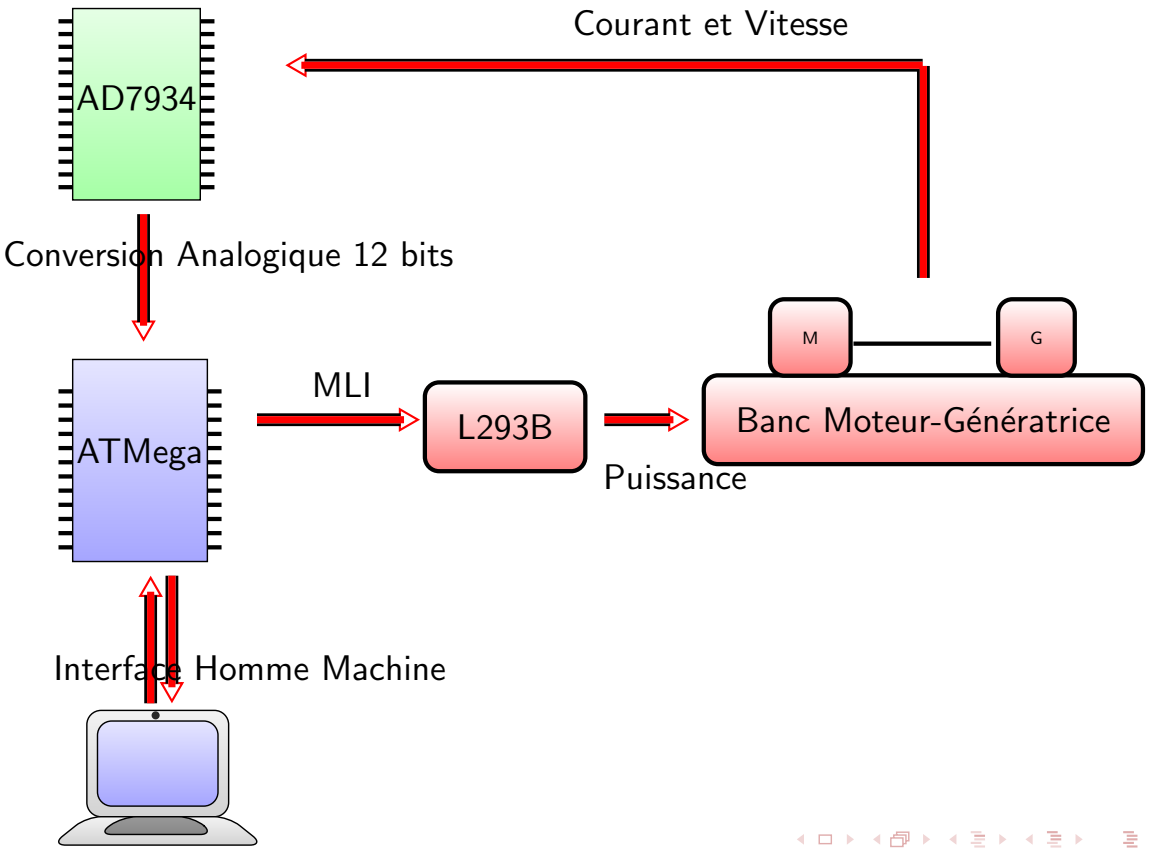
Micro-Projet d'Informatique Industrielle S7

