Cours de microprocesseur

mercredi 3 mars 2010
08:21

Le 68hc12 et son assembleur

68hc12= microcontroleur
--> intègre sur un meme morceau de silicium
-microprocesseur
-mémoire
-Entrées/sorties
112 broches

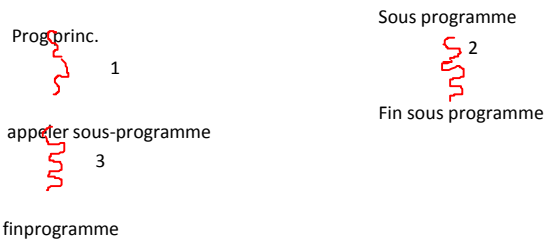
Le 68hc12 intègre :
- 32 Ko de mémoire flash EPROM mémoire programmables électriquement rapide utilisée pour le programme
- 1 Ko de RAM mémoire volatile utilisée pour les données temporaires
- 768 octets de EEPROM utilisée pour les données non volatiles
- CPU= µP
- Port A, port B: port d'entrées/sorties parallèles pour communiquer avec les périph externes. On peut également utiliser ces ports pour dialoguer ou pour gérer de la mémoire externe si la mémoire interne au microcontrôleur n'est pas suffisante.
- ATD converter= convertisseur analogique numérique (analogic to digital)

- Timer: gestion de temps
- SCI: interface de communication en série asynchrone
- SPI: // // synchrone
- PWM: pulse wave modulation
  permet de commander un moteur en modulant sur la largeur d'une impulsion
- BDLC: protocole de communication spécifique
- I/O: entrées/sorties

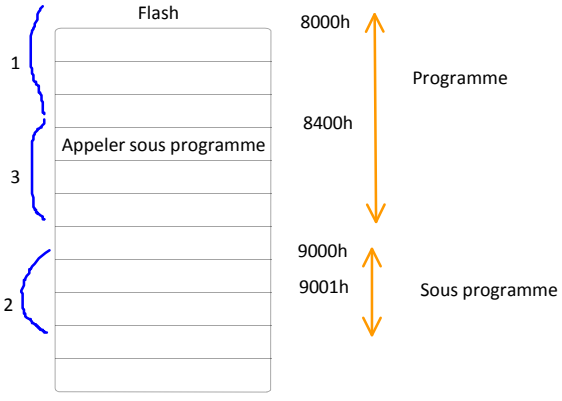
Le plan mémoire

Par défaut:
0000h-01FFh:512 octets de registres
0800h-0BFFh: 1Ko de RAM <-- début de données temp. en 0800h début de la pile en C00h (SP instantanée en C00h) car la pile se remplit dans le sens des adresses décroissantes
0D00h-0FFFh:768 octets d'EEPROM
8000h-FFFFh:52 Ko de Flash <--début du programme en 8000h

Pile: sert à la sauvegarde temporaire des données lors de l'appel d'un sous programme



La pile permet de mémoriser l'adresse de retour à l'éxécution du programme principal Après l'éxécution du sous programme.

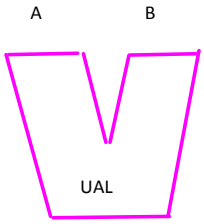


Pile gérée au LIFO last in first out

l'adresse de la dernière donnée mémoire est fournir par SP

1) Mémorisation de l'ad retour
   SP<-- SP-2
   (PCh)=84H est mémorisée dans la case d'adresse SP
   (PCl)=00H // // SP+1
2) Restauration de l'adresse de retour
   PCh <-- la donnée d'adresse SP
   PCl<-- // SP+1
   SP<-- SP+2

Registre du 68hc12



Internes au µP interne au 68hc12
Reg D 16bits

A	B

15 0

La partie haute de D s'appelle A
La partie basse de D s'appelle B

---> utilisé pour les opérations arithmétiques et logiques
Reg X --->16 bits
RegY---> 16 bits
Reg d'index--> utilisé pour mémoriser une valeur qui sert soit de données soit d'adresse

