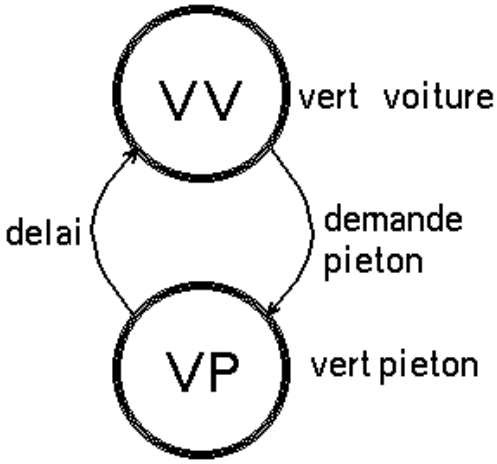
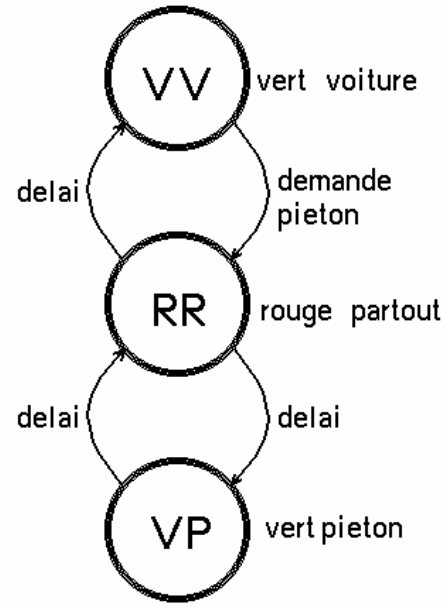
Ex 4.1 Feux de circulation

A)
Un feu pour piéton sur une route droite peut être vu comme une machine à 2 états (au total 4 feux a gérer).
Dans l'état de repos les voitures passent.
Si un piéton demande à passer, après un délai, le feu voiture passera au rouge et le feu piéton passera au vert, puis après on reviendra à l'état de repos.



<u>Contrainte</u> :
5s pour l'état vert pour les piétons – le délai de transition sera 3s

B)
Pour être un peu plus précis, si une personne fait une demande de passer, après un délai les deux feux passeront au rouge.



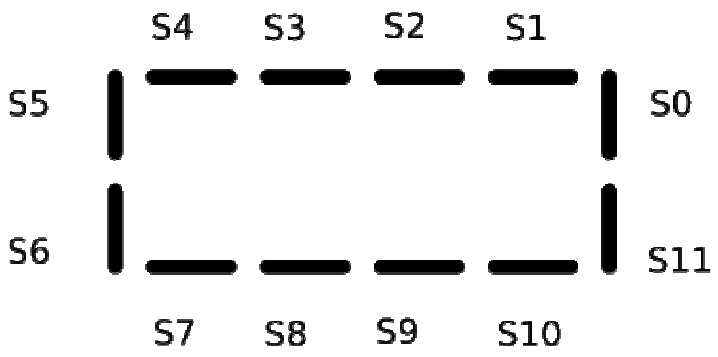
<u>Contrainte</u> :
+ 2s pour les deux feux au rouge

C)
Pour finir, imaginer un système avec les 3 états du feu (vert – orange – rouge) que ce soit pour les véhicules ou pour les piétons.

<u>Contrainte</u> :
+ 1s pour les led orange

Ex 4.2 Cadre lumineux d'une enseigne

Une enseigne publicitaire est entourée d'un cadre réalisé avec des segments lumineux, selon le croquis suivant :



On cherche à donner une impression de rotation, en allumant successivement les groupes de segments suivants :

- S0, S1, S3, S4, S6, S7, etc (011011011011)
- S1, S2, S4, S5, S7, S8, etc (110110110110)
- S0, S2, S3, S5, S7, S8, etc (101101101101)

On demande de développer les fonctions :
`AfficheCadre(unsigned int segments)` qui affiche les segments S0 à S11 selon les bits 0 à 11 du paramètre "segments" e
`AttenteMs(int ms)`.
`Main(int ms)`.qui fait "tourner" le cadre dix fois dans un sens, puis dix fois dans l'autre sens.

Ex 4.3 Poussoir et LED sur une même patte

On souhaite utiliser une seule patte pour brancher un poussoir et une LED.
Le programme devra inverser l'état de la LED à chaque pression sur le poussoir.