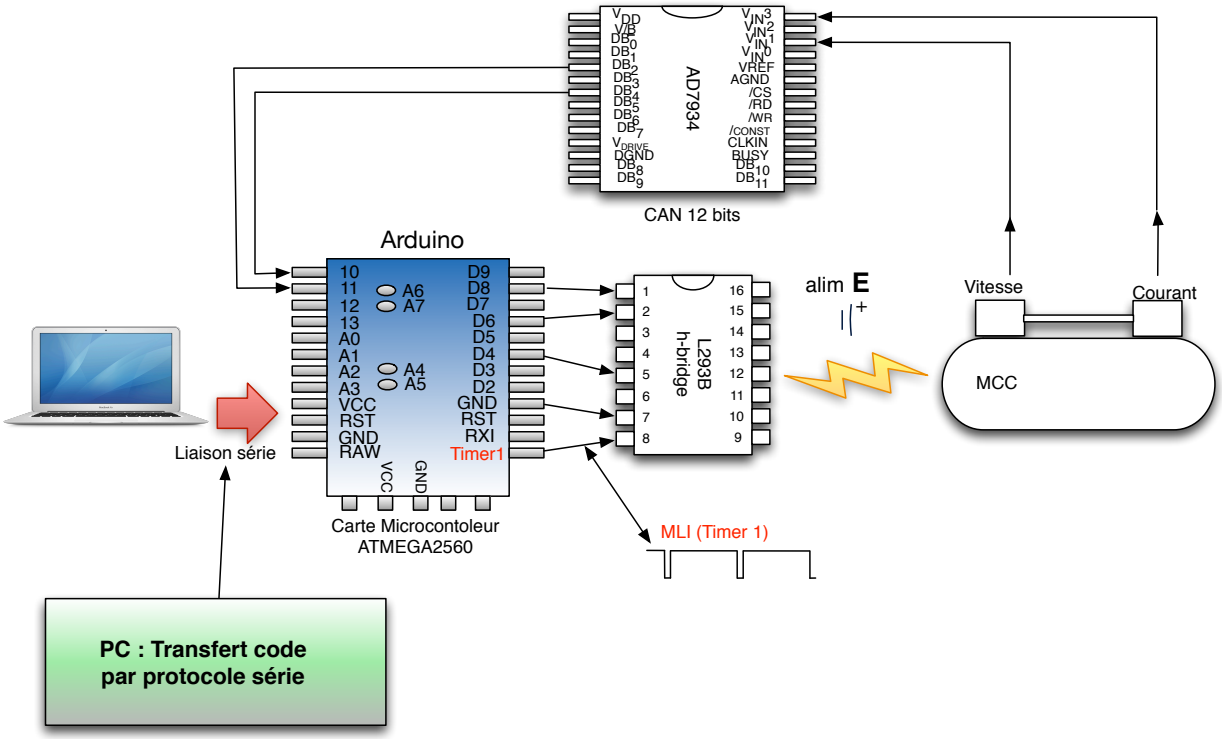
Nadia Aït Ahmed, David Delfieu

October 16, 2023

Commande en boucle ouverte d'une *MCC*



Conversationnel sous Arduino



Protocole série

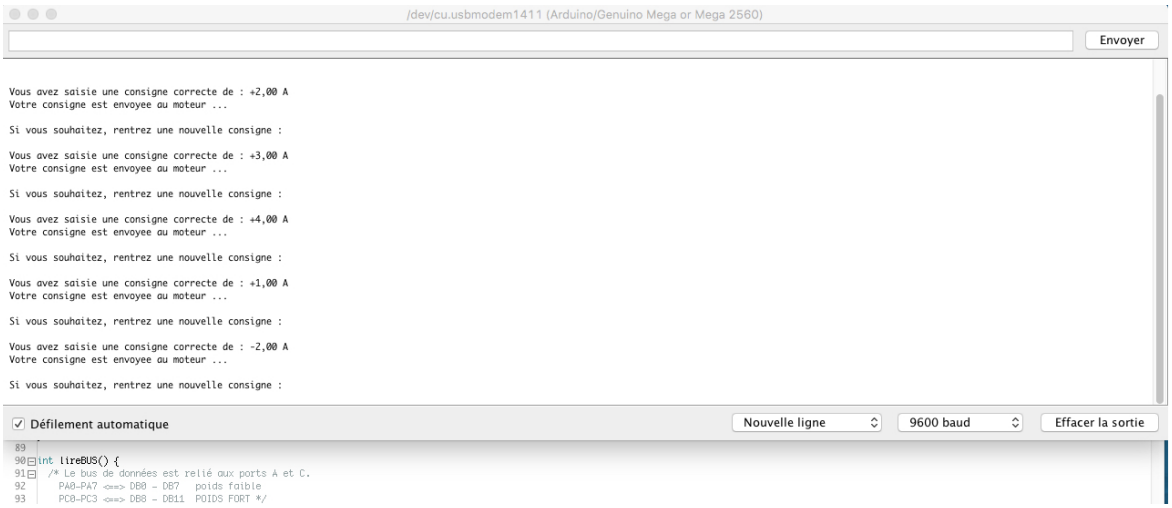
- Protocole RS232 série : Transmission d'octet
- Physique : 3 fils, émission, transmission, masse.
- Arduino : pont RS232-USB
- RS232 $\Rightarrow$ bibliothèque *Serial*
- Pins en TTL (5V ou 3.3V):
  Transmission TxD (PD3)
  Reception RxD (PD2)
- $ATMEGA_{2560}$: 3 ports séries additionnels:

Le Moniteur Serie

- Plusieurs liaisons série
- Outil "Moniteur série".

Le Moniteur Serie

- Une fenêtre avec deux panneaux
- Cette fenêtre affichée sur le PC modélise les entrées sorties de l'Arduino : Clavier/écran
  - Une ligne de saisie
  - Une écran d'affichage



Configuration du moniteur série

- Baud rate : 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, ou 115200.
- Second argument :

  bit de parité,
  bit de stop.
- Réglage 8N1 : 9600 bauds, 8 bits, pas de bit de parité,bit de stop.
- Instruction : Syntaxe Serial.begin(speed)
  Syntaxe Serial.begin(9600, SERIAL_8N1);

Serial.print

- Affiche à l'écran du PC, les données envoyées par l'Arduino qui transitent par le bus USB au format ASCII.
- Les nombres sont affichés caractères par caractères en ASCII.

```
S=Serial.read();
A=Serial.read();
B=Serial.read();
Si l'utilsateur tapes +15 alors
 S vaut \$2B : code ascii de +
 A vaut \$31 : code ascii de 1 (48 en base 10)
 A vaut \$35 : code ascii de 5 (53 en base 10)
```

- Les Flottants sont affichés de la même façon, avec deux décimales par défaut.
- affichage de valeur et de chaines :

```
Serial.print("\n Voici la valeur de A = ");
Serial.print(A);
```


Lecture sur le bus série

- Le bus série permet un communication de type asynchrone :
  - Celui qui écrit envoie des données dans le bus des données qui se remplit
  - Celui qui lit consomme des données dans le bus qui se vide alors
- La lecture sur le bus se fait en mode caractère et produit donc des codes ASCII.

readCar()

- ReadCar() récupère un caractère dès qu'il apparait sur le bus :

```
int readCar() {
   while (Serial.available() == 0); //Attente TQ il n'y a rien sur bus
   return(Serial.read());
}
```

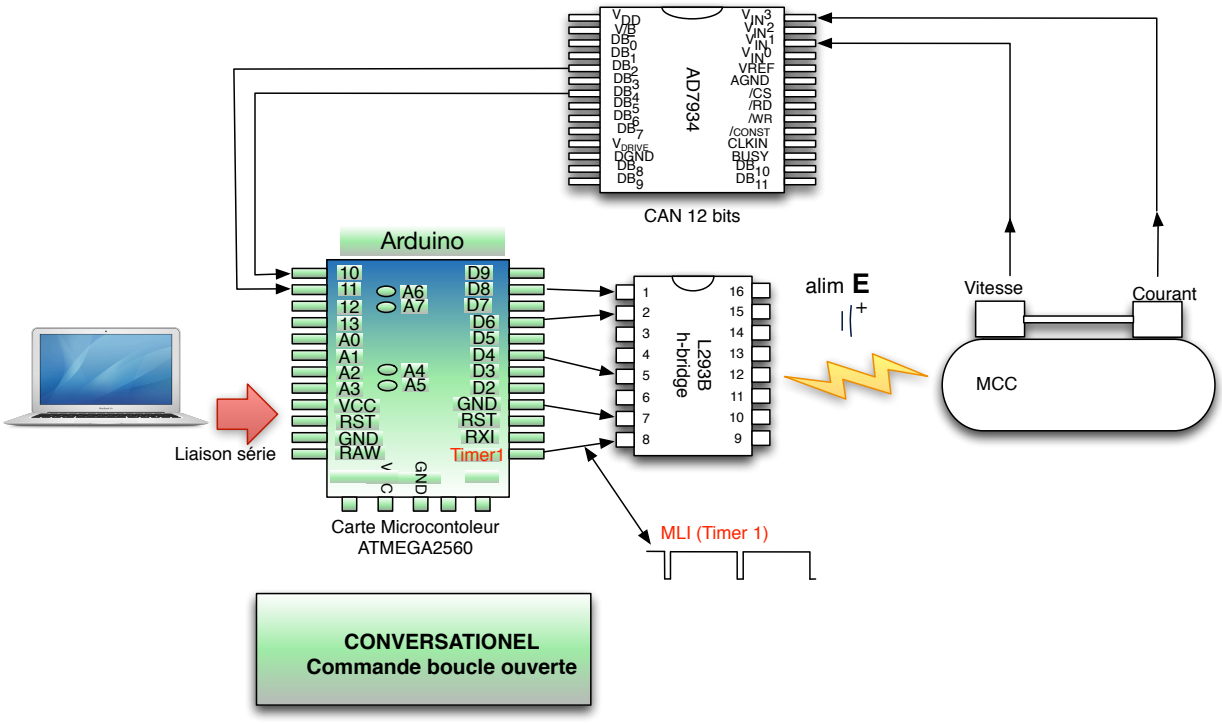
- Utilisation de readCar() :

```
int carLU = 0;   // caractère lu sur le bus série

void setup() {Serial.begin(9600);}

void loop() {
    carLU = readCar(); // On lit un caractère sur le bus
    Serial.print("J'ai reçu : ");
    Serial.println(carLU, DEC);
}
```


