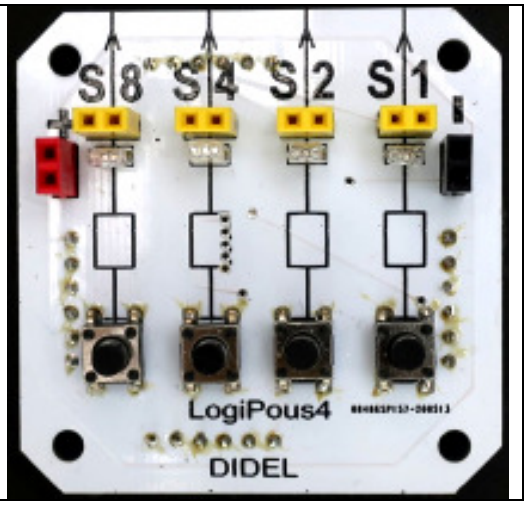
Quelques Logidules intéressants

LogiPous

Traditionellement, on utilise une bascule SR et un interrupteur 1P1T pour supprimer les rebonds de contact. Avec un processeur, on supprime les rebonds avec un temps d'attente de 20ms entre les lectures du poussoir, et on peut mesurer la durée du contact.

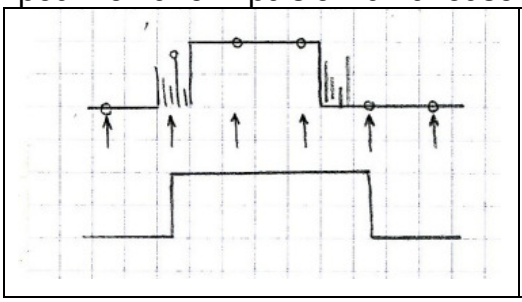
Ce logidule a deux fonctions: poussoir si on pèse rapidement, commutateur si on pèse plus d'une demi-seconde. Il faut presser une 2^{e} fois pour éteindre. Les sorties directes et complémentaires sont disponibles.

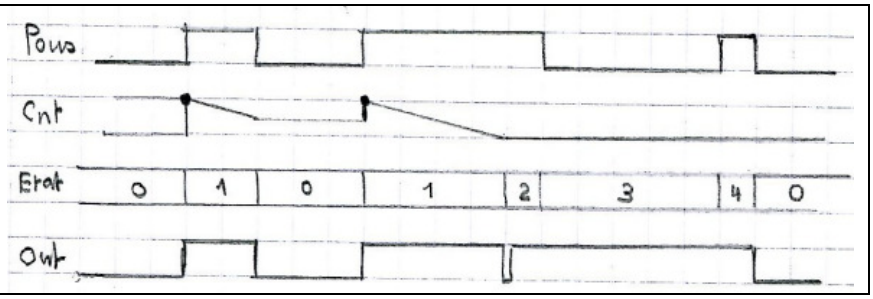


Le processeur suit une machine d'état qui est un bon exercice de compréhension et que l'on peut programmer dans un langage connu des élèves.

Il faut expliquer les rebonds de contact et leur supression par un échantillonage toutes les 20ms. Le diagramme des temps montre les deux situations à décoder.

Un décompteur appelé toutes les 20ms permet de savoir si on a pressé longtemps et qu'il va y avoir mémorisation. Petite astuce, en passant de l'état 1 à l'état 2, un hiatus de l'affichage montre que l'on a changé d'état; l'impulsion ne dure que 20ms et le clignotement est bien visible. L'expérience du préleveur de période (.. page..) permet de mesurer le temps minimum pour voir une impulsion lumuneuse.