SP: pointeur de pile, fournit l'adresse de la dernière donnée mémorisée dans la pile
PC: pointeur de programme, donne l'adresse de l'instruction suivante
Reg d'état: CCR regroupe des bits (drapeaux) dont la valeur est positionnée en fonction du résultat de l'opération arithmétique ou logique.
H: retenue intermédiaire
N: résultat négatif
Z: résultat nul
V: dépassement
C: retenue
S,I,X: sont positionnés par l'utilisateur
S: autorise l'instruction stop
I: autorise des interruptions sur maquette
X: en réserve

Format d'une instruction
{étiquette} Mnémonique {opérande 1}{opérande 2}{;commentaire}

obligatoire
indique la nature de l'instruction

Les modes d'adressage

Indiquent si l'opérande est une donnée ou une adresse
- Inhérent: - pas d'opérande --> ex: TBA transférer le contenu du reg B vers reg A
  - opérandes= registres du µP --> ex: EXG A B: échange les contenus des reg A et B
- Immédiat: l'opérande est une donnée codée sur 8 ou 16 bits
  ex: LDAA # $ 12 mettre dans le reg A la valeur 12h (# = adressage immédiat, $ = valeur hexadécimale
- Direct: l'opérande présente l'adresse de la donnée. Cette adresse est sur 8 bits entre 0000h et 00FFh
  ex: LDAA mettre dans le reg A la donnée dont l'adresse est 12h
- Etendu : idem adressage direct mais l'opérande représente une adresse sur 16 bits(entre 0000h et FFFFh
  ex: LDAA $1424
- Indexé : avec un offset constant
  ex: LDAA 12,X mettre dans le reg A la donnée donct l'adresse est fournie par le reg X auquel on ajoute 12
- Pré-décrémenté : ex: LDAA 3,-X
  1 décrémenter X de 3
  2 charger le reg A avec la valeur dont l'adresse est fournie par X
- Pré-incrémenté: ex: LDAA,+ X
  1 incrémentation de X
  charger dans reg A la donnée d'adresse fournie par X
- Post-incrémenté: LDAA 2,X+ LDAA 3,X-
  1 chargement du reg A avec la donnée donc l'adresse est fournie par X
  2modification du registre X
- Indexé- Indirect: LDAA [3,X]
  charger dans le reg A la donnée dont l'adresse se trouve à l'adresse X+3
- Relatif: utilisé pour les instruction de saut
  bcl

BRA bcl reprendre l'exécution du programme à partir d'une étiquette qui s'appelle bcl , l'opérande l'instruction représente une adresse

Jeu d'instruction
- 1: instructions de transfert et d'échanges entre registre (cf. tab 5-2)
- 2: instructions de lecture -écriture (cf tab. 5-1)
- 3: instructions arithmétiques
- 4: instructions logiques

5: instructions permettant de réaliser des boucles

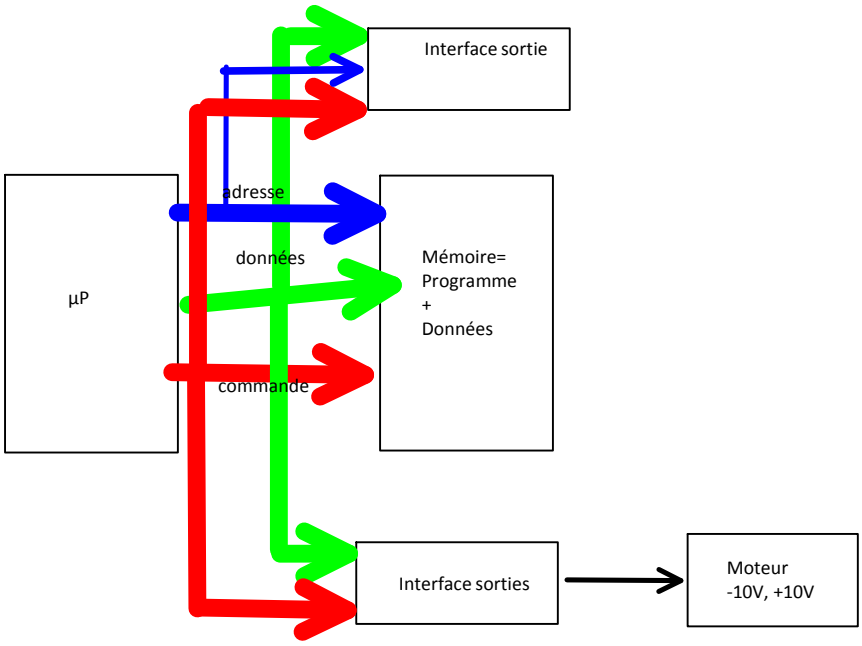
```
ex: pour i=10 to 1 faire          LDAA #10
        s                 encore   s
        s                          s
        s                 tq A>0   s
        s                          s
fin pour                           DBNE A encore   ;décrémente le reg. A de 1 si la valeur du reg  A est différente de 0
                                   s                  on reprend à encore sinon on passe à l'instruction suivante
                                   s
                          si A=0   s
                                   s
```

