

REGISTRE D'EXTENSION SÉRIE

PIERRE-YVES ROCHAT, I

RÉV 2015/09/18

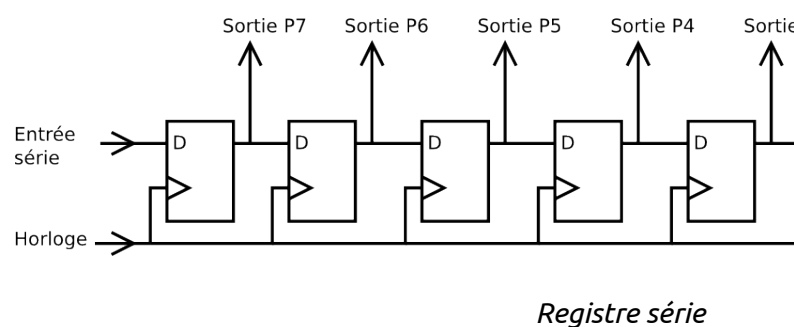
BESOIN DE BROCHES

Les enseignes et surtout les afficheurs à LED nécessitent beaucoup de broches dont on peut commander d'entrées-sorties pour faire face à ce besoin. Plusieurs circuits logiques classiques offrent la fonction *adressables*, qui sont très pratiques. Le circuit 74HC259 est un latch adressable à 8 sorties.

Mais le composant le plus souvent utilisé dans le domaine des afficheurs à LED est le *registre série-p*