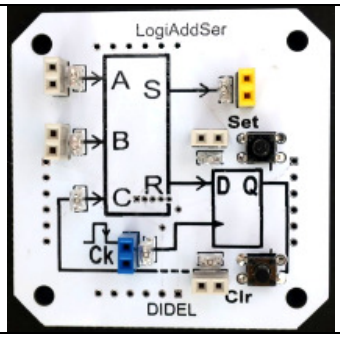


Programme simple pour supprimer les rebonds d'un poussoir

C	Python

LogiAddSer

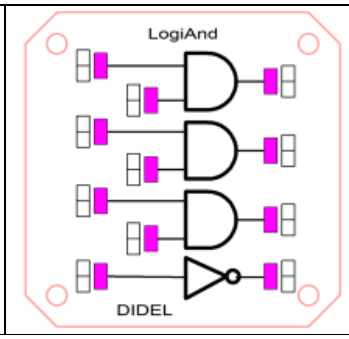
Ce logidule n'utilise pas un additionneur et une bascule, mais un microcontrôleur. Le programme est simple à écrire en C ou Python, avec une partie addition à traiter comme une addition 1 bit, ou en passant par une table. La bascule D a des entrées statiques (Set, Clr) et dynamiques (Ck).
La machine d'état qu'il faut programmer a la structure suivante:

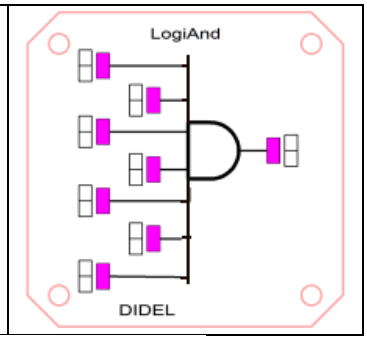


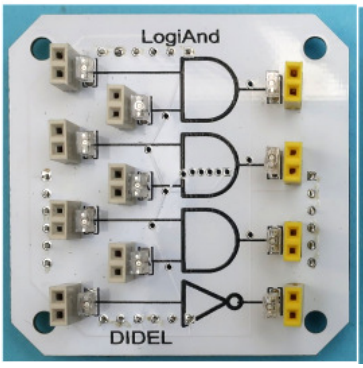
Etat 1 (tant que Ck=0)
    Afficher S et R selon A B C=Q
    Agir sur Q selon Set Clr
    Passe à l'état 2 si Ck=1
Etat 2 (tant que Ck=1)
    Afficher S et R selon A B C=Q
    Agir sur Q selon Set Clr (pas très logique de le faire)
    Passe à l'état 3 si Ck=0
Etat 3
   Copier la valeur de R dans Q (mémoriser l'entrée)
   Retourner à l'état 1

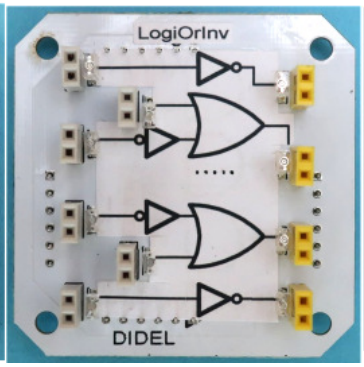
LogiAndInv

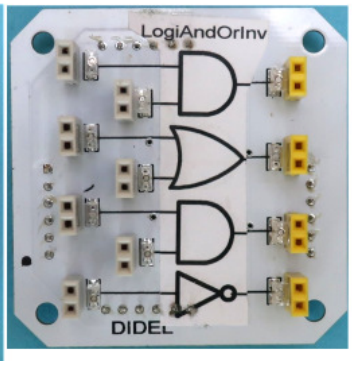
Ce logidule aurait pu se construire avec un circuit 74HC08(And) et un circuit 74HC00 (Nand) ou 2 circuit HC00.
Un processeur Pic 16F630 avec 11 entrées-sorties a été utilisé. On pourrait donc le programmer comme une porte And, Ou, Nand etc à 7 entrées ou toute sorte de combinaison; il suffit de changer le dessin!