Vecteur reset: à chaque fois que l'on met le µcontrôleur sous tension , il va toujours lire le contenu d'un même emplacement mémoire, celui d'adresse FFFEh, FFFFh.
On doit mettre dans cet emplacement mémoire l'adresse du programme à exécuter

Les circuits d'entrées/sorties (E/S)

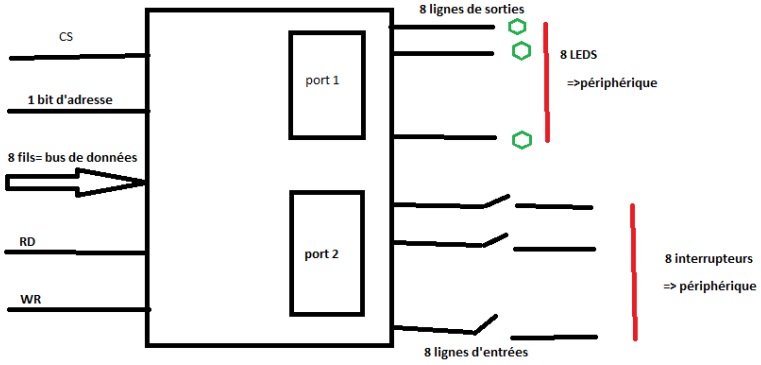
I) Introduction

Le microprocesseur avec l'extérieur via des interfaces qui adaptent les signaux (0V,5V) utilisés par le µP à ceux compris par les périphériques.



Le dialoque µP-interface s'effectue via les bus d'adresse , de données et de commande (de la même façon qu'avec une mémoire)
Le dialogue interface-périphérique s'effectue via des lignes d'E/S
Périph vers interface = entrée
Périph vers périph= sortie
Les lignes d'E/S sont regroupées dans des ponts

Exemple: interface avec 2 points de 8bits



CS sélectionne l'interface pour l'ensemble des interfaces et des mémoires
1 bit d'adresse pour faire

II) l'adressage des interfaces

2 possibilités
- Adressage cartographique
- // indépendant

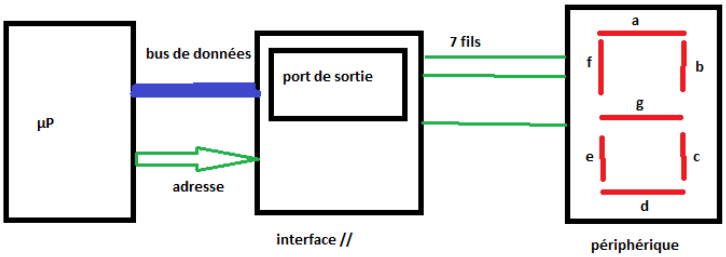
1) l'adressage cartographique

On a attribue une parti de l'espace mémoire disponiqble à des interfaces
Les o,terfaces sont projetées dans le plan mémoire
On accède aux interfaces avec les mêmes instruction que pour la mémoire
ex: LDAA port 1 le reg. A reçoit le contenu du port 2

2) l'adressage indépendant

Les interfaces ont leur propre espace d'adresage
Le µP utilise un signal supplémentaire pour faire la distinction entre les 2 espaces
M/IO= 1 --> accès mém.
M/IO=0 --> accès E/S (M=memory , I/O= input,output
M/IO est généré à partir de l'instruction utilisée

