Gestion de boite de vitesse séquentielle / automatique.

Avant propos :

Les travaux pratiques d'ingénierie microcontrôleurs sont répartis sur 3 séances, un seul compte-rendu par binôme sera ramassé à la fin de la 3ème séance, la travail des binômes est aussi évalué en cours de séance. C'est le même exemple applicatif qui servira de trame à l'organisation des TPs : la gestion électronique d'une boite de vitesse automatique ; cependant il ne s'agit pas de développer réellement la commande d'une boite automatique mais plutôt d'illustrer concrètement quelques concepts de l'ingénierie microcontrôleur à travers cet exemple. Aussi, après une présentation rapide de la boite de vitesse et de la problématique de sa gestion automatique, nous serons amenés à formuler des hypothèses simplificatrices et à énoncer les fonctionnalités logicielles à développer.

La boite automatique :

La boite de vitesse est un transformateur mécanique, il adapte une force primaire pour la mettre sous forme utile : la puissance du moteur doit être adaptée au récepteur, le véhicule, via la force de propulsion au contact pneu/route. Un véhicule en mouvement consomme une puissance qui dépend de sa vitesse et de l'inclinaison de la route (fig. 1), un moteur fournit une puissance en fonction de sa vitesse de rotation (fig. 2), en reportant la courbe de puissance consommée par le véhicule en mouvement vue du moteur, on comprend la nécessité de l'accord : c'est le rôle de la boite de vitesse (ou réducteur/multiplicateur). Les rapports de boite permettent d'adapter le régime du moteur à la vitesse du véhicule (fig. 3), le choix des rapports est sujet à de nombreux compromis. L'automatisation de la boite permet de sélectionner le meilleur rapport selon un critère choisi (mode éco, sport, …) ainsi que de s'adapter à l'état de la boite (température, vieillissement, …).

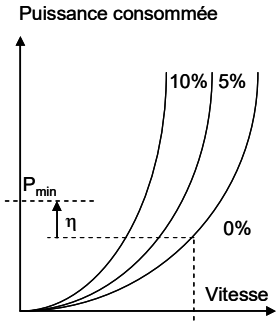


Figure 1

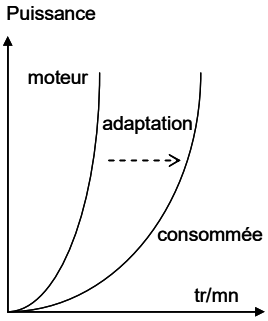


Figure 2

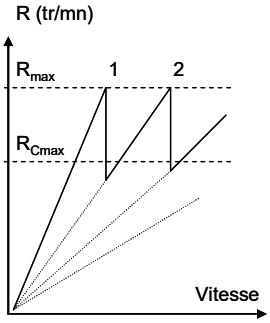


Figure 3

Organisation de la boite automatique :

Un schéma illustratif est donné à la figure 4, la boite est organisée selon différentes fonctions/organes :

- Interface conducteur : on retrouve les sélecteurs de vitesse (PRND, park, reverse, neutre, drive), de mode (sport, eco, …), ou de rapport (+, -) ; les capteurs de conduite (contacteur de pédale de frein, de pleine accélération) ; les afficheurs.
- Calculateur d'injection : il est interfacé au calculateur de boite car le changement de rapport, qu'initie le calculateur de boite, doit s'accompagner d'une diminution du couple du moteur qui est liée à une baisse de l'injection de carburant géré par le calculateur d'injection. Ce calculateur communique la valeur de la charge du moteur (l'ouverture du « papillon ») et reçoit la commande d'estompage de couple.