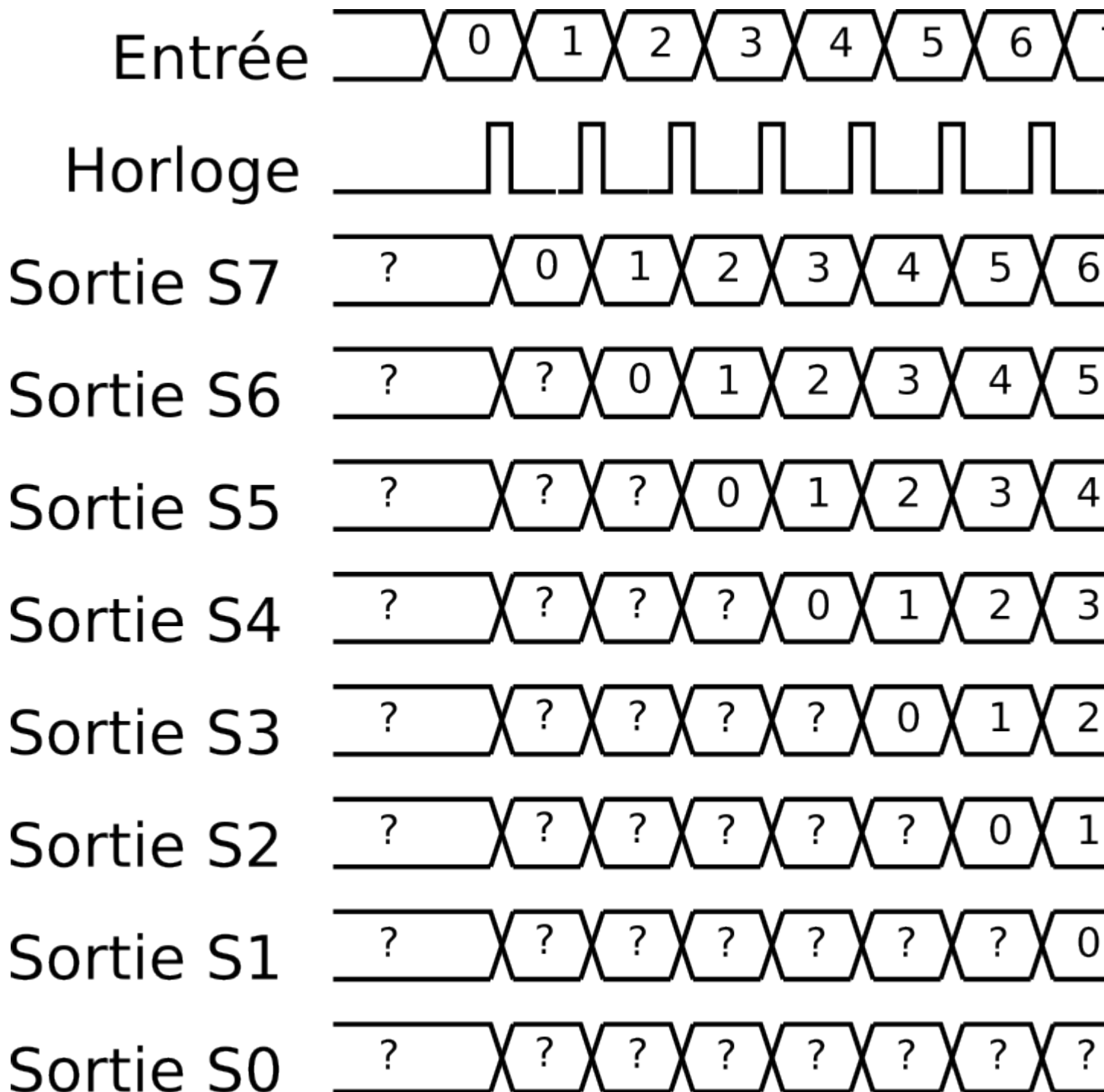
REGISTRE SÉRIE

Les registres à décalage sont réalisés avec des bascules. Voici le schéma d'un registre série 8 bits :



On y trouve 8 bascules D. Les horloges des 8 bascules sont reliées ensemble. L'entrée D de la première bascule est l'entrée série. La sortie de la première bascule est l'entrée D de la deuxième bascule et ainsi de suite. Le système a 8 sorties.

Voici un diagramme des temps qui permet de comprendre comment fonctionne ce registre. Pour ch
flanc montant de l'horloge.

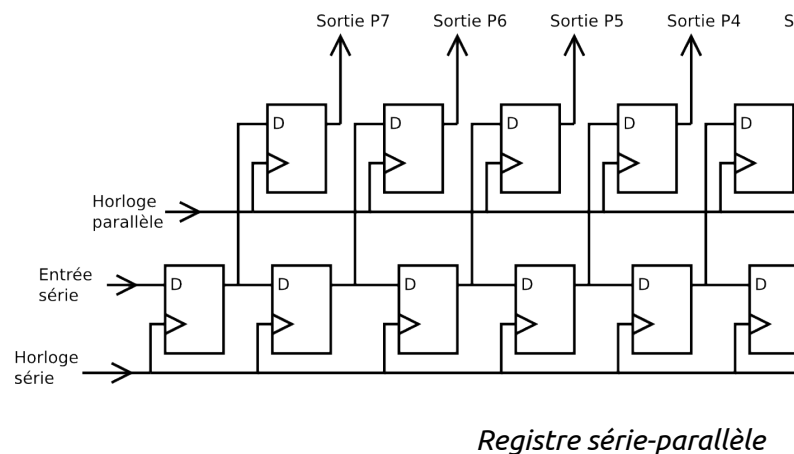


Les valeurs notées 0,1,2..7 correspondent à des valeurs binaires 0 ou 1 placées successivement sur l'entrée à chaque coup d'horloge. Après 8 coups d'horloge, les 8 valeurs envoyées en série vont se retrouver sur les 8 sorties.