III) Les liaisons parallèles



Si le µP veut aficher 1 il doit activer les segments b et c, le chiffre 2 a b g e d
Chaque fil du port de sortie est relié à un segment de l'afficheur
L'interface dialoque avec le pr"phérique via n fils en //
Chaque bit d'une même donnée est déposé sur un fil

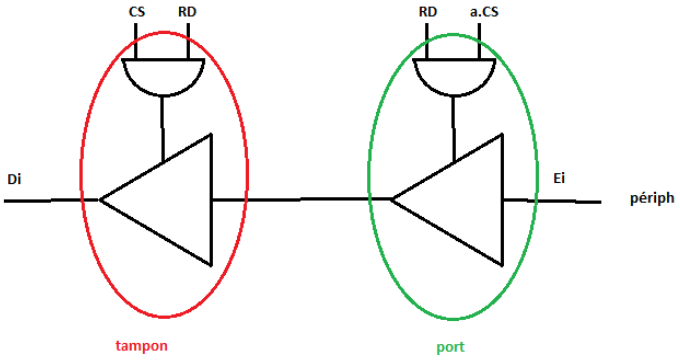
Exemple de périphérique fonctionnant en //

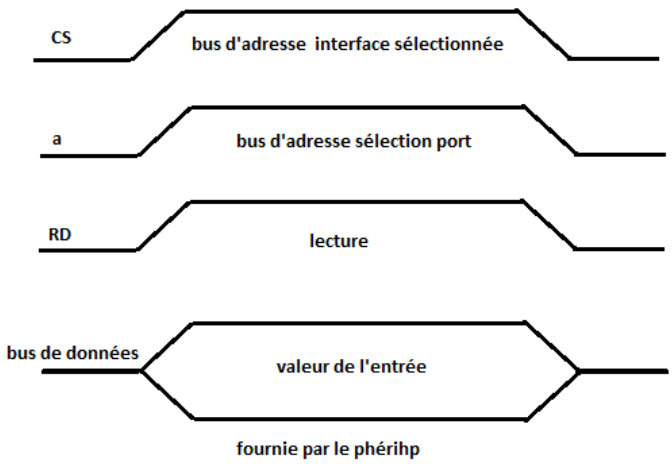
- Imprimante: affichage de caractère A code ascii (60_h : 0110 0000 => n fils en //)
- Commande d'un bras de robot:
  Rotation selon 3 axes
  + translation selon trois axes ==> 6 moteurs à commander en //

1) Architecture interne d'un circuit d'E/S //
   Cf schéma "circuit E/S // p.14 du poly microprocesseur et microcontrôleurs.

CS: sélectionne l'interface parmi les autres et parmi les mémoires
a:sélectionne port A ou port B
RD=1 si le µP veut lire une donnée qui vient d'un périphérique
WR=1 ---------- écrire une donnée vers un périphérique

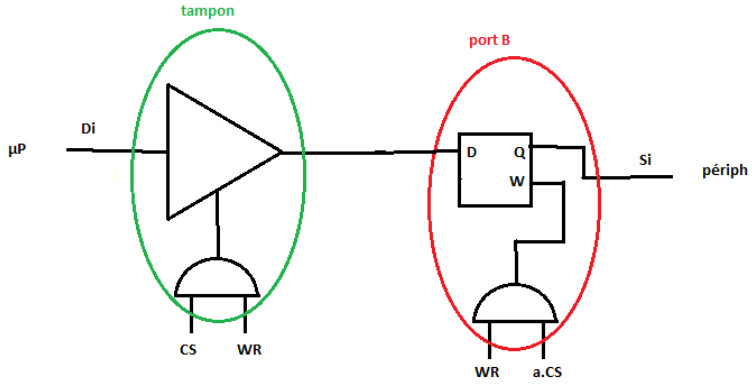
Mode entrée: le µP lit une donnée d'un périph (CS= celui de l'interface
a= sélection du bon port
RD=1)