Conversationnel

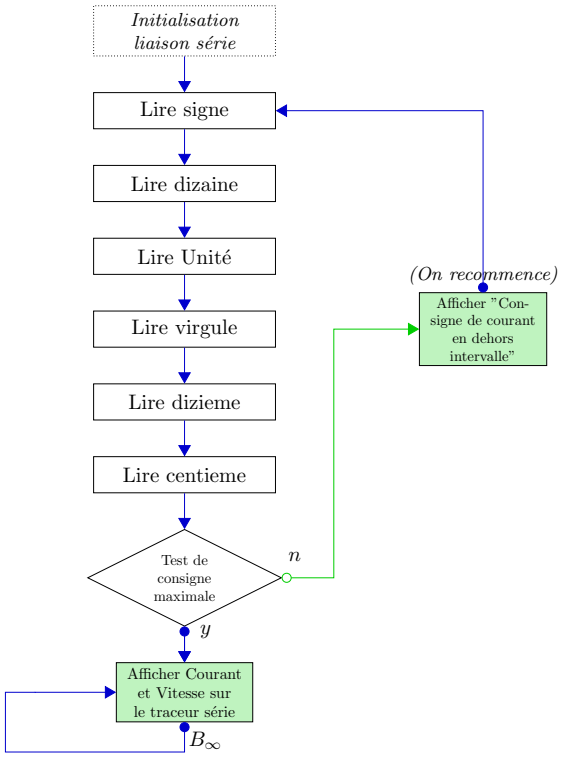


Conversationnel

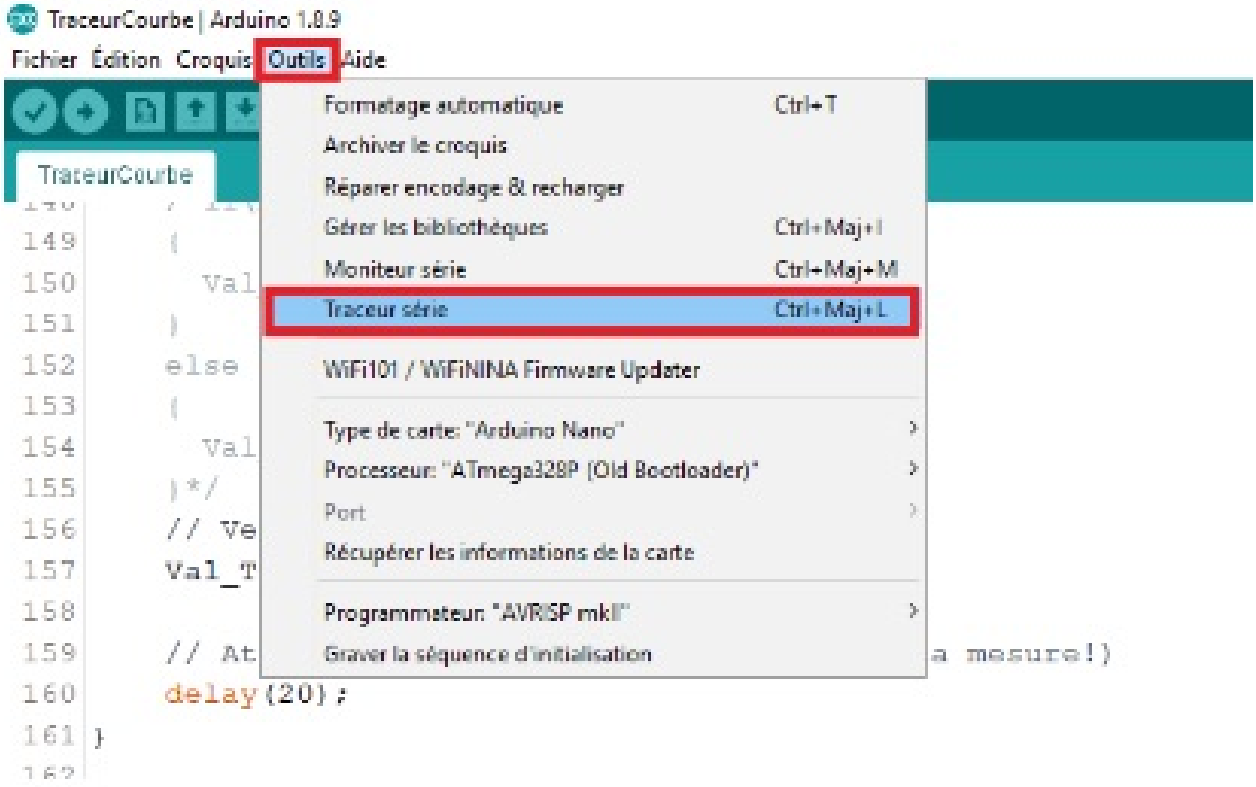
Interface conversationnelle entre le micro-contrôleuret le moniteur sériel :

- Après un message d'invite
- On saisit une seule consigne en tension de commande de la *MCC* avec deux chiffres de précision après la virgule.
- Après la saisie on bouclera infiniment sur l'affichage courant ou vitesse.
- Coder, à partir des fonctions qui vous sont données dans le squelette, l'organigramme du slide suivant :

