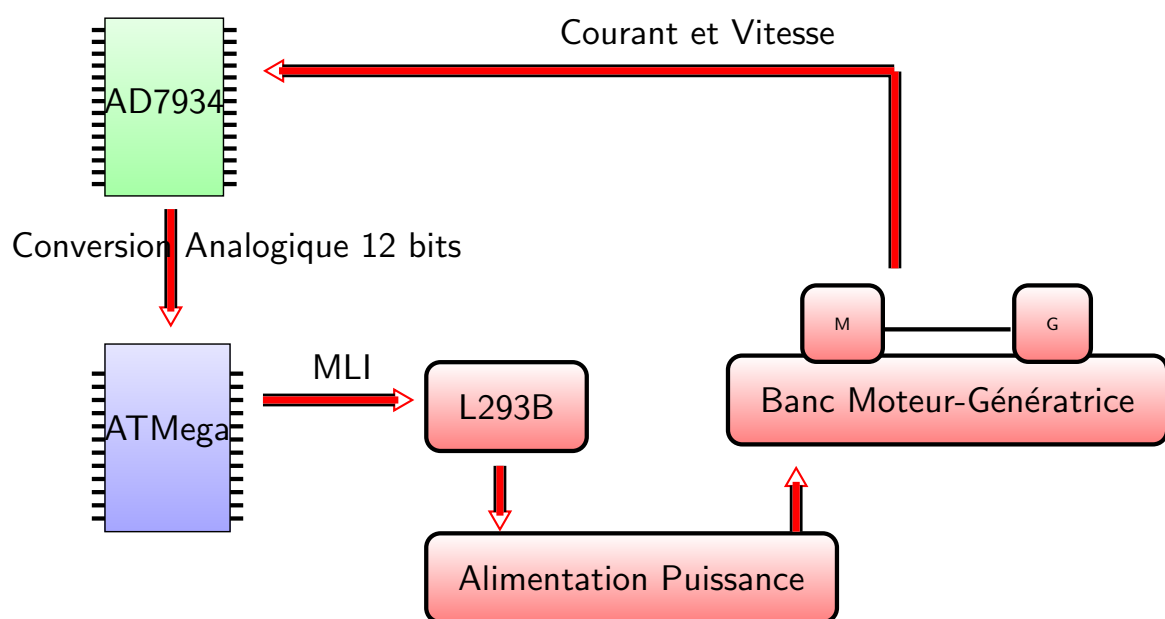


Micro-Projet d'Informatique Industrielle S7

Nadia Aït Ahmed, David Delfieu

September 14, 2026

Commande en boucle ouverte d'une *MCC*



- 1 Génération de la MLI
 - Hacheur et *MCC*
 - Programmation du Timer de l'*ATMEGA*₂₅₆₀
- 2 Etude et écriture du driver de l'ADC AD7934
 - Le brochage
 - Entrées du CAN
 - L'interface parallèle
 - Driver de l'*AD7934*
- 3 Tracé des courbes courant et vitesse

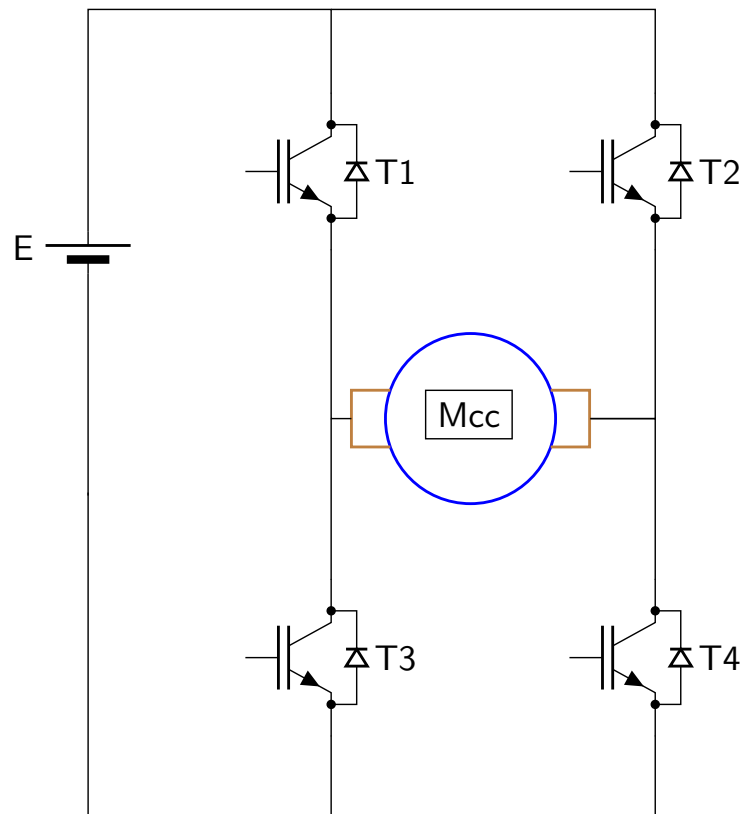
Objectif

Dans cette section on va compléter le sous-programme :

```
#define ThN ?
#define Vmoy -1789

void initimer(){
    DDRB = ?;
    TCCR1A = ?;
    TCCR1B = ?;
    OCR1A = ?;
    OCR1B = ?;
}
```