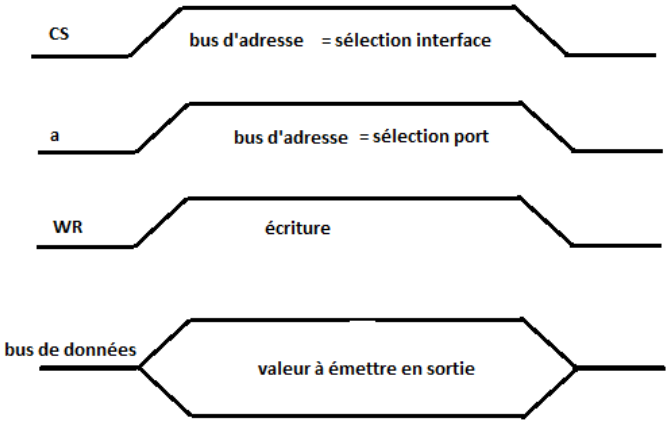




Di => fil i du bus de donnée du tampon
Ei=> fil i reliant le port A et le périph

Mode sortie: le µP envoie une donnée vers un périphérique (CS= celui de l'interface
a= celui du port de sortie
WR=1)

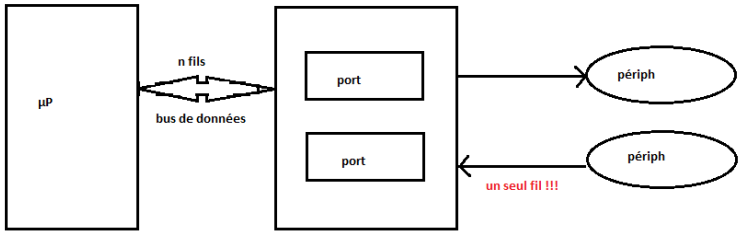




Logique de commande: génère des signaux de commande des portes 3 états et des bascules
Registre de commande: accessible en écriture permet à l'utilisateur de définir sir les lignes de dialogue avec le périphérique sont des entrées ou des sorties (ex: DDRA, DDRB pour le 68hc12)
Registre d'état: accessible en lecture . Donne des informations sur l'état de la communication (donnée émise, donnée reçue, périphérique prêt….)

IV) Les liaisons séries

On dialogue avec le périph via 1 seul fil

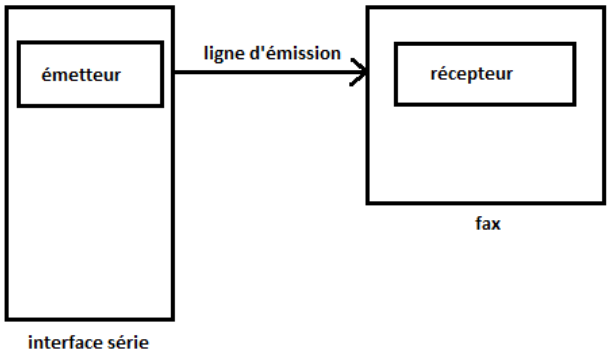


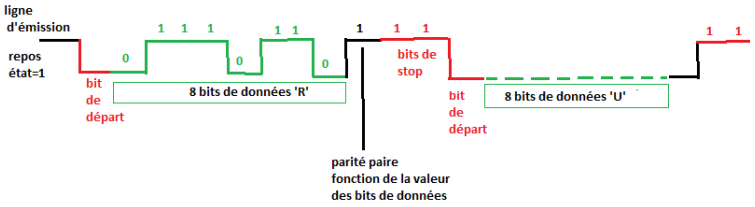
Les n bits qui forment un même caractère sont envoyés les uns à la suite des autres
Ex: 'A': code 60_h =>0110 0000
Ex de périphérique série: souris, fax, modem, réseau

2 modes de communication possible:
- Asynchrone
- Synchrone

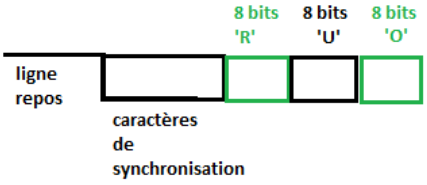
Mode asynchrone:
'BONJOUR' : 8 bits pour 'R': 0110 1110
8 bits pour 'U': 1001 0001
8 bits pour 'O': .....