Organigramme simple, non robuste



Traceur serie



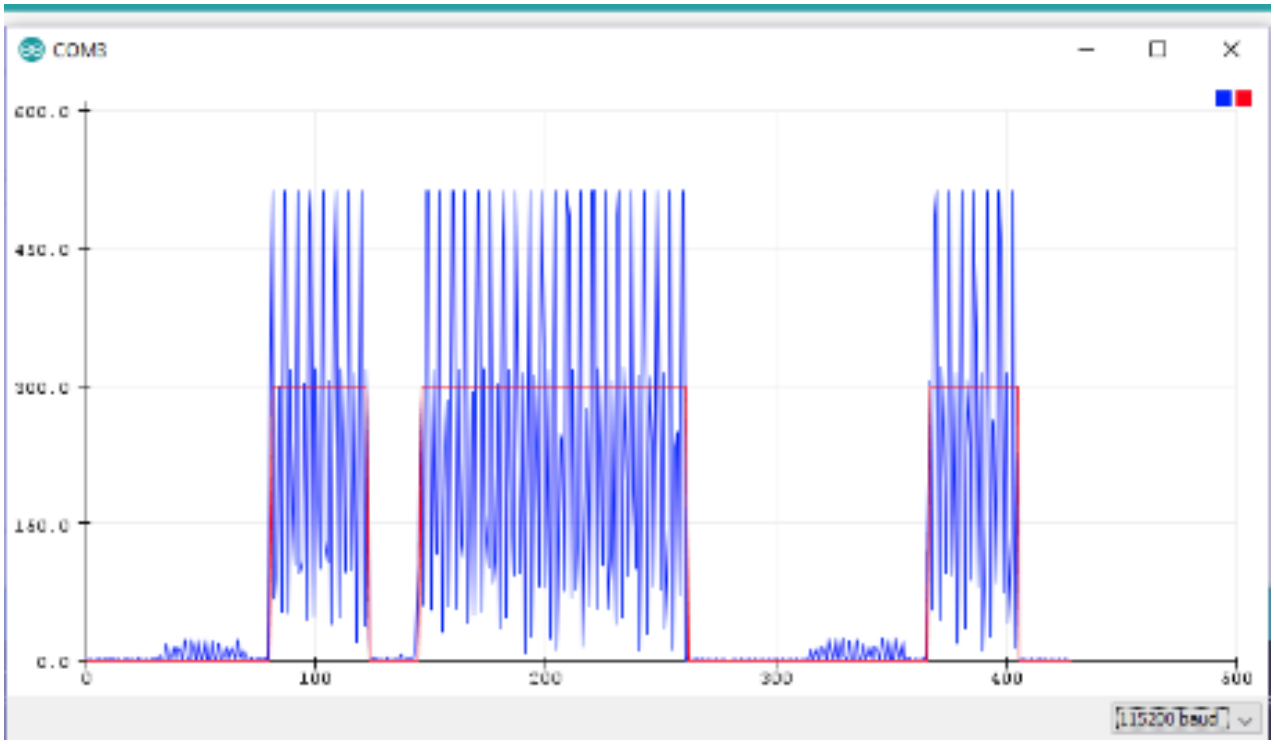
Traceur serie

- 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* :