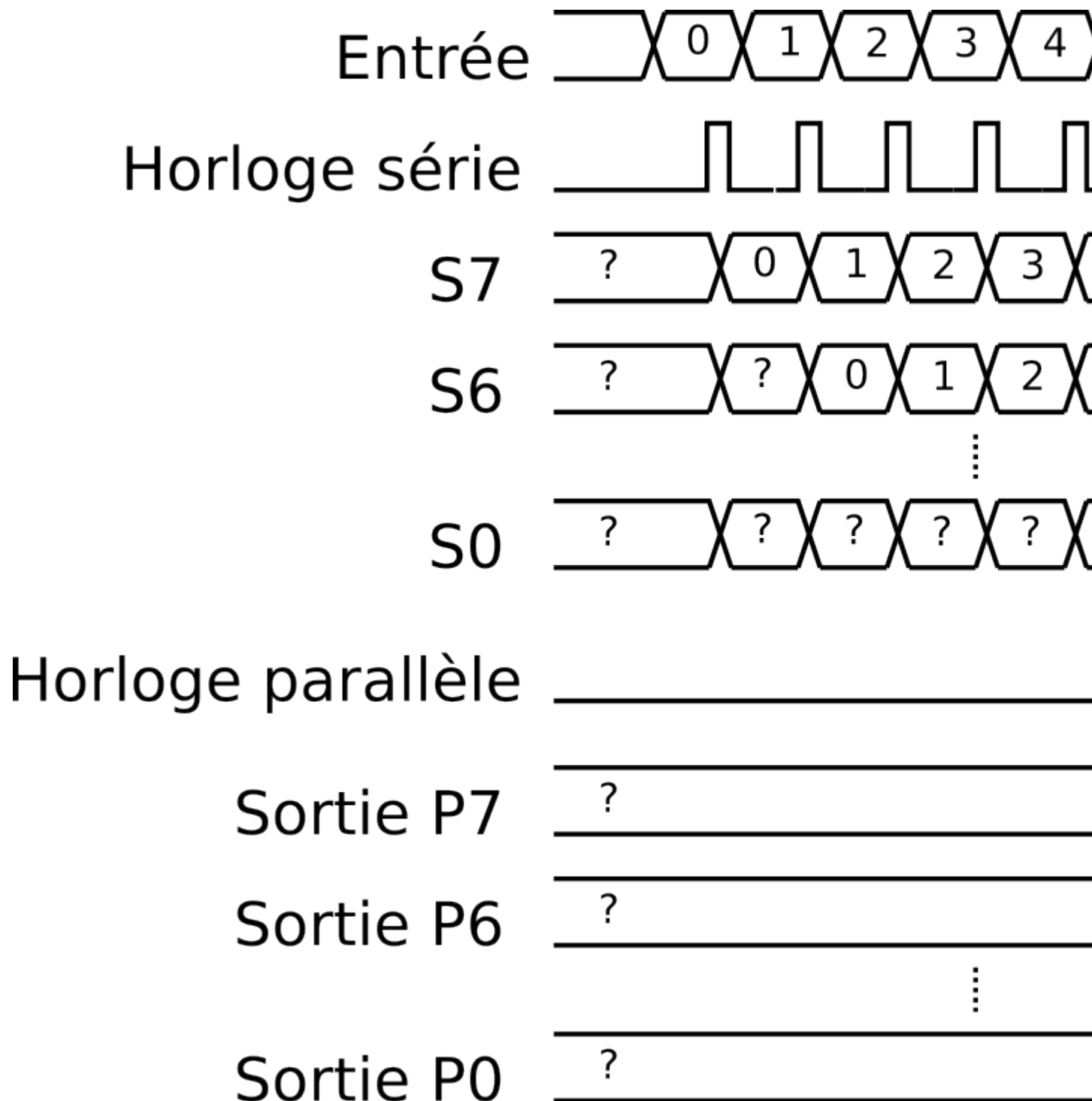
Ce dispositif permet donc de disposer 8 sorties, tout en ne monopolisant que 2 broches du microcontrôleur. Dans certains cas particuliers, ce n'est pas grave si des valeurs non désirées vont apparaître sur les sorties. Dans certains cas particuliers, ce n'est pas grave si le microcontrôleur est très sensible et des artefacts sur les LED deviennent vite gênants.

REGISTRE SÉRIE-PARALLÈLE

Ajoutons 8 bascules D supplémentaires à notre montage. Ces 8 bascules forment cette fois un registre à décalage. Les horloges des 8 bascules sont reliées ensemble.