Chaque caractère émis est formé par :
- 1 bit de départ qui permet de prévenir le récepteur de l'arrivée d'un nouveau caractère
- 8 bits qui correspondent au code du caractère= bits de données
- Éventuellement 1 bit de parité (paire, impaire)
- 1 ou 2 bit(s) de stop.



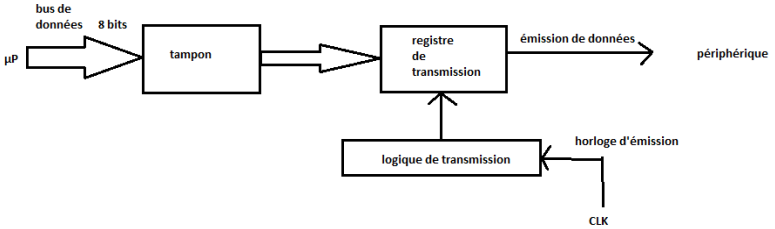


Mode synchrone:
Les données forment 1 suite continue de bits sans qu'il soit possible d'identifier les limites de chaque caractère dans ce flot. Il existe juse quelques bits de synchronisation au départ pour synchroniser le récepteur sur l'émetteur. Avantage: + rapide inconvénient erreur si décalage dans la lecture des bits.

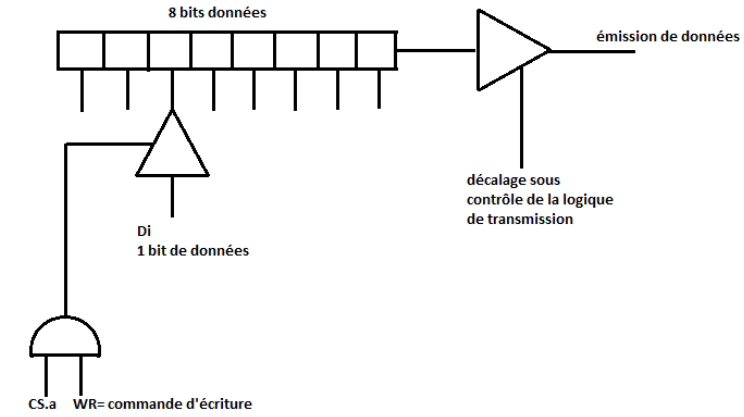


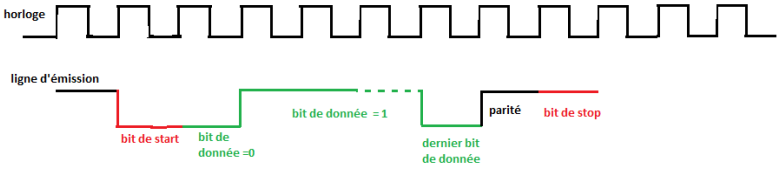
1) Architecture interne d'une interface série

Partie émission: le µP veut émettre une donnée sous forme série à un périph



Reg. De transmission = registre à décalage chargé en // via le bus de données, décalé sous le contrôle de la logique de transmission
Les bits sont émis sur la ligne émission de données à une fréquence fixée par l'horloge d'émission

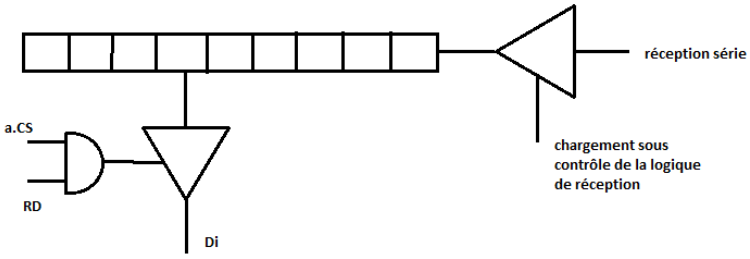




On peut modifier la vitesse de transmission en envoyant les bits toutes les 1 2 4 8 16 périodes d'horloge.