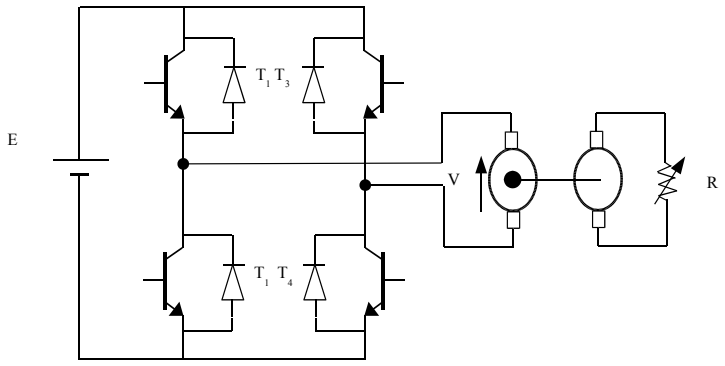
```
void setup() {
  Serial.begin(9600);
}

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

Traceur serie

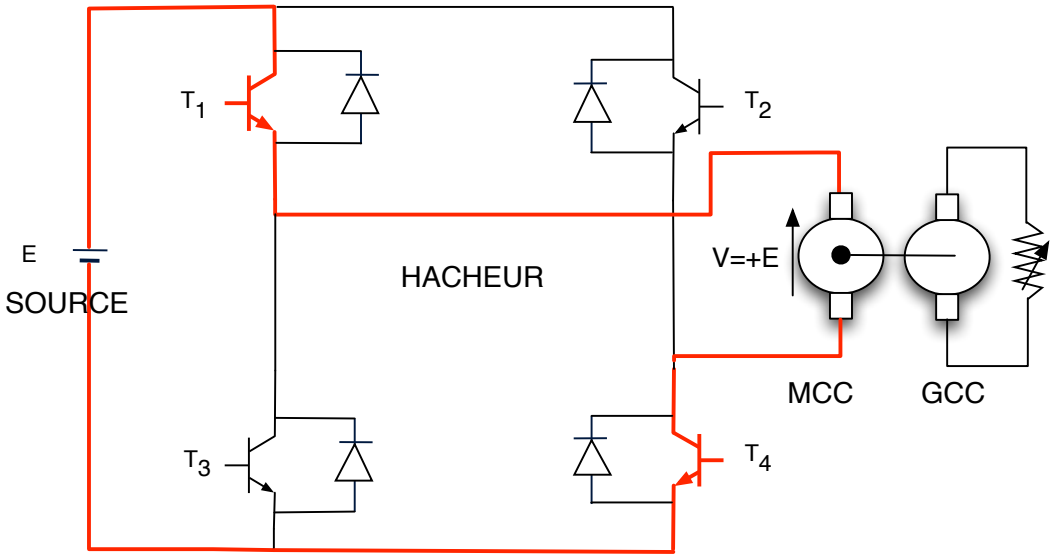


Principes de commande du hacheur (L293 H-Bridge)



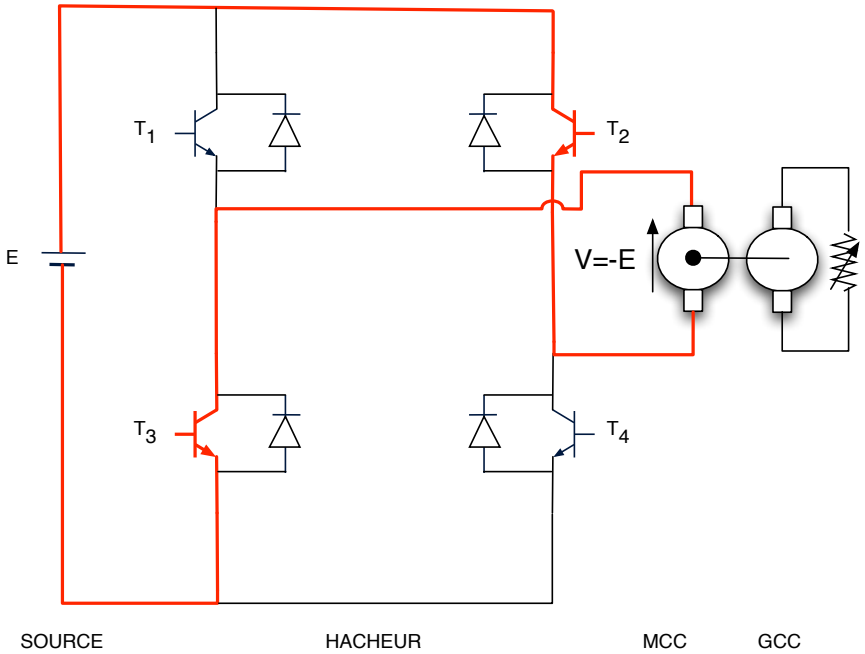
SOURCE HACHEUR MCC GCC

Principes de commande du hacheur



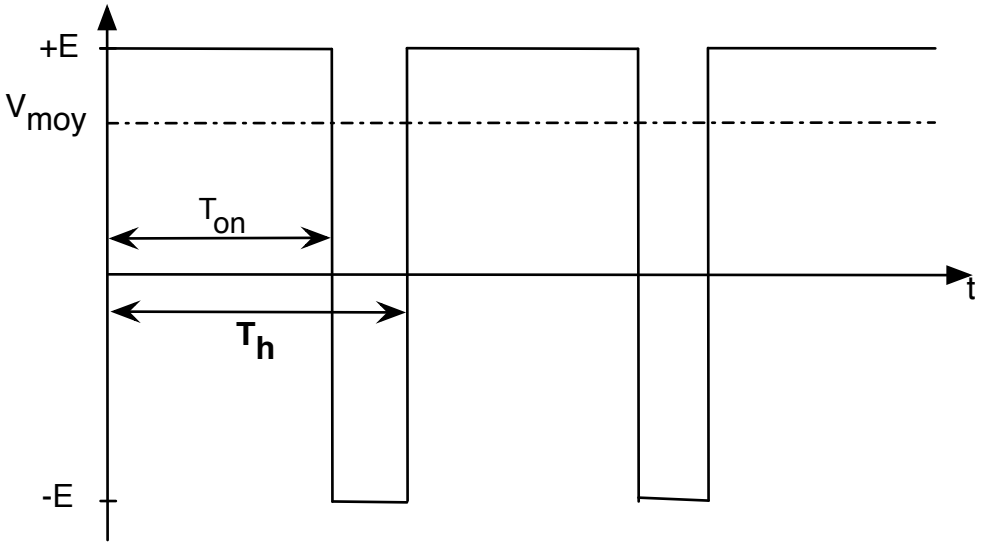
- T_1 et T_4 passant on a $V = +E$ aux bornes de la MCC

Principes de commande du hacheur



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

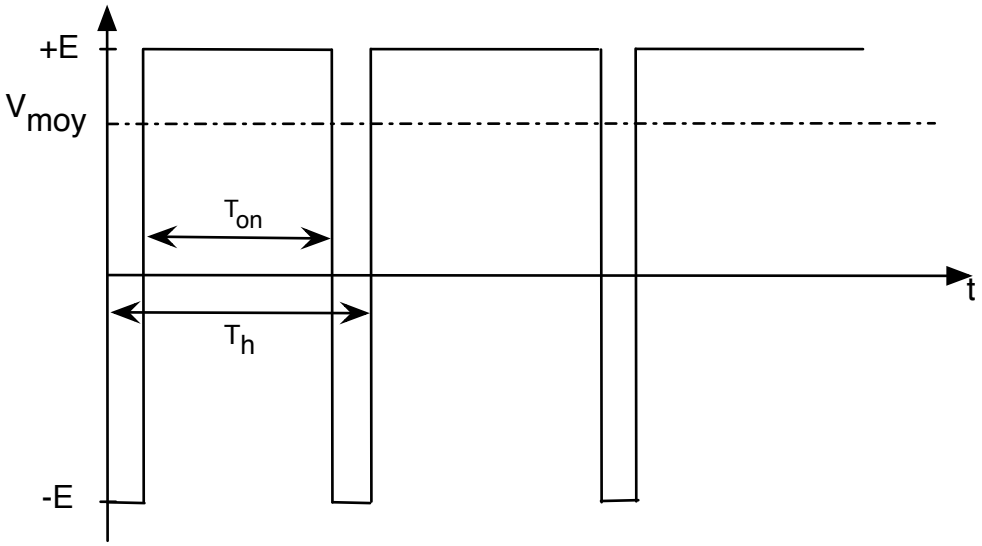
Consigne de tension hachée : T_{on} non centré



T_{on} non centré.

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

T_{on} centré sur T_h



T_{on} centré.

Numérisation du Calcul

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

$$\frac{T_{on}}{T_h} = \frac{T_{on}^N}{T_h^N} = \frac{OCR_{1A}}{OCR_{1B}}$$

Conversationnel en virgule fixe : V_{moy}

C'est le conversationnel qui fournit V_{moy} :

- Si on considère que l'utilisateur a tapé $+21,37$
- On récupère la valeur entière (Virgule fixe) 2137 = 100 * Saisie
- Conversationnel produit donc $100 * V_{moy}$.
- E est connu : $E = 35\ v$.
- T_h est déterminé par les caractéristiques électriques du moteur : $T_h = 200\mu\ s$
- D'après l'équation précédente :

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

- Donc pour produire V_{moy} avec un timer je vais devoir déterminer T_{on} :