Hacheur L293 H-Bridge



$+E$ aux bornes de la *MCC*

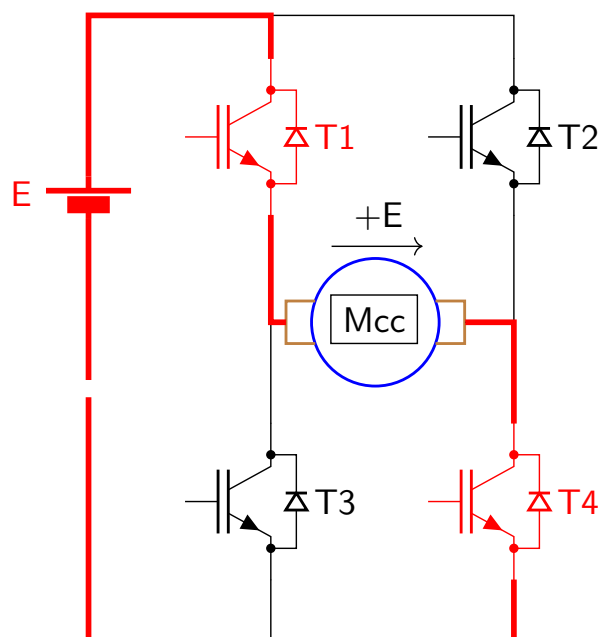


Figure: Hacheur

- T_1 et T_4 passant on a $V = +E$ aux bornes de la *MCC*

-E aux bornes de la *MCC*

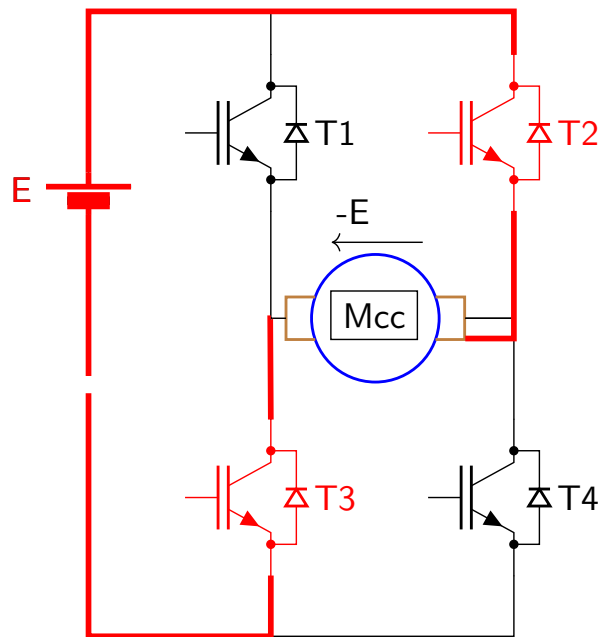
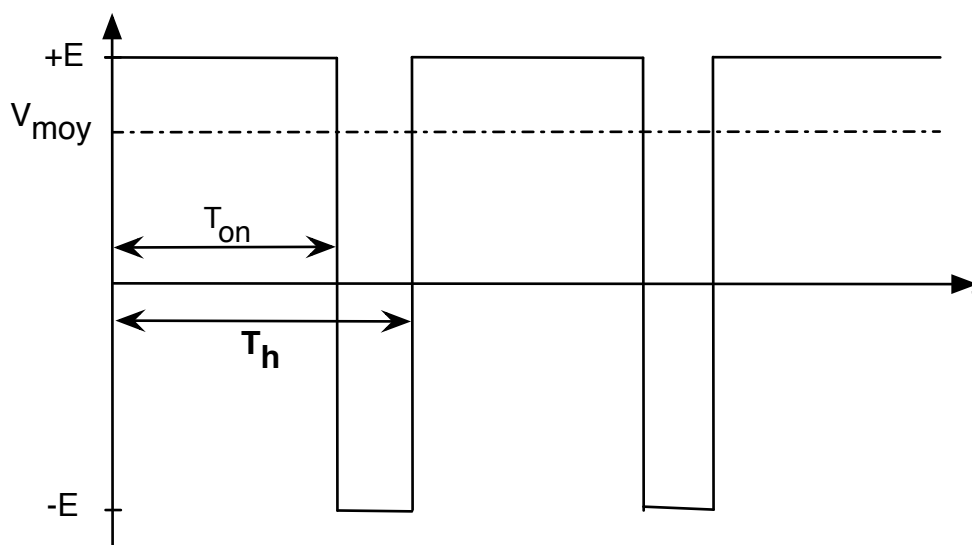