- Calculateur de la boite de vitesse : il gère le passage des vitesses et les différents modes d'utilisation, les commandes et consignes de l'utilisateur, l'affichage, l'auto-adaptativité des programmes (à la température de l'huile de boite, au vieillissement, …), … C'est la programmation de ce calculateur qui servira d'illustration pour ce TP.
- Eléments mécaniques de la boite : arbres primaire, secondaires et de sortie ; pignons ; embrayages ; capteurs de vitesse d'entrée et de sortie, …
- Commande hydraulique[1] : les différents actionneurs et distributeurs (qui permettent le changement de vitesse et l'embrayage), les électrovannes qui les pilotent, leurs relais électriques de commande, la pompe, …
- Les capteurs : de vitesse d'entrée / de sortie de boite, de température et pression d'huile, de pleine accélération et de freinage.

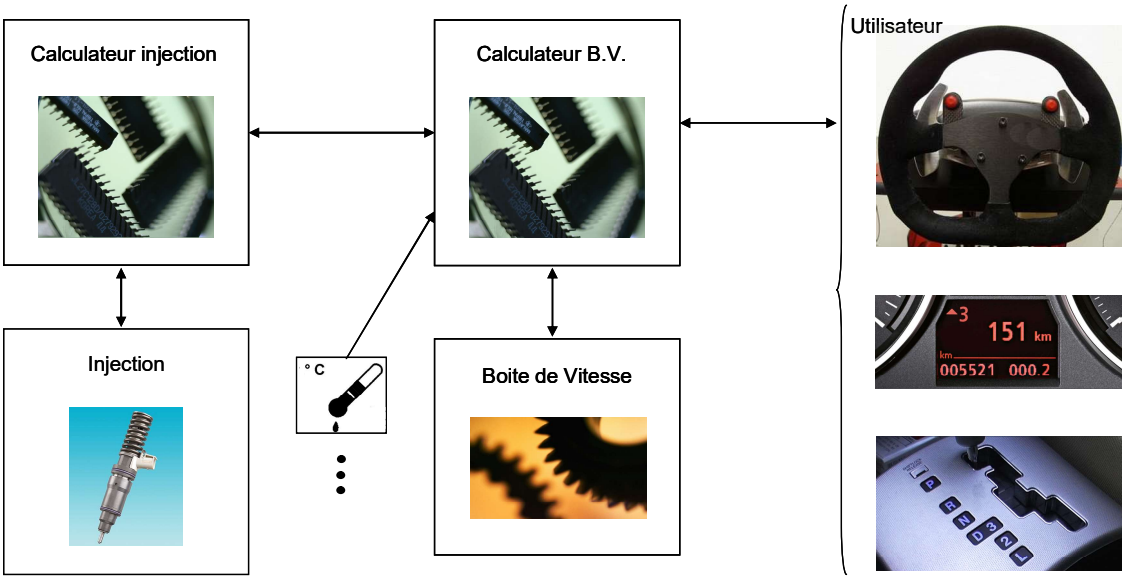


Figure 4

Hypothèses simplificatrices :

Afin de programmer le calculateur BV nous considérons que :

- L'interface conducteur est constituée d'un écran LCD, d'un sélecteur de mode (roue codeuse) et de boutons poussoirs pour engager les rapports en mode séquentiel.
- Calculateur d'injection : le calculateur BV envoie la commande d'estompage de couple par un front montant, la valeur de la charge n'est pas prise en compte.
- La commande hydraulique est pilotée comme suit : envoi du code rapport à engager codé sur 3 bits, suivi d'un front montant de commande (temporisation : suit la commande d'estompage de couple, normalement la tempo est calculée en fonction du passage des vitesse et de la différence de vitesse entrée / sortie de boite ; on considère une tempo constante de 150 ms)
- Le seul capteur que nous considérons est celui de sortie de boite (proportionnel à la vitesse du véhicule), il fournit 20 « tops » par tour par usinage d'une roue sur l'arbre de sortie.
- Nous ne traitons pas le caractère adaptatif de la commande

[1] ou électrique selon les constructeurs

Travail demandé :

Ecrire un code assembleur pour 8051 qui gère la boite de vitesse en développant des routines pour :

- récupérer l'information vitesse dans une plage 10-100 km/h (utilisation des temporisateurs) un critère de 5% sera admis sur la précision
- afficher la vitesse à l'écran (voir annexe LCD), le mode et le rapport enclenché
- gérer le passage automatique des rapports en fonction de la vitesse (mode auto)