Calcul de V_{moyen}

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

- Calculer cette intégrale et exprimer V_{moy} en fonction de E, T_{on}, T_h
- Calculer l'expression littérale du rapport $\frac{T_{on}}{T_h}$

Mise en oeuvre de la MLI

- Timer produit T_h^N (OCR_{1A}) et T_{on}^N (OCR_{1B})
- Timer devra générer une MLI centrée
- Trouver dans ATmega2560_datasheet la table des modes du timer 1
- Quel terme est utilisé dans cette documentation pour désigner une MLI centrée ?

Les Timers

Dans la documentation ATmega2560_datasheet :

- Trouver dans la documentation la formule qui permet d'établir T_h^N. Donner la référence.
- A partir de la figure 17.1 de la page 134, retrouver les 4 registres qui sont importants pour la génération d'une MLI sur la sortie B, c.a.d $OC1B$.
- Rechercher dans la doc les rôles des 4 registres et donner les références.
- Dans la table des 16 modes des timers, déterminer quels sont les modes possibles et qu'est-ce qui les différencie ?
- quelles valeurs doit-on mettre dans WGM_{13} WGM_{12} WGM_{11} WGM_{10} ?

Rôles de OCR_{1B} et OCR_{1A}

- Trouver la figure qui montre dans un chonogramme 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^{ieme} ligne de ce tableau.

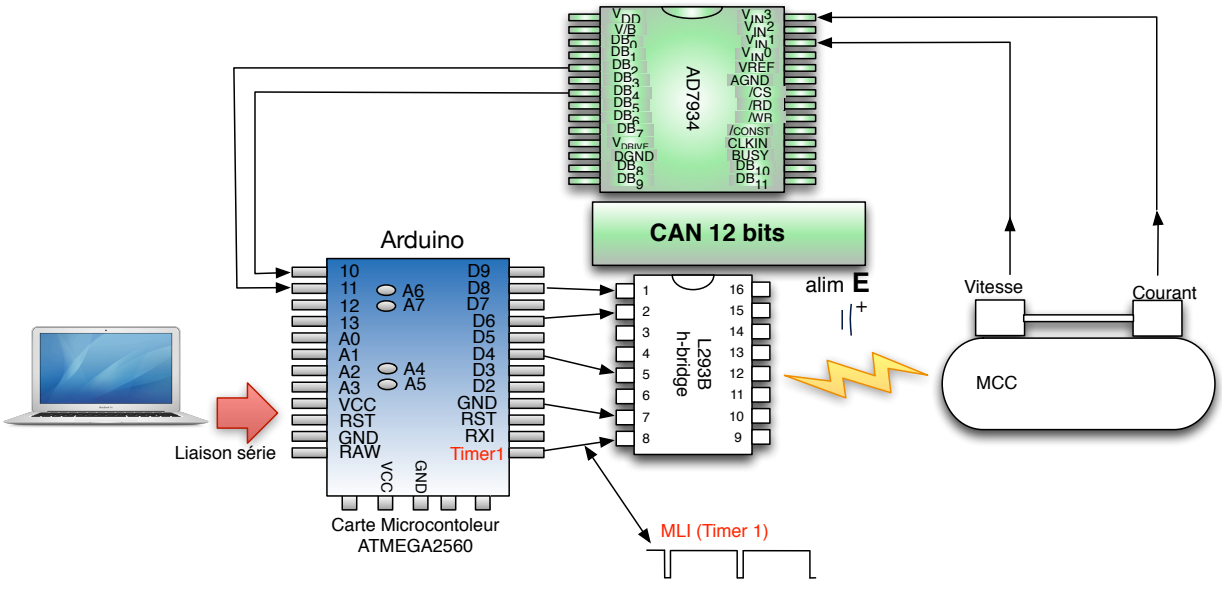
Initialisations des registres $TCCR_{1A}$ et $TCCR_{1B}$

- Donner les lignes d'initialisations des registres $TCCR_{1A}$ et $TCCR_{1B}$