Figure: Hacheur

- T_2 et T_3 passant on a $V = -E$ aux bornes de la *MCC*

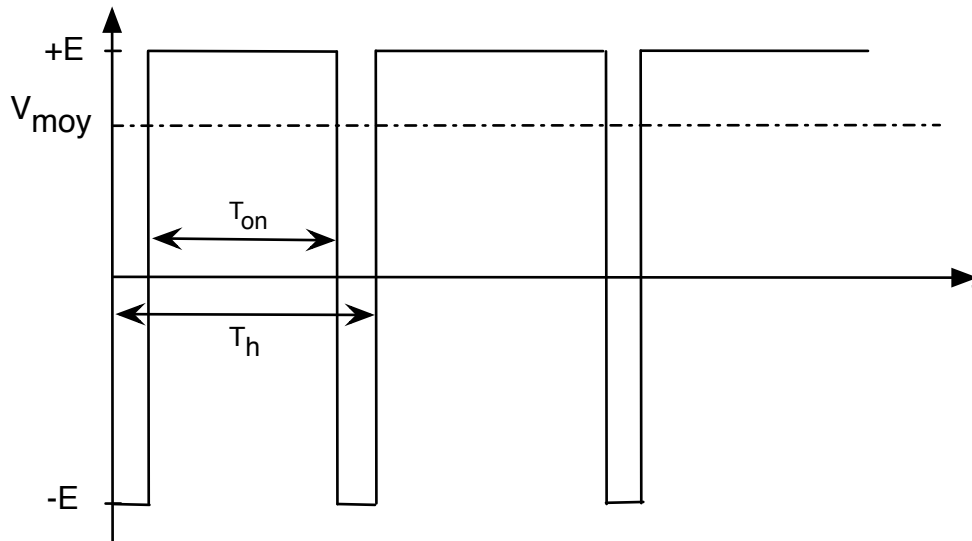
Consigne de tension hachée : T_{on} non centré



T_{on} non centré.

$$V_{moy} = \frac{1}{T_h} \left(\int_0^{T_{on}} E dt + \int_{T_{on}}^{T_h} -E dt \right)$$

T_{on} centré sur T_h



T_{on} centré.

$$V_{moy} = \frac{1}{T_h} \left(\int_0^{\frac{T_h - T_{on}}{2}} -E dt + \int_{\frac{T_h - T_{on}}{2}}^{\frac{T_h + T_{on}}{2}} E dt + \int_{\frac{T_h + T_{on}}{2}}^{T_h} -E dt \right)$$

Navigation icons: back, forward, search, etc.

Numérisation du Calcul

- V_{moy} est exprimé en volt,
- T_{on}, T_h , sont exprimés en secondes (microsecondes).
- micro-contrôleur : registres sont des entités sans unités
- Numérisation : Passer de variables avec unités à des registres
- Notation : Numérisation une variable $V \Rightarrow V^N$

$$\frac{T_{on}}{T_h} = \frac{T_{on}^N}{T_h^N}$$

Navigation icons: back, forward, search, etc.

Calcul d'expressions littérales

V_{moy} étant défini au préalable à l'aide une étiquette #define:

- Calculer les deux intégrales précédentes.
- En déduire l'expression littérale de V_{moy}
- En déduire l'expression littérale de T_{on}
- Calculer l'expression littérale du rapport $\frac{T_{on}}{T_h}$

Etude de la documentation du Timer (1/3)

A partir de la documentation [ATmega2560_datasheet](#) :

- Dans la figure 17.1 page 137, retrouver les registres du timer 1 permettant de régler TOP (c.a.d. ThN) et T_{on}^N
- Trouver dans [ATmega2560_datasheet](#) la table des modes du timer 1. A quel mode du timer 1 correspond à une MLI centrée ?
- Pour ce mode, trouver la formule qui définit la fréquence de hachage en fonction de la valeur TOP .
- Compléter dans la fonction `initTimer()`, le registre ad. hoc pour avoir une fréquence de hachage de 15khz.
- Fig. 17.1, p 134, retrouver les 4 registres pour MLI sur la patte **OC1B**.

Etude de la documentation du Timer (2/3)

- Quelles valeurs doit-on mettre dans WGM_{13} WGM_{12} WGM_{11} WGM_{10} ?
- Trouver la figure qui montre dans un chronogramme les rôles des registres OCR_{1A} et OCR_{1B} en mode phase et fréquence correcte.
- Trouver la table qui définit comment déterminer les bits COM_{1B1} et COM_{1B0} et donner la référence.
- Traduire et reformuler la 3^{ème} ligne de ce tableau.

Etude de la documentation du Timer (3/3)

