bouton
poussoir 1(SW1)
              {
                VDS=0; // DEL rouge allumée
                delay(200);// temporisation
          VDS=1;// DEL rouge éteinte

                delay(200);// temporisation

              }
            else
              {
                VDS=1; // DEL rouge éteinte
              }



           if((portb & MasquePortB) == 0x20) //  cas on l'on appuye uniquement sur bouton
poussoir 2(SW2)
             {
               VMS=0; // DEL verte allumée

               delay(200);// temporisation

          VMS=1;// DEL verte éteinte

                delay(200);// temporisation

             }
            else
              {
              VMS=1;// DEL verte éteinte

              }


           if((portb & MasquePortB) == 0x00)// cas on l'on appuye sur les boutons poussoirs
              {
              VDS=0; // DEL rouge allumée

              delay(200);// temporisation

              VDS=1;// DEL rouge éteinte
              delay(100);// temporisation
```

```
            VMS=0;// DEL verte allumée

              delay(200);// temporisation

              VMS=1;// DEL verte éteinte

              delay(100);// temporisation


              }

       }
}

////////////////////////////////////////////////////////////////
// <FIN> SOUS PROGRAMME IHMBOUDI
////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////
// <DEBUT> SOUS PROGRAMME COMCHENI
////////////////////////////////////////////////////////////////

void comcheni()
      {


      indice = 0xFE; // affectation de la valeur $FE (11111110) dans la variable indice.

      while(1)   // boucle infinie
      {

      i2c_out_1oct(0x20,indice); // Envoie de la donnée $FE à l'adresse $20 (PCF8574) via le
bus I2C
      delay(200); // attente de 200 ms
         if(indice == 0) // Si toutes les leds sont allumées
             indice=0xFE; // On éteint toutes les leds sauf la première
            else
            indice=indice<<1; // On effectue un décalage de 1 bit à gauche
          }
    }

////////////////////////////////////////////////////////////////
// <FIN> SOUS PROGRAMME COMCHENI
////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////
// <DEBUT> SOUS PROGRAMME defillcd
////////////////////////////////////////////////////////////////

void defillcd()
{
 delays(1); // attente de 1 seconde
 i2c_lcd(0x70,2,1); // efface l'afficheur (adresse=$70 2=SelectrionModeCommande 1=N°Commande)
 delays(1); // attente de 1 seconde
 i2c_lcd(0x70,2,1); // efface l'afficheur (adresse=$70 2=SelectrionModeCommande 1=N°Commande)
 delays(1); // attente de 1 seconde
 while(1)
   {
    for( b = 0 ; b <= 0xF ; b++ ) // execute 16 fois les instructions suivantes en incrémentant
b de 1 en 1.
      {
        b_ascii=b;          // On affecte à la variable B_ascii le contenu de la variable B
        if (b_ascii>9)            // SI b_ascii est > 9 alors
          b_ascii=b_ascii+0x37;   // b_ascii = b_ascii+$37
        else                // sinon
         b_ascii=b_ascii+0x30;    // b_ascii = b_ascii+$30

        i2c_lcd(0x70,1,b_ascii); // Afficher un caractère (adresse=$70;
1=ModeAffichageCaractère ; Caractère à afficher=b_ascii)
```

```
          delay(250);          // attendre 250ms
      }
   i2c_lcd(0x70,2,1);           // efface l'afficheur (adresse=$70 2=SelectrionModeCommande
1=N°Commande)
   delays(1);                   // attendre 1 seconde
  }
}

////////////////////////////////////////////////////////////////
// <FIN> SOUS PROGRAMME defillcd
////////////////////////////////////////////////////////////////


////////////////////////////////////////////////////////////////
// <DEBUT> SOUS PROGRAMME CNATRIAN
////////////////////////////////////////////////////////////////

void cnatrian()

{
     while(1) // Boucle infinie
     {
     for(indice=0;indice<250;indice=indice+10) // executer 25 fois l'instruction suivante en
incrémentant indice de 10 en 10.

        i2c_out_2oct(0x48,0x40,indice); // Lancement d'une CNA à l'adresse $48, avec un octet
de contrôle=$40, et un nombre à convertir = indice

        for(indice=250;indice>0;indice=indice-10) // executer 25 fois l'instruction suivante
en décrémentant indice de 10 en 10.

        i2c_out_2oct(0x48,0x40,indice); // Lancement d'une CNA à l'adresse $48, avec un octet
de contrôle=$40, et un nombre à convertir = indice
        }
}

////////////////////////////////////////////////////////////////
// <FIN> SOUS PROGRAMME CNATRIAN
////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////
// <DEBUT> SOUS PROGRAMME CNAVERCAN
////////////////////////////////////////////////////////////////

void cnavercan()

   {
while(1) // Boucle infinie
   {
 resultatcan = i2c_can(0x48); // resultatcan récupère la valeur numérique issue de la CAN

 i2c_out_2oct(0x48,0x40,resultatcan); // Lancement d'une CNA à l'adresse $48, avec un octet de
contrôle=$40, et un nombre à convertir = resultacan

   }
   }

////////////////////////////////////////////////////////////////
// <FIN> SOUS PROGRAMME CNAVERCAN
////////////////////////////////////////////////////////////////

// FIN DU PROGRAMME
```