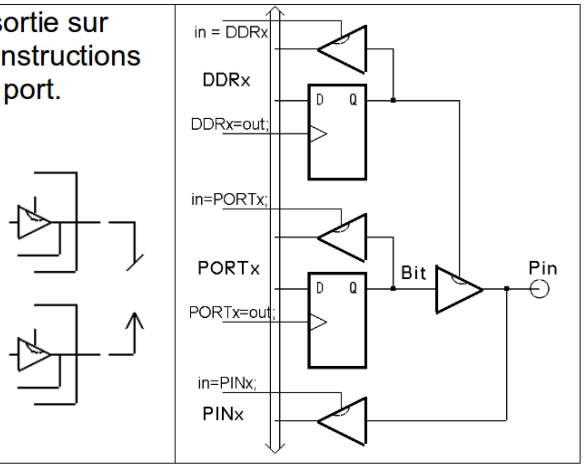
Le programme est plus compliqué qu'il n'y paraît !
Commencez par trouver un schéma utilisable (il y en a deux possibles).
Ne le câblez pas avant l'avoir lu le corrigé : ce serait dommage de "griller" votre microcontrôleur...
Ecrivez ensuite un programme qui lit le bouton et qui fait changer l'état d'une autre LED. Utilisez une procédure PoussON ();
Ajoutez alors le fait que la lecture du poussoir ne modifie pas l'état de la LED (c'est la partie délicate du programme).

Piège : une lecture de l'entrée juste après avoir passé la patte en mode de lecture sera fausse ! Ajoutez simplement une instruction (par exemple mettez une seconde fois la patte en lecture). Ensuite, ce sera facile d'écrire une procédure ToggleLed () qui fait changer l'état de la LED.

Ex 4.4 Testeur de courts-circuits

On se souvient du modèle d'une pin d'entrée-sortie sur AVR ou MSP (leçon 3.2). pour chaque port, 5 instructions permettent de configurer, ecrire, vérifier, lire le port.

On veut tester s'il y a un court-circuit sur une sortie avant de lancer l'application. L'idées est pour chaque port, d'activer toutes les sorties et relire. S'il y a différence on met le port en entrée et signale l'erreur. Le court-circuit dure le temps de quelques instructions, beaucoup moins que ce qu'il peut supporter (quelques dixièmes de seconde)



On écrira un premier programme de test pour le portC, en gardant la fonctionnalité des leds (et des poussoirs, même s'ils ne servent à rien dans cet exercice). Le port C n'a que 6 bits connectés, il faudra masquer toutes les lectures.
Le programme tourne en boucle pour surveiller s'il y a court-circuit. Si oui, le port est mis en entrée et une led est allumée pendant une seconde avant de recommencer le test. Avec un fil vers le + ou vers le moins, on peut alors jouer à faire des courts-circuits sur les pins du port.

Ensuite, on transformera ce test pour en faire une fonction destinée à être appelée au début du setup et qui rend une valeur booléenne, 0 si court-circuit, 1 si OK.

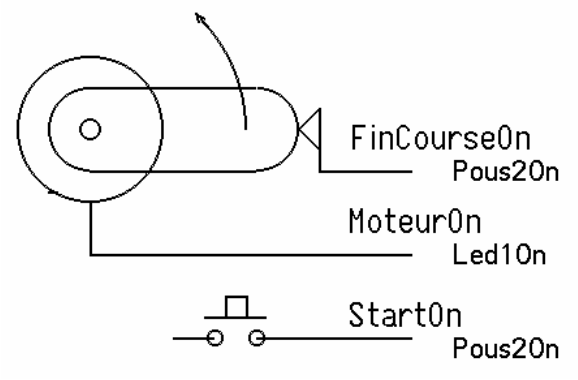
Ex 4.5 Détection de sortie par l'entrée d'un super-marché

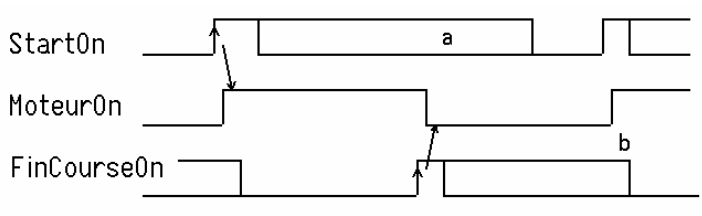
Afin de lutter contre le vol à l'étallage, la direction d'un super-marché souhaite contrôler que les clients n'utilisent pas l'entrée pour sortir du magasin. Deux barrières lumineuses vont être installées au niveau de l'entrée, à la même hauteur, distantes l'une de l'autre de 5 cm. Lorsqu'un client entre, la première barrière est coupée, puis la seconde. Si un client passe dans l'autre sens, ce sera le contraire. Un microcontrôleur reçoit les signaux de ces deux capteurs.