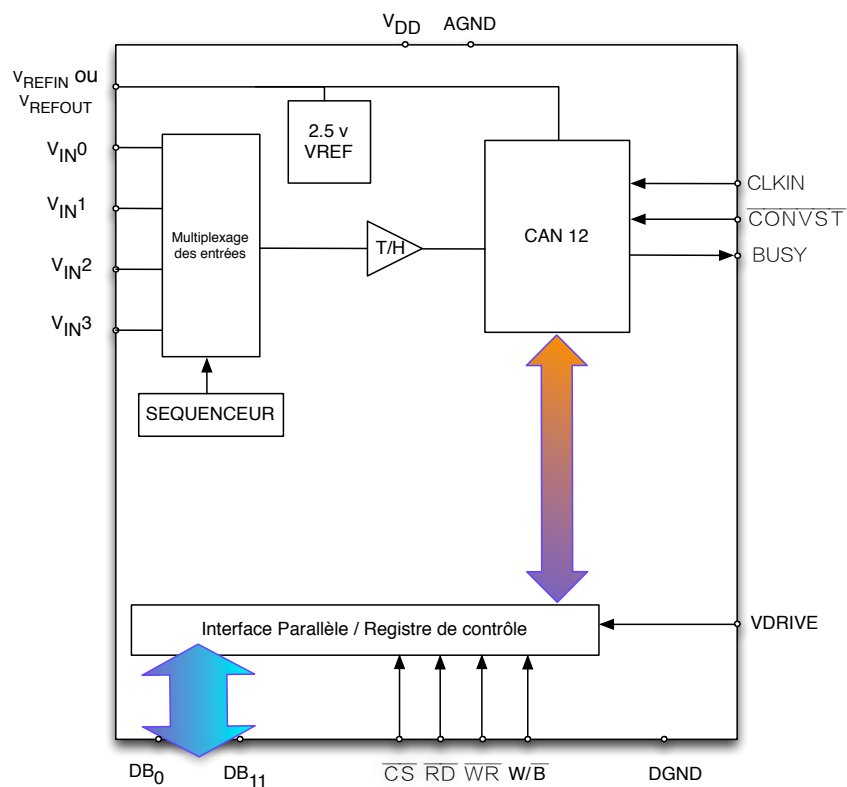
- Donner les lignes d'initialisation des registres $TCCR_{1A}$ et $TCCR_{1B}$
Exemple d'initialisation du Mode 11 :
 $TCCR1A = (1 \ll \text{bit3}) + (1 \ll \text{bit5});$
 $TCCR1B = (1 \ll \text{bit1}) + (1 \ll \text{bit0});$
- Retrouver les registres tels que $\frac{T_{on}}{T_h} = \frac{reg_1}{reg_2}$
- A partir des formules littérales définissant $\frac{T_{on}}{T_h}$ établir le calcul la valeur d'initialisation de reg_1 .
- Tester des valeurs positives, négatives et nulles de V_{moy}

Visualisation de la MLI à l'oscilloscope

- V_{moy} étant défini à l'aide une étiquette #define:
- Brancher la sortie à l'oscilloscope
- Visualiser la MLI
- Faire varier V_{moy} en positif, négatif ou nul.

Convertisseur AD7934

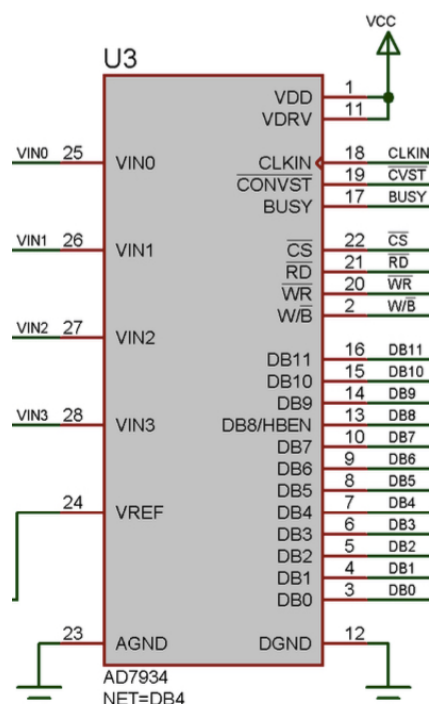
Nous allons étudier et écrire un driver pour l'AD7934 :



Convertisseur AD7934

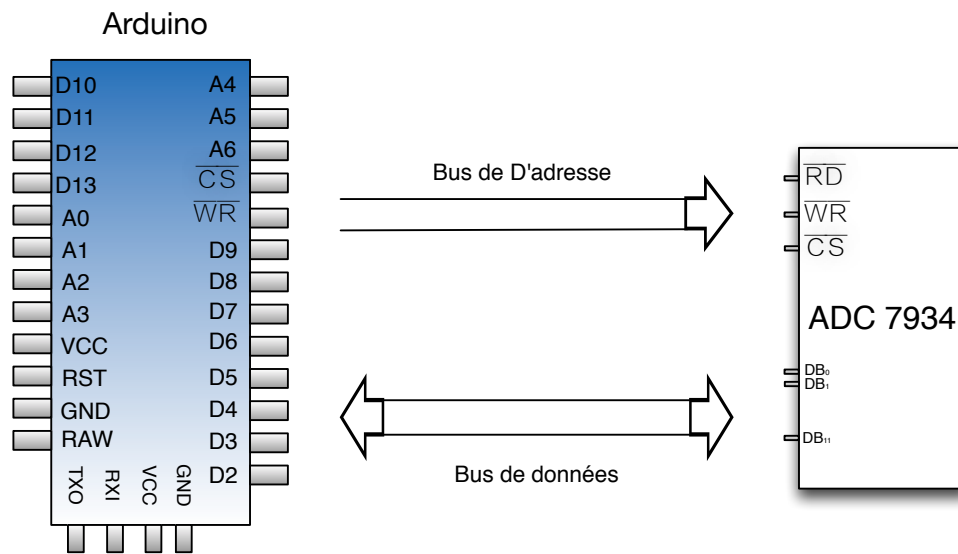
- 4 entrées analogiques $Vin_0, Vin_1, Vin_2, Vin_3$ en mode simple ou en mode différentiel
- L'**AD7934** est connecté à l'*Arduino* par :
 - ▶ un bus d'adresse ($\overline{CS}, \overline{RD}, \overline{WR}, W/B$)
 - ▶ un bus de données ($DB_0, \dots, DB_{11}$) pour une lecture rapide de la conversion.
- Il utilise une référence de courant interne.
- Echantillonnage déclenché par le signal $\overline{CONVST}$