```
Exemple d'initialisation du Mode 11 :
  TCCR1A = (1<< WGM11) + (1<< WGM10);
  TCCR1B = (1<< WGM13)+(1<<CS10);
```

- Compléter alors la fonction *initTimer*(), de façon à avoir une fréquence de hachage de $20khz$ et un rapport cyclique initial de 0.5
- Tester cette fonction

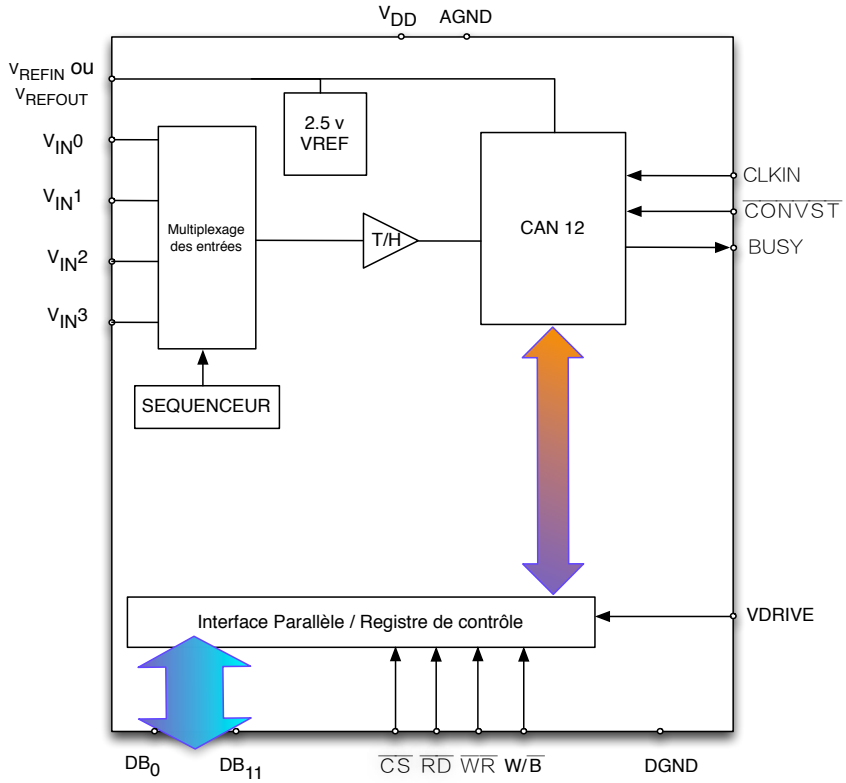
Convertisseur AD7934



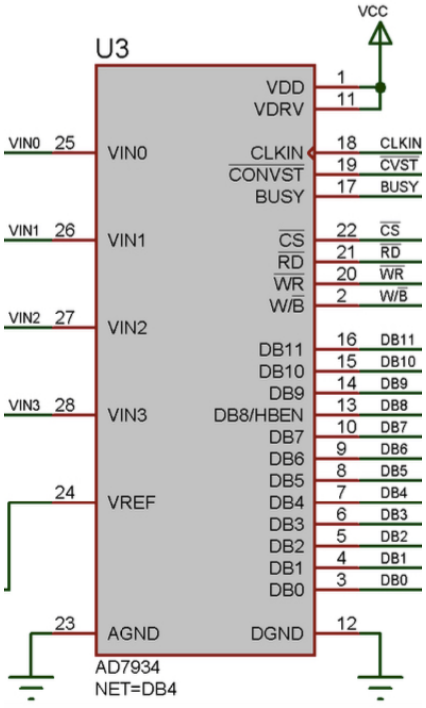
Convertisseur AD7934

- CAN 12 bits rapide 1.5 MSPS (Million Samples Per Second)
- 4 entrées analogiques séquençables en mode simple ou en mode différentiel
- Possibilité de connecter l'*AD7934* à un bus d'adresse ($\overline{CS}$,$\overline{RD}$,$\overline{WR}$,$W/\overline{B}$) et un bus de données ($DB_0, ..., DB_{11}$) pour une lecture rapide de la conversion.
- Possibilité d'utiliser une référence interne précise de 2.5 v ou une référence externe (*VREF*).
- Echantillonnage déclenché par le signal $\overline{CONVST}$

Convertisseur AD7934

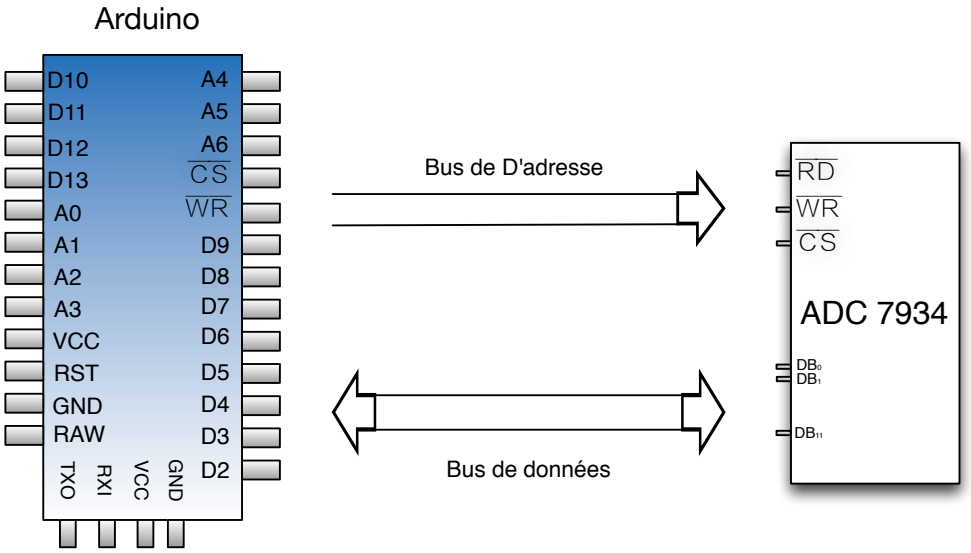


Présentation



Bus d'adresses - Bus de données

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

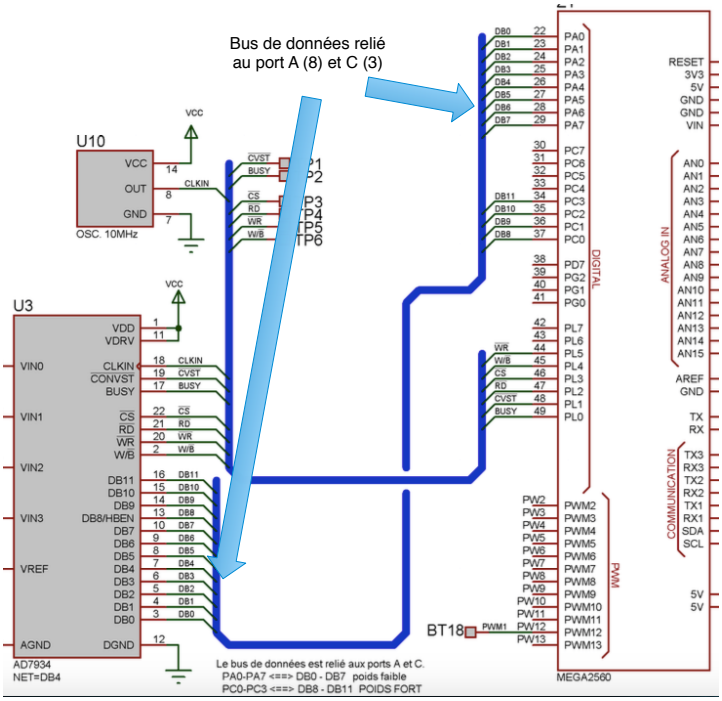


Bus de données

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

- 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

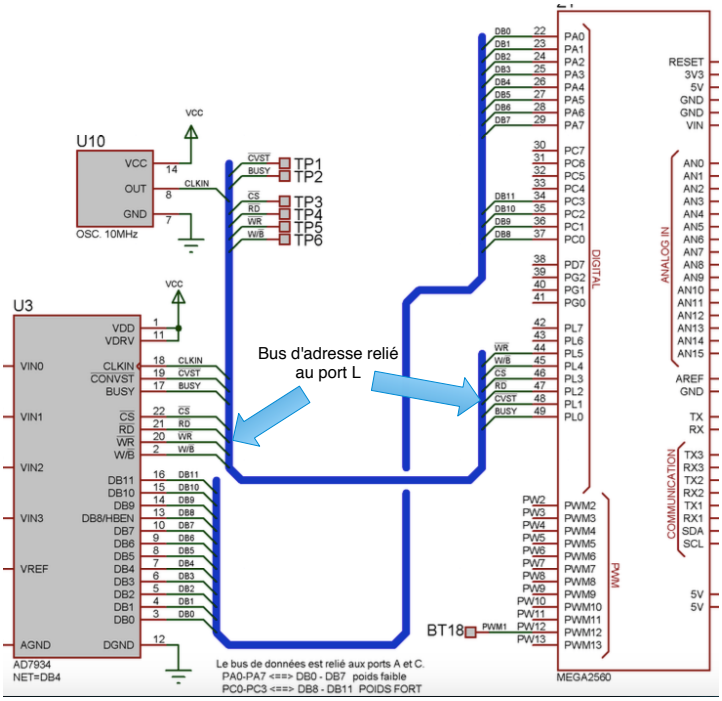


Bus d'adresse et de contrôle

Bus d'adresse et de contrôle :

- $\overline{CS}$, $\overline{WR}$, $\overline{RD}$ et $W/\overline{B}$ servent à adresser le composant
- $BUSY$, $\overline{CONVST}$ et $CLKIN$ servent à contrôler le composant

Bus d'adresses (et de contrôle)



Analyse des liasons 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 réference (numéro de page).
  - Exemple pour $\overline{CONVST}$: Un front descendant de ce signal démarre une conversion (p 23, AD7934Bruz) .

Entrées analogiques de l'*AD7934*

A partir de la documentation CarteTP_ATMega :

- Identifier les entrées analogiques du convertisseur.
- A partir de la documentation AD7934Bruz expliquer comment ces entrées anlogiques peuvent être utilisées suivant 4 modes différents.
- En examinant ce montage, quel est le mode utilisé ?

Entrées differentielles

A partir de la documentation CarteTP_ATMega et de AD7934Bruz :