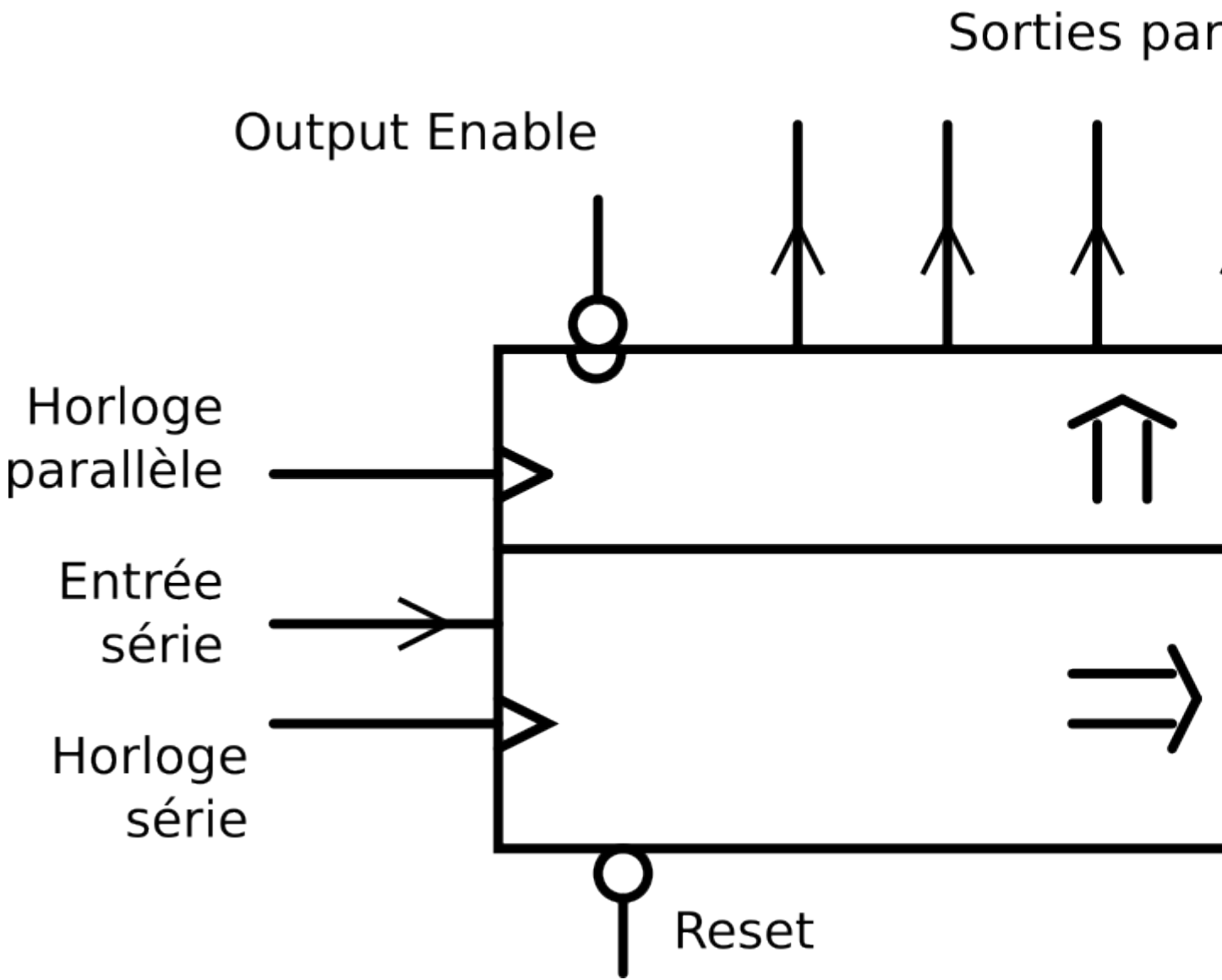
Sur ce diagramme des temps, on voit que les données transmises en série sont ensuite copiées sur les



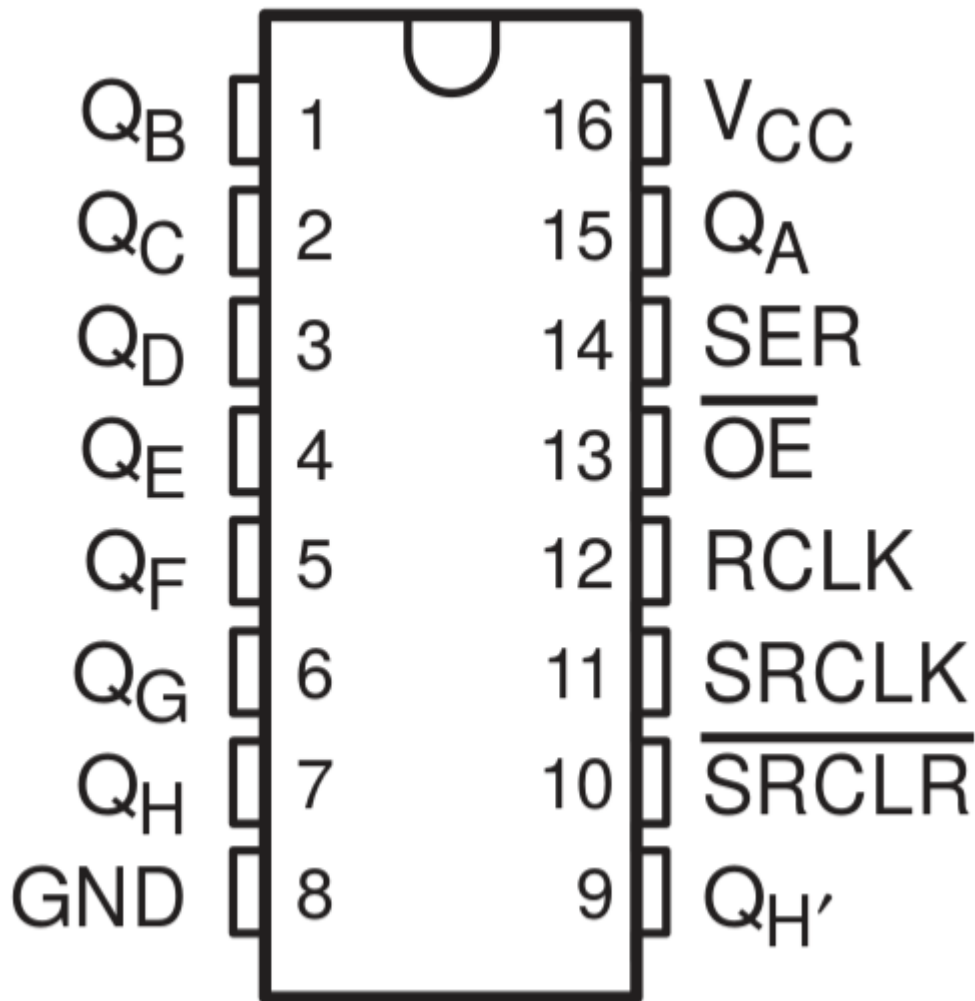
Les 8 valeurs arrivent en même temps sur les sorties du registre parallèle. Les anciennes valeurs sont effacées en même temps. Il n'y a donc pas de risque d'artefacts.