Etude du cablage



Bus d'adresses - Bus de données

Tout composant relié à un processeur peut être relié de la façon suivante :

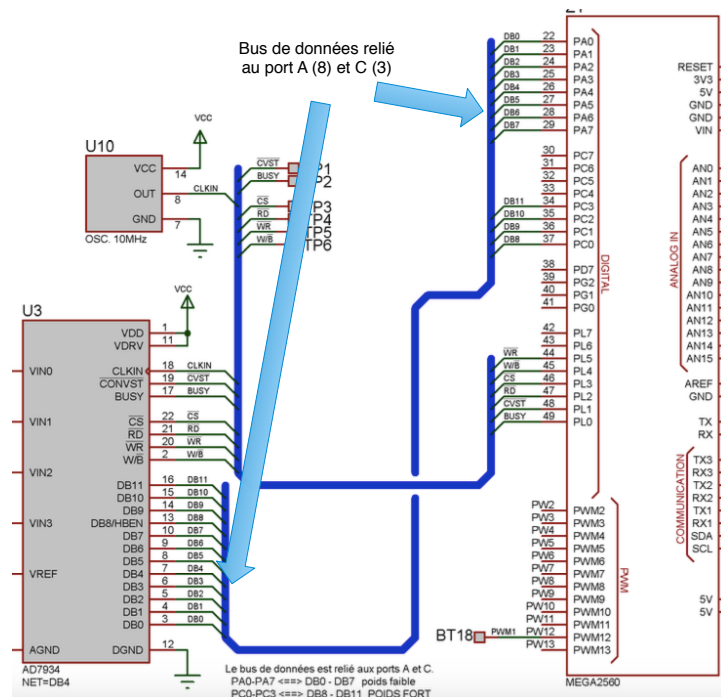


Bus de données

Bus de données (Data Bus) : $DB_0, \dots, DB_{11}$

- Bus bidirectionnel.
- Initialisation : Bus en écriture sur : *ATMega* $\Rightarrow$ *AD7934*:
Informations pour programmer les modes
- Conversion : Bus en lecture sur : *AD7934* $\Rightarrow$ *ATMega*:
Resultat de conversion
- Il doit être connecté à des ports de l'*ATMega*

Bus de données

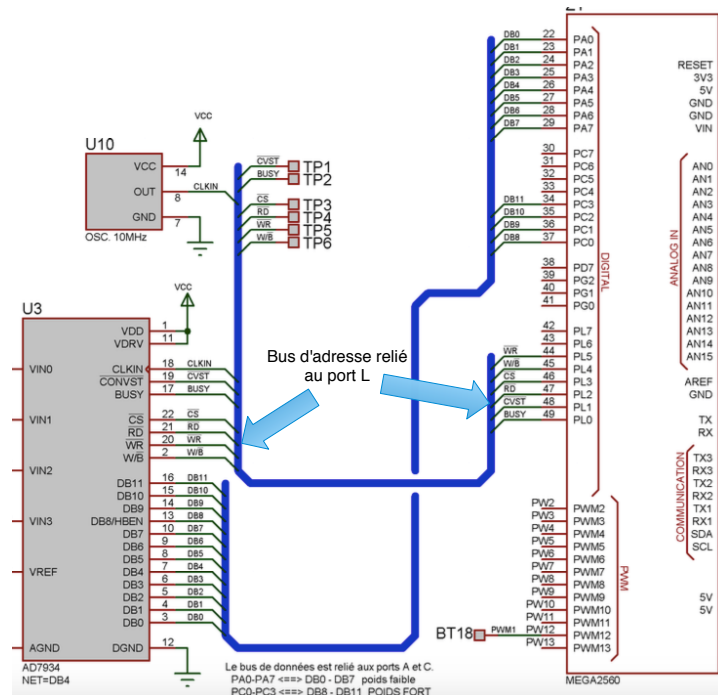


Bus d'adresse et de contrôle

Bus d'adresse et de contrôle :

- $\overline{CS}$: Chip Select
- $\overline{WR}$: Write
- $\overline{RD}$: Read
- $W/\overline{B}$: Write Byte
- $BUSY$: Conversion en cours
- $\overline{CONVST}$: Conv Start
- $CLKIN$: Horloge

Bus d'adresses (et de contrôle)



Analyse des liaisons entre l'AD7934 et l'ATmega

A partir de la documentation [CarteTP_ATMega](#) :

- Identifier chaque liaison connectant une pin de l'ATmega à une pin de l'AD7934.
- Pour chacune des pins identifiées de l'AD7934, à partir de la documentation [AD7934Bruz](#) déterminer :
 - Le sens de circulation de l'information
 - Leurs rôles : Une simple phrase accompagnée de la référence (numéro de page).
- En déduire l'initialisation de DDR_E
- En déduire l'initialisation de DDR_A , DDR_C et suivant que l'on veut écrire ou lire l'AD7934

Entrées analogiques de l'AD7934

A partir de la documentation [CarteTP_ATMega](#) :

- Repérer sur [CarteTP_ATMega](#) les entrées analogiques correspondant aux *BNC*
- A partir de la documentation [AD7934Bruz](#) expliquer comment ces entrées analogiques peuvent être utilisées suivant 3 modes différents.
- En examinant ce montage, quel est le mode utilisé ?