- Les *bnc* J_1 et J_2 acceptent des tensions sur l'intervalle $[-10, +10v]$
- Sachant que $V_A = J_1/2$ et $V_B = J_3/2$.
- Expliquer pourquoi 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 cette opération arithmétique.
- Donner un exemple de tension sur J_1, calculer la paire VIN_0, VIN_1, et donner le résultat de l'opération de l'*AD7934* avant la conversion. Expliquer ce qui se passe.

Registre de contrôle

- 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.

Mot de contrôle

PM_1 PM_0 $CODING$ REF $ZERO$ ADD_1 ADD_0 $MODE_1$ $MODE_0$ SEQ_1 SEQ_0 $RANGE$

- Donner le rôle de ces bits par fonctionalités
- En déduire la valeur du mot pour la conversion du courant
- En déduire la valeur du mot pour la conversion de la vitesse

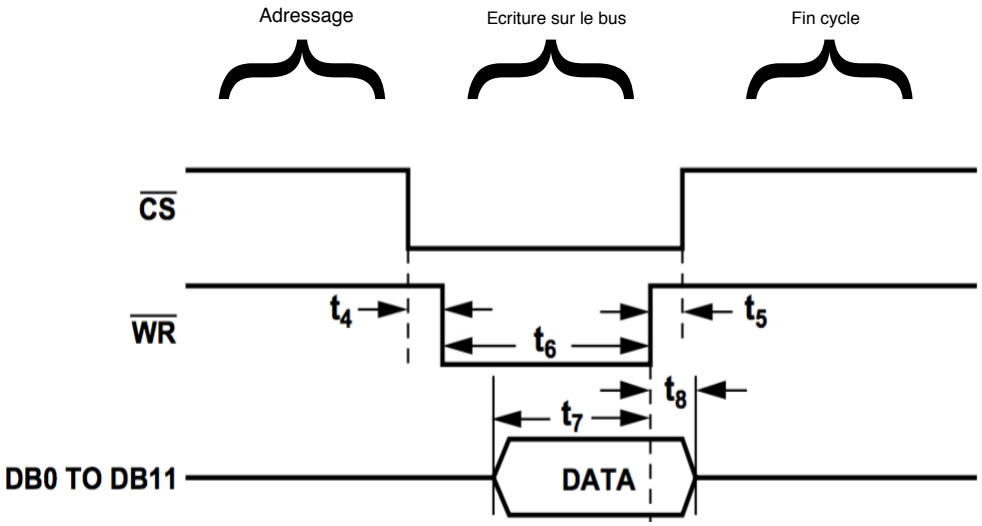
Ecriture du mot de controle l'*ATMega* sur l'*AD7934*

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 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 d'écriture sur l'*AD7934*

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



Chronogramme d'écriture sur l'*AD7934*

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.
- Détaillons la figure précédente :
  - $\overline{CS}$ et $\overline{WR}$ active l'*AD7934*
  - La data est écrite dans l'*AD7934* après que le signal $\overline{WR}$ soir 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
  - Completer maintenant la fonction *programAD7934*(*int voie*) qui, selon la voie (COURANT ou VITESSE), initie les modes de fonctionnement de l'*AD7934*

Conversion sur l'*AD7934* et lecture du résultat

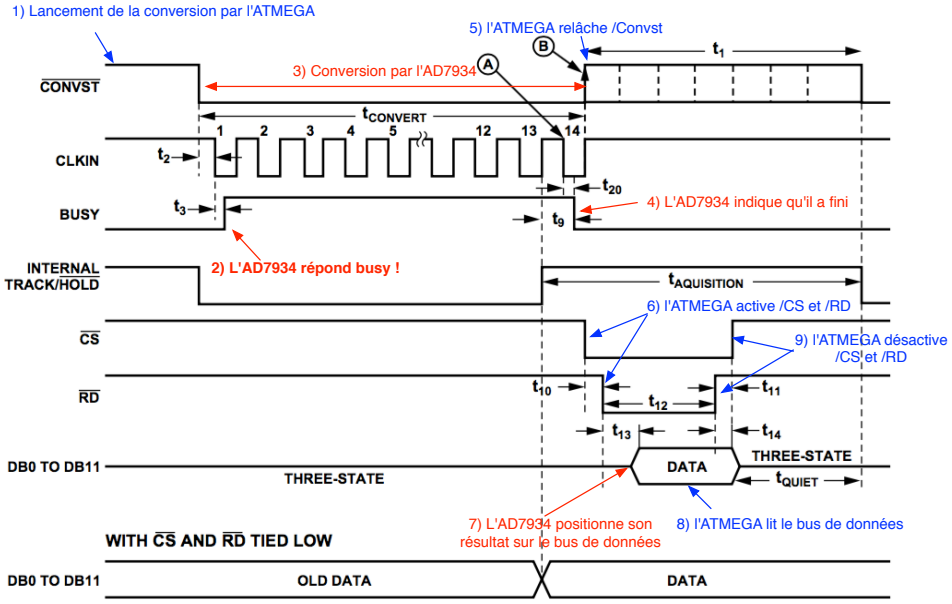


Figure 34. AD7933/AD7934 Parallel Interface—Conversion and Read Cycle in Word Mode ($W/\overline{B} = 1$)

Cycle de conversion et lecture du résultat sur l'*AD7934*

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 bas 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 $\overline{RD}$

lireAD7934()

A l'aide des deux slides précédents :

- Completer la fonction *lireAD7934()*
- Tester maintennant les fonctions *programAD7934(int voie)* et *lireAD7934()* en lisant alternativement le courant et la vitesse que vous afficherez sur le moniteur série.
- Afficher maintenant le courant et la vitesse par le traceur série.