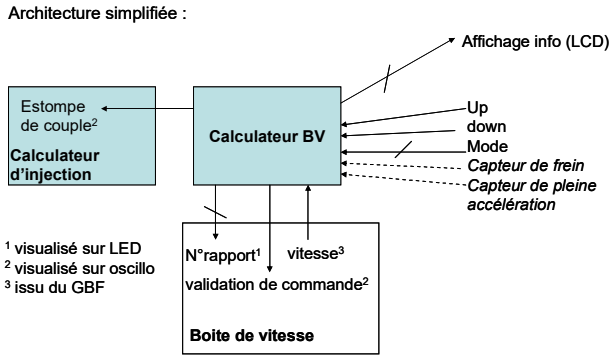
Puis, selon votre avancement améliorer le code : développer le mode séquentiel, prendre en compte en compte le capteur pleine accélération, prendre en compte le capteur freinage, prendre en compte la charge moteur, …

En pratique :

Vous développez les codes sous l'environnement Keil microvision (CD d'installation sur demande), vous disposez d'un kit de développement AT89C51 d'Atmel, d'un GBF et d'un oscilloscope et pour les interfaces :

- up / down = bouton poussoir sur le kit Atmel
- mode = roue codeuse
- vitesse = GBF
- affichage utilisateur = LCD
- affichage rapport engagé / boite de vitesse : utilisation de LEDS

Mode auto : piloter la boite selon la vitesse
Boite de vitesse : coder la vitesse à enclencher(N°rapport) / 3 bits, ajouter le signal de commande temporisé
Mode séquentiel : selon la vitesse, afficher quand changer de rapport (+ ou -), changement de rapport avec up / down



Signaux à prendre en compte pour le développement des codes du calculateur BV

Avancement suggéré :

- TP 1 Codes pour mesurer la fréquence du GBF et calculer la vitesse
- TP 2 Gérer l'affichage sur LCD ; Implémenter la récupération des info up / down ; Mode
- TP 3 Changer automatiquement de vitesse et améliorations diverses

Annexes :

Comparaison de deux mots :

L'instruction CJNE permet d'écrire une routine de comparaison. Cette instruction Compare et réalise un saut (Jump) si les mots sont Non Egaux, de plus le bit de retenue (C, carry) est mis à 1 si le mot 1 (M1) est inférieur au mot 2 (M2), mis à 0 sinon. Par exemple :

```
clr C
mov A,M1
cjne A,M2,suite
suite :
jc inf ; si c = 1 (ie M1 < M2) saute à inf
sup :
...
inf :
...
ret
```

Au besoin, l'égalité aura été testée avant (*subb A,B / jz egal*).

Utilisation d'un afficheur LCD :

L'écran est interfacé au microcontrôleur par un bus de données 8 bits, par un signal de commande RW pour « read write » (seulement l'écriture sur l'écran sera considérée ici), un signal de sélection des registres RS, et un signal d'autorisation (E). Ci-joint une routine pour initialiser l'écran (init_LCD), envoyer une instruction (send_data) et écrire une donnée (write_data) ; le bus de données est connecté à P2, RS à P1.2 et P1.0 à E :

```
init_LCD:
        mov r0,#01h          ; initialisation de l'écran0
        lcall send_inst
        lcall tempo
        mov r0,#0fh          ; activation du curseur et clignotement
        lcall send_inst
        lcall tempo
        mov r0,#3fh          ; 2 lignes 8 bits 5x10 dots
        lcall send_inst
        lcall tempo
        mov r0,#53h          ; affiche un s
        lcall write_data
        lcall tempo          ; temporisation de 40 µs nécessaire
        mov r0,#06h          ; décalage du curseur d'une case
        lcall send_inst
        lcall tempo
        mov r0,#61h          ; affichage d'un a
        lcall write_data
        lcall tempo
        mov r0,#06h
        lcall send_inst
        lcall tempo
        mov r0,#6ch          ; affichage d'un l
        lcall write_data
```

```
                lcall tempo
                mov r0,#06h
                lcall send_inst
                lcall tempo
                mov r0,#0afh      ; pointer adresse 1fh de la deuxième ligne
                lcall send_inst
                lcall tempo
                mov r0,#35h               ; affichage d'un 5
                lcall write_data
                lcall tempo
                mov r0,#06h               ; decalage du curseur d'une case
                lcall send_inst
                lcall tempo
                ret
send_inst:
                clr p1.2
                mov p2,r0
                clr p1.0
                setb p1.2
                setb p1.0
                ret
write_data:
                setb p1.2
                mov p2,r0
                clr p1.0
                setb p1.0
                ret
tempo:
                mov r1,#04h
tempo2:
                mov r0,#00h
tempo1:
                nop
                djnz r0,tempo1
                djnz r1,tempo2
                ret
```