Entrées différentielles

A partir de la documentation [CarteTP_ATMega](#) et de [AD7934Bruz](#) :

- Les *bnc* J_1 et J_2 acceptent des tensions sur l'intervalle $[-10, +10\text{V}]$
- Le composant $U8$ (Traco) convertit les tensions dans la plage $[-12\text{V}, +12\text{V}]$ qui est nécessaire pour les *AOP*
- SUB_1 et SUB_2 ramènent ces tensions sur $[0,5\text{V}]$ et de plus produisent deux tensions antagonistes positives centrées autour de $V_{REF} = 2.5\text{V}$
- Expliquer l'intérêt que SUB_1 et SUB_2 produisent de telles sorties.
- Chercher l'opération arithmétique (indiquez la référence dans la documentation) que va faire l'[AD7934](#) sur chaque paire de signaux (VIN_0, VIN_1) et (VIN_2, VIN_3) .
- Quel est l'intérêt de ce montage différentiel ?

Registre de contrôle

PM₁ PM₀ CODING REF ZERO ADD₁ ADD₀ MODE₁ MODE₀ SEQ₁ SEQ₀ RANGE

- Le registre de contrôle est un mot de 12 bits.
- Il permet de programmer différents modes de fonctionnement.
- On l'initialise au début de programme.
- Donner le rôle de ces bits par fonctionnalités
- En déduire la valeur du mot pour la conversion du courant
- En déduire la valeur du mot pour la conversion de la vitesse

Ecriture du driver AD7934

Le driver va se faire en codant deux fonctions :

- Fonction *void programAD7934(int voie)*: Permet de définir le mode de fonctionnement de **AD7934** et de définir le canal sur lequel la conversion sera faite
- Fonction *int lireAD7934(void)*: Permet de lire le résultat de conversion de l' **AD7934** sur le canal prédéfini par l'autre fonction

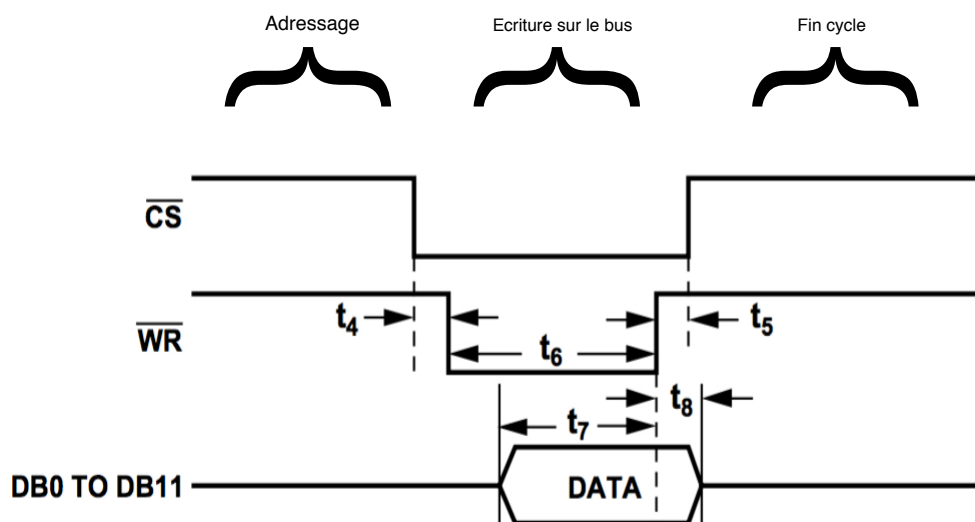
Fonction *void programAD7934(int voie)*: Ecriture du mot de contrôle l'*ATMega* sur l'*AD7934*

La programmation de l'*AD7934* se réalisera alors par 3 phases :

- Adressage : Le positionnement d'une valeur sur le bus d'adresse (*PORT_L*) qui permet d'activer et d'écrire sur le composant sur le bus d'adresse (avec insertion éventuelle de délais)
- Ecriture sur le bus de données (*PORT_A* et *PORT_C*)
- Fin de cycle : Le positionnement d'une nouvelle valeur sur le bus d'adresse qui désactive le composant et le mode écriture du composant sur le bus d'adresse

Chronogramme correspondant à la fonction *void programAD7934(int voie)*

Voici le chronogramme d'écriture de l'*AD7934* :



Description du chronogramme précédent

Dans la figure précédente :