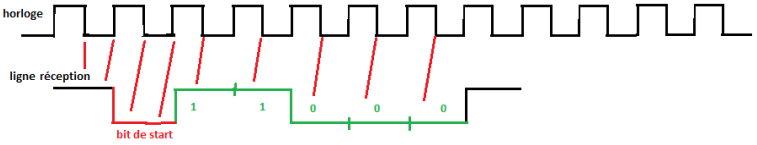
Réception d'un caractère:

La donnée en provenance du périphérique est reçue sous forme série dans le registre de réception , sous contrôle de la logique de réception avant d'être fournie au µP sous forme // via le tampon et bus de données.



registre
CS: sélection de
l'interface
Di: bus de données

l'horloge de réception regarde à chaque demi-période l'état de la ligne réception de données. Dès que la ligne réception de données passe à 0, un début de communication est supposé, l'horloge vérifie qu'il s'agit bien d'un bit start en vérifiant que la ligne réception de données est toujours à 0 après une demi-période.



On récupère la valeur des autres bits par lecture de la ligne réception à chaque période d'horloge.

Le registre d'état

Donne des informations sur l'état de la communication. On sait ainsi si ,

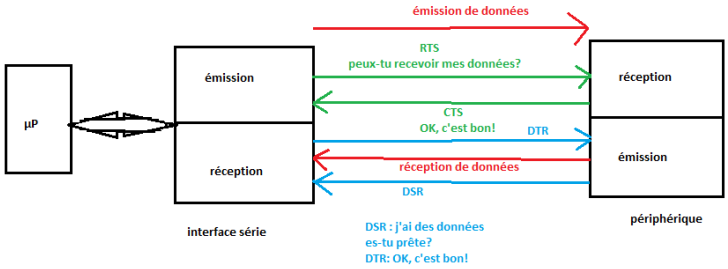
- Le registre de réception est plein
- Le registre de transmission est vide
- s'il y a une erreur de parité (s'il existe une différence entre le bit de parité calculé par le récepteur à partir des bits de données reçues et bits de parité émis)
- s'il y a une erreur de format (la ligne n'est pas à 1 au moment où le récepteur devrait lire les bits de stop)

Le registre de commande:

Permet de définir le protocole de communication (vitesse, nb de bits de données, parité…)

Les signaux de commande du modem:

Permettent d'établir un code de poignée de main (hand-shaking)



1)RTS
2)CTS
3)émission de données

1)DSR
2)DTR
3)réception de données

V) Les techniques de gestion des E/S

A chaque E/S correspond 1 ou plusieurs tâches à effectuer:
Exemple d'un PC: clavier, souris, moniteur, Modem, imprimante, HP, micro…
A chaque tâche correspond un sous-programme. Le µP ne peut excuter qu'un sous-programme à la fois. c'est au programmeur de définir les instants d'éxécution de chaque sous-programme= Ordonnancement.
Il existe deux grandes techniques pour gérer les E/S:

- La scrutation
- Les interruptions

1) La scrutation

Le µP interroge les interfaces à tour de rôle pour savoir s'il doit faire quelque chose pour elles.
Le µP sait si l'interface demande quelque chose en lisant son registre d'état.
Ex: l'interface série indique qu'elle est prête à émettre ou qu'une donnée reçue est disponible en positionnant un bit d'état

Boucle de scrutation:
Tq non fin
    Lire reg. état interface 1
        Si bit x=1 alors traitement x
        Si bit y=1 alors traitement y
            .
            .
            .
    Lire reg état interface 2
        Si bit w=1 alors traitement w
        Si bit z=1 alors traitement z
            .
            .
            .
    .
    .
    .
Fin tq

Avantages:

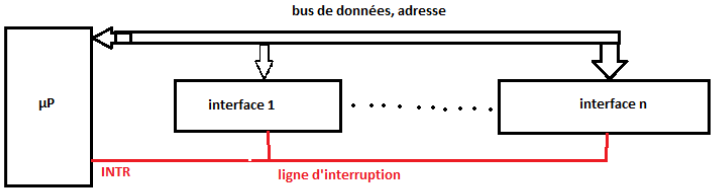
- Pas de matériel supplémentaire
- La scrutation est synchrone avec l'exécution du programme. On sait exactement combien de temps on mettra pour répondre à un é vènement.