Ci-joint une routine pour la division 16 bits :

```
; Division 16 bits : _____________________________________________
; Placer le dividende dans R1 (high-byte) et R0 (low-byte)
; et le diviseur dans R3 (high-byte) et R2 (low-byte).

mov R1,#07h
mov R0,#0D0h
mov R3,#00h
mov R2,#0C8h

div16_16:
  CLR C      ;Clear carry initially
  MOV R4,#00h ;Clear R4 working variable initially
  MOV R5,#00h ;CLear R5 working variable initially
  MOV B,#00h  ;Clear B since B will count the number of left-shifted bits

div1:
```

```
  INC B     ;Increment counter for each left shift
  MOV A,R2  ;Move the current divisor low byte into the accumulator
  RLC A     ;Shift low-byte left, rotate through carry to apply highest bit to high-byte
  MOV R2,A  ;Save the updated divisor low-byte
  MOV A,R3  ;Move the current divisor high byte into the accumulator
  RLC A     ;Shift high-byte left high, rotating in carry from low-byte
  MOV R3,A  ;Save the updated divisor high-byte
  JNC div1  ;Repeat until carry flag is set from high-byte

div2:       ;Shift right the divisor
  MOV A,R3  ;Move high-byte of divisor into accumulator
  RRC A     ;Rotate high-byte of divisor right and into carry
  MOV R3,A  ;Save updated value of high-byte of divisor
  MOV A,R2  ;Move low-byte of divisor into accumulator
  RRC A     ;Rotate low-byte of divisor right, with carry from high-byte
  MOV R2,A  ;Save updated value of low-byte of divisor
  CLR C     ;Clear carry, we don't need it anymore
  MOV 07h,R1 ;Make a safe copy of the dividend high-byte
  MOV 06h,R0 ;Make a safe copy of the dividend low-byte
  MOV A,R0  ;Move low-byte of dividend into accumulator
  SUBB A,R2 ;Dividend - shifted divisor = result bit (no factor, only 0 or 1)
  MOV R0,A  ;Save updated dividend
  MOV A,R1  ;Move high-byte of dividend into accumulator
  SUBB A,R3 ;Subtract high-byte of divisor (all together 16-bit substraction)
  MOV R1,A  ;Save updated high-byte back in high-byte of divisor
  JNC div3  ;If carry flag is NOT set, result is 1
  MOV R1,07h ;Otherwise result is 0, save copy of divisor to undo subtraction
  MOV R0,06h

div3:
  CPL C     ;Invert carry, so it can be directly copied into result
  MOV A,R4
  RLC A     ;Shift carry flag into temporary result
  MOV R4,A
  MOV A,R5
  RLC A
  MOV R5,A
  DJNZ B,div2 ;Now count backwards and repeat until "B" is zero
  MOV R3,05h  ;Move result to R3/R2
  MOV R2,04h  ;Move result to R3/R2

Affiche:
  mov P3,R3
  mov P2,R2
ret
```

Principe de la conversion hexa-BCD :