LE CIRCUIT 74HC595

Le circuit 74HC595 est très souvent présent dans des afficheurs à LED. C'est un circuit C-MOS, de la famille 74HC. Les sorties sont à *trois états*, commandées par le signal Output Enable. Une entrée Reset permet de forcer les v

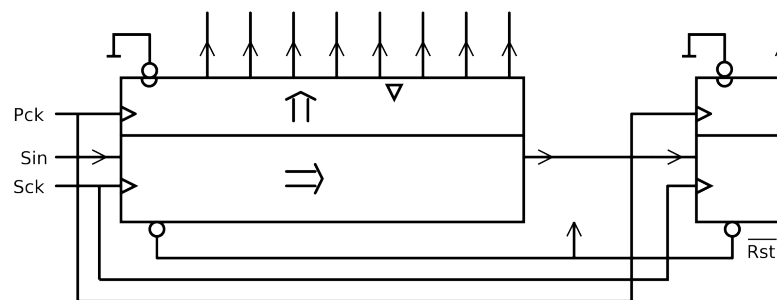


{ width=10cm }



{ width=4cm }

Sa sortie série permet de cascader les circuits, c'est-à-dire les placer les uns à la suite des autres, comme illustré ci-dessous.



Registre série 16 bits utilisant 2 registres 74HC164

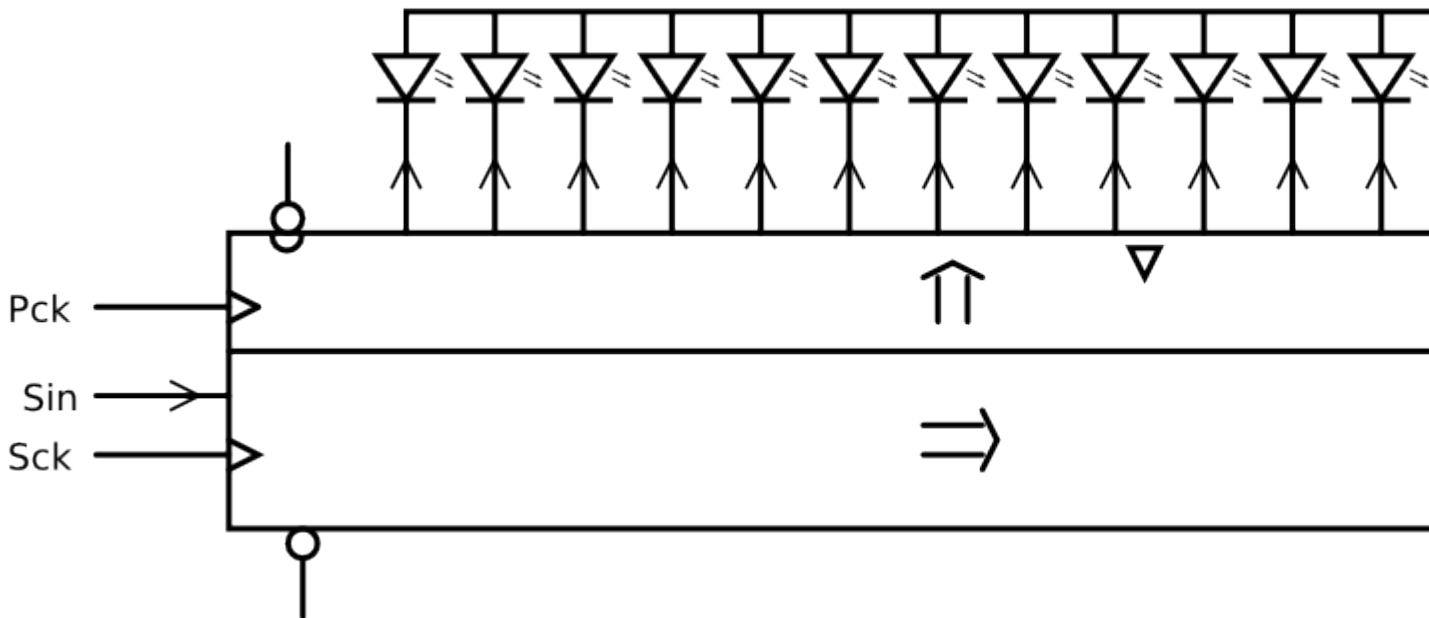
Pour ce faire, les horloges doivent être communes à tous les registres : l'horloge série, notée **Sck**, est l'entrée du deuxième et ainsi de suite. Il est possible de créer de très longs registres. Quelle que soit la longueur, la sortie série permet de cascader les registres.