- C'est la conjonction de $\overline{WR}$ et $\overline{CS}$ qui précède l'écriture d'un mot de 12 bits sur le bus de données. Attention vous devez positionner une valeur sur tous les bits d'adressage $\overline{WR}$, $\overline{RD}$, $W/\overline{B}$, $\overline{CONVST}$, $\overline{CS}$.
- Détaillons la figure précédente :
 - ▶ $\overline{CS}$ et $\overline{WR}$ active l'**AD7934**
 - ▶ La data est écrite dans l'**AD7934** après que le signal $\overline{WR}$ soit retombé au niveau bas.
 - ▶ L'écriture nécessite un temps de maintien t_7 avant la remontée de $\overline{WR}$.
 - ▶ $\overline{CS}$ et $\overline{WR}$ sont théoriquement liés par t_4 et t_5
 - ▶ Cherchez dans **AD7934Bruz**, la table des temporisations qui donne les valeurs de t_4 et t_5
 - ▶ Compléter maintenant la fonction `void programAD7934(int voie)` qui, selon la voie (COURANT ou VITESSE), initie les modes de fonctionnement de l'**AD7934**

Compléter `void programAD7934(int voie)()`

```
void programAD7934(int voie) {
    int cwrH = 0, cwrL = 0, adr;

    DDRA = ??; DDRC = ??;
    // baisser WR et CS
    adr = ??; PORTL = adr;

    // envoie du mot de controle : Mod1 Mod0 = 1 0 => Differential mode
    cwrH = ??;
    cwrL = ??;

    if (voie == COURANT) cwrL |= ??;
    else cwrL |= ??;

    PORTA = cwrL;
    PORTC = cwrH;

    // remonter WR et CS
    adr = ??; PORTL = adr;
}
```

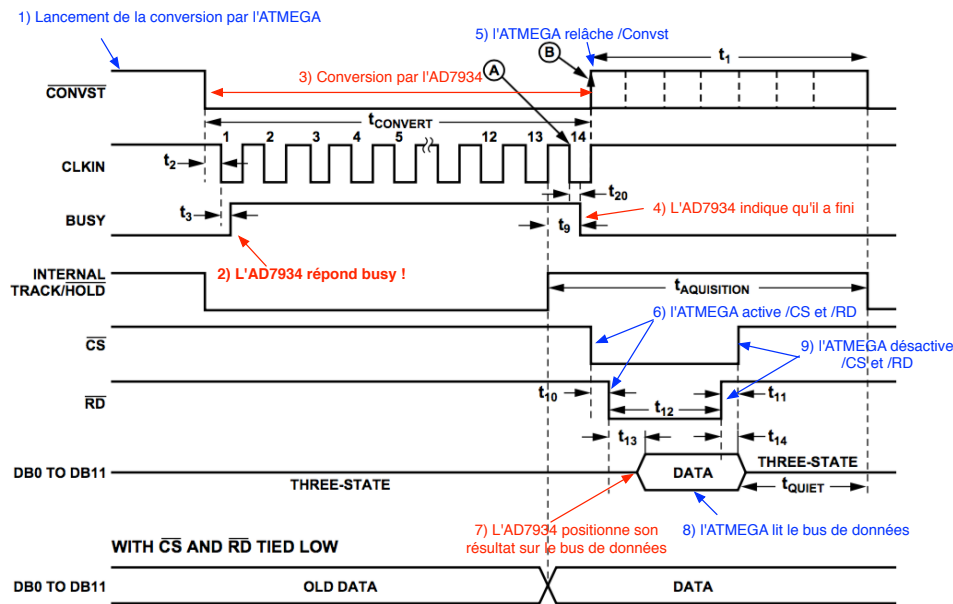


Figure 34. AD7933/AD7934 Parallel Interface—Conversion and Read Cycle in Word Mode ($W/\bar{B} = 1$)

Description du chronogramme précédent

Voici les phases principales du chronogramme précédent :

- 1 *ATMega* envoie le signal $\overline{CONVST}$ qui lance la conversion
- 2 *AD7934* active $BUSY$ pour indiquer qu'une conversion est en cours
- 3 $\overline{CONVST}$ et $BUSY$ restent à l'état haut pendant tout le cycle.
 - * Le temps de conversion dure 14 cycles d'horloge ($t_{convert}$)
 - * Entre 2 conversions $\overline{CONVST}$ est à l'état haut pendant t_1
- 4 Fin de la conversion *AD7934* relâche $BUSY$ pour informer l'*ATMega*
- 5 *ATMega* relâche $\overline{CONVST}$ et commence la lecture du résultat de la conversion
- 6 *ATMega* active $\overline{CS}$ et $\overline{RD}$: Adressage de l'*AD7934*
 - * $\overline{CS}$ et $\overline{RD}$ qui sont théoriquement liés par t_{10} et t_{11}
 - * Retrouver dans *AD7934Bruz* ces valeurs
- 7 Données placées par *AD7934* sur bus de données après retombée de $\overline{CS}$ et $\overline{RD}$
- 8 *ATMega* lit le résultat de la conversion sur le bus de données
- 9 *ATMega* désactive $\overline{CS}$ et $\overline{RD}$

Compléter *int lireAD7934(void)()*

Compléter *int lireAD7934(void)* en codant les étapes précédentes bleus

```
int lireAD7934() {
    int adr = 0, finConv = 1, IV;

    // /CONVST  adr = ?? ; PORTL = adr;

    // Attent du signal /BUSY

    // Convstart doit durer au minimum 14 * 125 nano sec = 1,750 mus
    _delay_us(2);

    // relacher CONVST  adr = ??; PORTL = adr;

    // Lecture du bus de données

    // Masquage et recentrage valeurs lues
    IV &= 4095; IV = IV- 2047;

    // Fin du lecture :  adr = ??; PORTL = adr;
    return (IV);
}
```