Ecrivez le programme pour commander le contact d'un avertisseur sonore durant 5 secondes lorsqu'un client sort du magasin par l'entrée.

Ex 4.6 Automate simple

On a construit une machine avec un moteur et un contact qui se ferme à chaque tour. En pressant sur un bouton on veut que la machine fasse un tour et s'arrête. Pour tester le programme, on utilise Pous1 pour démarrer, Led1 pour dire que le moteur tourne et Pous2 comme fin de course
Le diagramme des temps montre les états possibles. Il faut tenir compte du fait que le signal Start peut durer plus longtemps que l'action, et le fin de course peut encore être activé après l'arrêt du moteur. Le programme est plus simple si on enlève ces conditions, mais ce n'est que de 2 instructions!





Ex 4.7 Commande d'un monte-charge

La cabine d'un monte charge entre deux étages est actionné par un moteur, dont la commande dispose des entrées Monte et Descend.
Deux interrupteurs de fin de course FinHaut et FinBas détecte les positions des étages.
Deux poussoirs AppelHaut et AppelBas permettent de faire monter et descendre la cabine.

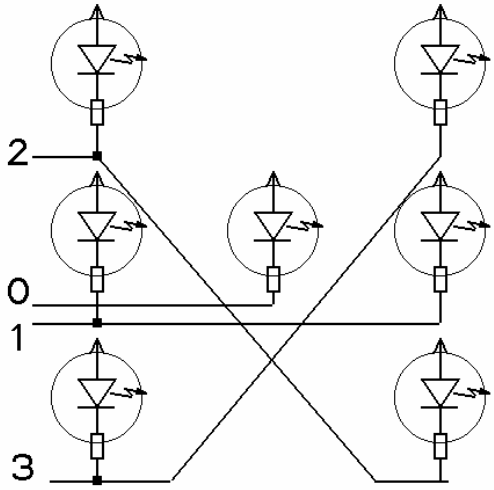
Le programme qui commande le monte charge est une machine d'état.
Voici une proposition de marche à suivre pour faciliter le dessin du graphe d'état :
- établissez la liste des entrées et des sorties
- définissez des états pour le système
- notez les valeurs de sortie pour chaque état
- repérez les transitions entre les états
- notez les conditions associées à ces transitions.

Une fois le graphe d'état établi, écrivez le programme par étapes successives, en suivant les information du graphe d'état :
- définissez les états (par un enum)
- associez les valeurs des sorties
- écrivez les transitions

Il reste à s'occuper de l'implémentation matérielle des entrées et des sorties.

Ex 4.8 Dé électonique

Les 7 leds d'un dé sont câblées comme ci-contre pour utiliser un nombre minimum de sorties du microcontrôleur. Ces sorties sont les pins 0 à 4 du PORTC Arduino (pins 14,15,16,17,18) ou pour MSP les pins 0 à 4 de P2. Créer la table correspondant aux 6 états du dé et écrire le programme qui "lance le dé" toutes les secondes. Evidemment, on ne va pas définir les pins indépendamment. Dans le setup on dit que les bits 0 à 3 sont en sorties, et on copie les configurations de bits correspondant aux faces du dé directement sur le port C.



Ex 4.9 Deux poussoirs pour deux leds en C

On copie les leds sur les poussoirs correspondant. Le programme est trivial et le problème maintenant est de ne pas utiliser d'instructions Arduino, mais d'agir sur le Port D et son registre de direction (MSP430 port P1).
Remarquez l'adjonction d'un b devant les noms de bit pour bien les distinguer des noms des pins. Pour le PORTD seulement ces nombres sont identiques.

Ex 4.10 Moyenne

On a 10 valeurs 8-bits: 23,43,66,51,11,24,92,7,33,15
Il faut calculer la moyenne, qui sera un nombre entier.
On demande encore d'arrondir au plus près (si la partie décimale est supérieure à 0.5, on force, mais attention, le C calcule en nombre entiers).
Inutile de sortie votre calculatrice; le total est 365 et le terminal doit afficher 37.
Attention, l'addition de 10 valeurs 8 bits ne doit pas créer un dépassement de capacité; la somme intermédiaire doit être déclarée en 16 bits.