Inconvénients:

- Le temps de réponse à un évènement peut être long
- On perd du temps à interroger des dispositifs qui ne demandent rien

2) La technique des interruptions

Ce sont les interfaces qui d'elles même indiquent au µP qu'elles ont besoin qu'on s'occupe d'elle par une ligne de communication spécifique appelée interruption

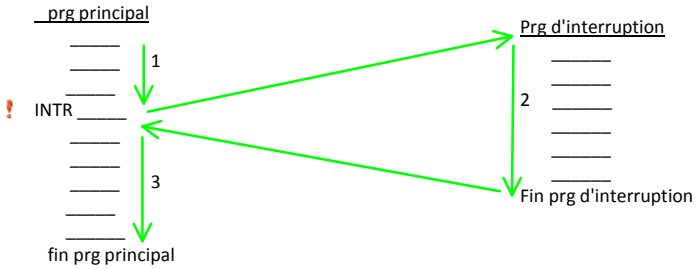


Le traitement des interruptions
A chaque instruction le µP regarde l'état de la ligne INTR
Si la ligne INTR est activée:

- Le µP termine l'instruction en cours
- Il sauvegarde dans la pile l'adresse de l'instruction suivante plus le contenu d'autres registres

- Il exécute le sous programme qui correspond à l'interruption
- Il recharge de la pile l'adresse de reprise du programme principal.

Ex:



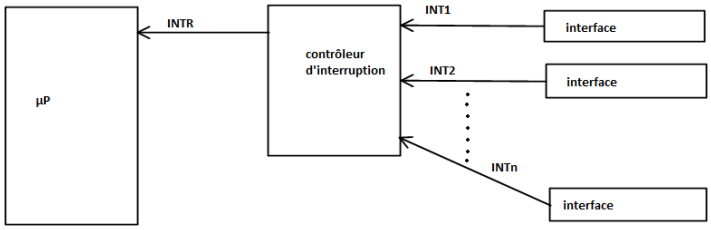
Identification de l'origine de l'interruption

- Identification logicielle

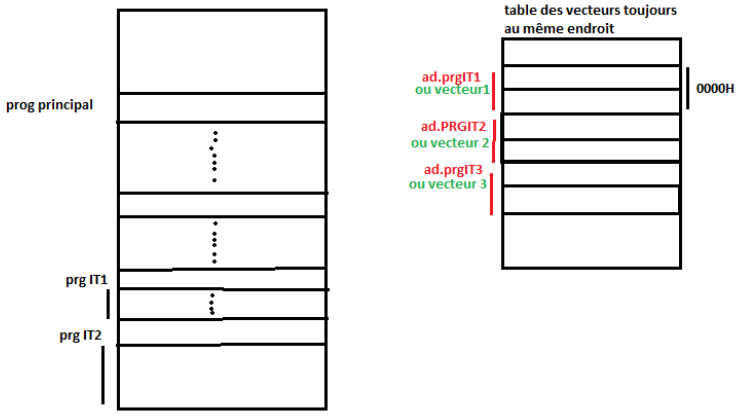
À chaque fois que le µP reçoit un signal d'interruption , il lit les reg d'état des interfaces pour savoir laquelle a généré l'interruption.

- Identification matérielle

On utilise un composant matériel supplémentaire capable de recevoir plusieurs demandes d'interruption, d'en déterminer l'origine et de fournir l'adresse du programme d'interruption à exécuter au µP.



Il faut écrire autant de programmes d'interruption qu'il y a d'interruptions possibles. Les adresses de chaque programme d'interruptioin sont rangées à un endroit précis de la mémoire table des vecteurs.



Le contrôleur d'interruption regarde en permanence quelles sont les lignes d'interruption qui passent à 1. dès qu'il en voit une passer à 1, il envoie le signal INTR au µP et dépose sur le bus de données le numéro de vecteur correspondant.
Le µP n'a plus qu'à lire le contenu de ce vecteur pour avoir l'adresse du programme d'interruption à exécuter.

Priorité des interruptions

La priorité des interruptions détermine l'interruption à traiter en premier lorsqu'il y a plusieurs interruptions simultanées.
Elle peut-être définie par le programmeur.

Masquage des interruptions

Il est posssible de ne pas prendre en compte certaines interruptions dans certaines parties du programme par l'utilisation d'instruction ou de mots de commande particuliers.
STI: autorise les interruptions
CLI: masque les interruptions