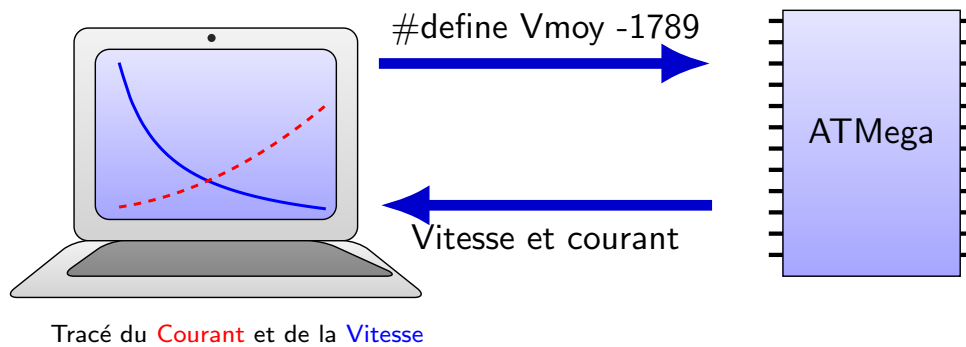
Navigation icons: back, forward, search, etc.

Test de *int lireAD7934(void)* et *void programAD7934(int voie)*

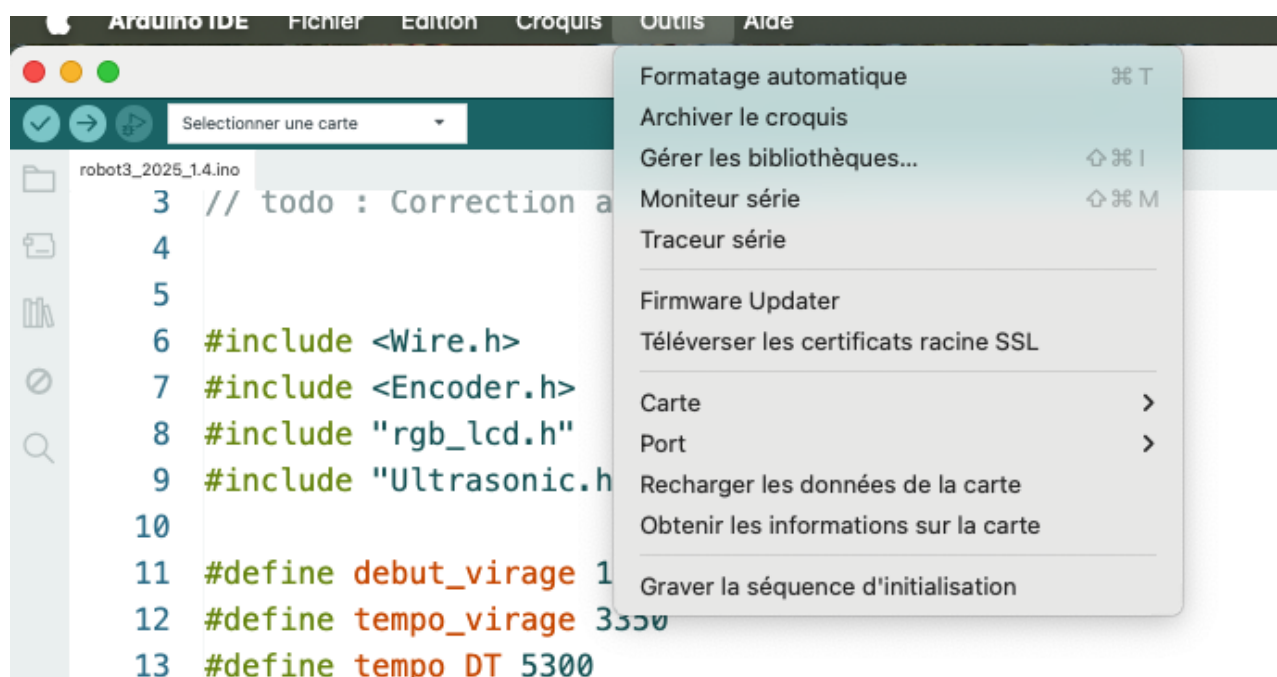
- Tester maintenant les fonctions *void programAD7934(int voie)* et *int lireAD7934(void)* en lisant alternativement le courant et la vitesse que vous afficherez sur le traceur série.
- Appliquer une charge sur le moteur : Expliquer l'évolution des courbes.

Navigation icons: back, forward, search, etc.

Traceur



Traceur serie



Traceur serie

- Dans le code, les valeurs des courbes sont séparées par une virgule.
- La dernière valeur d'une courbe est suivie par le retour à la ligne.
- Exemple avec 2 courbes *Courant* et *Vitesse* :

```
void setup() {  
  Serial.begin(9600);  
}  
  
void loop() {  
  Serial.print(Courant);  
  Serial.print(',');  
  Serial.print(Vitesse);  
  Serial.print("\n");  
}
```

Traceur serie