Programmer avec quelque if(..) est trivial. Le Pic a été programmé en assembleur et c'est intéressant de comprendre une astuce de programmation qu'il permet. Son architecture date des années 70 et il n'a que 33 instructions et peu de mémoire. C'est terriblement efficace pour les petits programmes (moins de 500 bytes) en temps réel.
Tous les processeurs savent tester un bit dans un port et ont une instruction de saut conditionelle (if sans else). Les Pics ont à la place un "Skip condition". Toutes les instructions sont codées dans un mot de 14 bits, qui est donc assez long pour inclure une adresse 8 bits. Skip peut donc être suivi d'un Jump et implémenter le saut conditionnel.
On a 2 instructions TestSkip,BS et TestSkip,BC pour tester un bit en mémoire ou sur un port (entrée/sorties). Et une instruction SetBit (en écriture simplifiée). La première porte a les bits In1 In2 Out. Ecrivons le instructions

Set	Out
TestSkip,BC	In1
TestSkip,BS	In2

Pour arriver en Clr Out, il faut un 1er test vrai OU un 2e faux. Si on applique DE Morgan, DeMorgan, un premier faux ET un 2e vrai, avec un résultat est inverse. Donc Clr avec un Set préparé à l'avance. Le "glitch" de la sortie qui passe à 1 temporairement n'est pas acceptable (cela pourrait faire avancer un compteur) et il faut préparer l'information dans un registre avant de mettre à jour la sorties.
Tout s'arrange merveilleusement bien avec une seules variable pour nos 4 sorties et on a une boucle de décision de 14 instruction. Il faut initialiser les ports le programme a finalement 32 mots de 14 bits, pas plus!

```
No de ligne -  adresse mémoire -  code 8+6 = 14 bits permutés -  étiquette  - instruction
  49 0000           Debut:
  50 0000  1220             Call     Init
  51 0001           Loop:
  52 0001  2020             Call     Del20ms ;
  53 0002  0C30             Move     #2'001100,W
  54 0003  A400             Move     W,Cra
  55 0004  0718             TestSkip,BC  PortC:#bin11
  56 0005  051D             TestSkip,Bs  PortA:#bin12
  57 0006  2410             Clr      Cra:#bout1
  58 0007  0719             TestSkip,BC  PortC:#bin21
  59 0008  871C             TestSkip,Bs  PortC:#bin22
  60 0009  A410             Clr      Cra:#bout2
  61 000A  871A             TestSkip,BC  PortC:#bin31
  62 000B  071E             TestSkip,Bs  PortC:#bin32
  63 000C  A412             Clr      Cra:#bout3
  64 000D  8519             TestSkip,BC  PortA:#bin4
  65 000E  2412             Clr      Cra:#bout4
  66 000F  2408             Move     Cra,W
  67 0010  8500             Move     W,PortA ; pas de glitch sur les bits
  68 0011  0128             Jump     Loop
```

En C on écrit l'équivalent de `If (In1&In2) {Out=1;} else {Out=0:}`

```
if (PORTC&0b01 & PORTC&0b10) {
  PORTC |= (0b101)
} else {
  PORTC &= ~(0b101);
}
```

La boucle pour le logidule complet prend 32 bytes.

En Arduino, c'est apparemment plus simple if (digitalRead(2) & …) { .. } mais le soft prend 560 bytes (si,si, vérifiez!).

LogiOscillateur

Un interrupteur passe du mode manuel au mode oscillation.
Le potentiomètre sans fin de course (env 30 degrés à une valeur proche de zéro) sélectionne la fréquence avec une variation exponentielle sur 5 décades (de 0.1 Hz à 2 kHz).
Plus de détails dans le document LogiOsc.pdf

LogiCompare

Le circuit a un encodeur rotatif pour choisir la valeur à comparer. Il est cascadable en comparant les poids faibles d'abord. Le premier modules a une entrée EQin (égal) active par défaut et une entrée GTin (greater than) à zéro. Il transmet au modules suivant un signal EQout et un signal GTout. La condition est LT (lowerthan) si les deux signaux à zéro.

LogiMoteur

bien comprendre les capteurs associés. Avec les logidules, on peut câbler des aller-retour sur les butées et les compter pour arrêter le mouvement.
Il manque un compteur décompteur pour tester l'encodeur, mais avec quelques portes on peut afficher le sens. Il manque un convertisseur A/D (4bits ou 8 bits?) pour convertir le potentiomètre, ce qui permettrait avec le comparateur de remplacer les butées.
Ce module n'est en fait pas un logidule; il a sa place piloté par un microcontrôleur et est compatible avec les Micro:bit, Arduino et autres utilisés dans l'enseignement.

jd 200620/200812