saires pour le commander. Si le nombre de registre est vraiment important, il sera judicieux de planifier le circuit suivant. Dans beaucoup d'afficheurs à LED, on trouve des 74HC245 qui jouent ce rôle. Il s'agit de pas

REGISTRES À SORTIES À COURANT CONSTANT

Bien entendu, les sorties du 74HC595 sont des sorties C-MOS normales. Pour les utiliser pour commander des LEDs, il faut ajouter une résistance pour limiter le courant.

Il existe plusieurs circuits registres série-parallèles dont les sorties incorporent des sources de courant constant. Cela simplifie le schéma :



La valeur du courant dans toutes les sorties est fixé par une seule résistance, notée R sur le schéma.

Plusieurs fabricants proposent différents modèles de registres série-parallèle avec sources de courant constant, mais qui ont en commun leur brochage. Toshiba propose le TB62701, Texas Instrument le TLC595, et enfin ST propose le TDA1980 dont je n'ai trouvé la documentation... qu'en chinois !

PROGRAMMATION

La procédure pour envoyer 8 bits dans notre registre série parallèle est constituée d'une boucle *for* qui permet de transmettre 8 bits. Le premier bit à transmettre est d'abord placé sur une sortie, puis un flanc montant est produit sur l'horloge. L'horloge est ensuite envoyée sur l'horloge parallèle. Tout à la fin, une impulsion est envoyée sur l'horloge parallèle.

```

1  #define SortieSerieOn P1OUT |= (1<<0)
2  #define SortieSerieOff P1OUT &=~(1<<0)
3
4  #define ClockSerHaut P1OUT |= (1<<1)
5  #define ClockSerBas P1OUT &=~(1<<1)
6
7  #define ClockParHaut P1OUT |= (1<<2)
8  #define ClockParBas P1OUT &=~(1<<2)
9
10 void Envoie8bitsSerie (uint8_t valeur) {
11     uint16_t i;
12     for (i=0; i<8; i++) {
13         if (valeur & (1<<i)) {
14             SortieSerieOn;
15         } else {
16             SortieSerieOff;
17         }
18         ClockSerHaut; ClockSerBas;
19     }
20     ClockParHaut; ClockParBas;
21 }

```

Cette procédure devient souvent la procédure centrale dans un programme qui gère un afficheur à LED. On peut même jusqu'à regarder comment le compilateur l'a traduite en instructions assembleur, pour chercher des optimisations.

Voici par exemple une version plus optimisée de l'envoi des bits :

```

11 ...
12     for (i=0; i<8; i++) {
13         if (valeur & (1<<0)) {
14             SortieSerieOn;
15         } else {
16             SortieSerieOff;
17         }
18         ClockSerHaut; ClockSerBas;
19         valeur = valeur >> 1;
20     }
21 ...

```

Cette version supprime l'opération `valeur & (1<<i)`, qui prend davantage de cycles d'horloge du microcontrôleur.

Ce n'est qu'un petit exemple d'optimisation. Une autre technique sera décrite plus loin dans ce cours. Dans ce cas, ce n'est plus le microcontrôleur qui va effectuer le travail d'envoi des bits, mais un contrôleur dédié.