Enseignes à LED

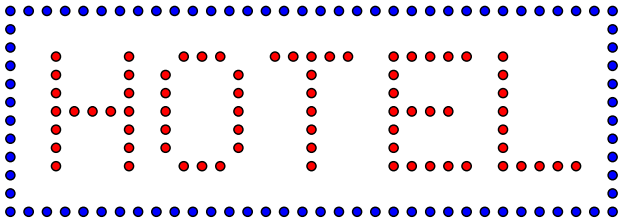
Pierre-Yves Rochat

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Grâce à leur faible coût, leur bon rendement et leur simplicité de mise en oeuvre, les diodes lumineuses (LED) sont utilisées dans de nombreuses applications. Il est par exemple possible de réaliser toute sorte d'enseignes lumineuses, soit des enseignes à motifs fixes, soit des enseignes à matrices de LED, permettant de composer des texte ou même des motifs graphiques.

La disponibilité de LEDs rouges, vertes, bleues, orange et blanches (pour ne citer que les plus courantes) permet de réaliser des enseignes multicolores. Il est même possible d'utiliser des LED multicolores, contenant chacune trois puces générant les couleurs rouge, verte et bleu, permettant de générer toutes les couleurs par mélange additif.

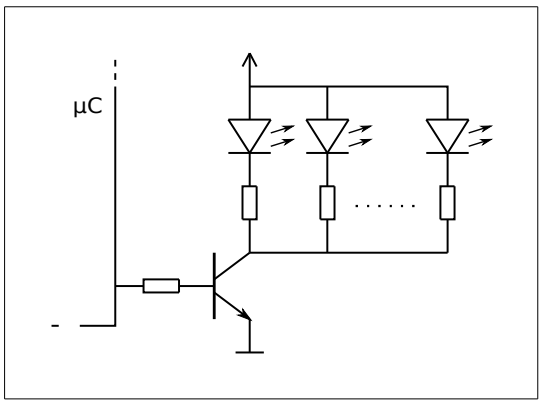
Enseignes à motif fixe



En fixant des LEDs sur des panneaux (par exemple en plexiglas), il est possible de composer des motifs tels que des caractères ou des logos. En commandant les motifs séparément, il est possible de générer des animations qui attirent l'oeil, tels que des clignotements, des chenillards et même des variations continues d'intensité par PWM.

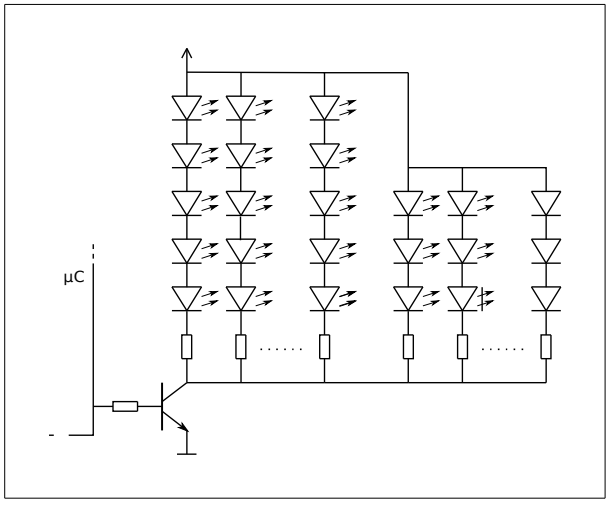
Comment connecter ensemble un grand nombre de LED, pour que les motif formés puissent être commandé par un microcontrôleur ?

Le courant consommé par une LED standard est d'environ 10 mA. Il existe bien entendu aussi des LED beaucoup plus puissantes, plutôt utilisées pour l'éclairage. Les sorties des microcontrôleurs permettent en général de fournir quelques dizaines de mA, suffisant pour une ou deux LED. Pour davantage de LEDs, un transistor sera utilisé (transistor bipolaire ou MOS.



Même si elles sont regroupées en un seul motif, commandé par une sortie d'un microcontrôleur, il est toujours nécessaire de placer une résistance de limitation de courant pour chaque LED. Une seule résistance en série avec plusieurs LEDs reliées en parallèle est fortement déconseillée. Toutes les LED n'étant pas identiques, la luminosité ne sera pas constante.

Par contre il est possible d'augmenter la tension d'alimentation et de placer plusieurs LEDs en série pour une seule résistance de limitation. Le courant est exactement me même dans chaque LED d'une branche ! Il



faut connaître la tension de chaque LED pour choisir un nombre optimum. Les LEDs rouges ont en général une tension d'environ 2 Volt, alors que les vertes ou les bleues ont plutôt 3 Volt. Mais il existe plusieurs technologies pour produire des LEDs et il faut bien se renseigner concernant les LEDs choisies... ou simplement les mesurer !

Avec une tension d'alimentation de 12 Volt, on peut mettre 5 LED rouges ou 3 LED vertes. Notez qu'un seul transistor faible signal, par exemple un BC337, dont le courant maximal est d'environ 500 mA (cette valeur varie selon les fabricants), il est possible de commander environ 250 LEDs rouges avec cette technique, chaque groupe de 5 LEDs en série recevant 10 mA.

Quizz : Avec une alimentation de PC portable, fournissant 16 Volt et un maximum de 4.5 ampères, combien de LED vertes est-il possible de commander ? La tension au borne de chaque LED doit être d'environ 3 Volt, pour un courant de 10 mA.

( ) 45 LEDs ( ) 450 LEDs ( ) 2'250 LEDS ( ) 12'300 LEDs

Réponse : on peut mettre 5 LEDs en série (15 Volt) avec une résistance de limitation. Il est possible de placer 450 groupe de LEDs (450 * 10mA = 4.5A). C'est donc un total de 2'250 LED's qu'il est possible de commander.

Dans ce cas, voici comment calculer la valeur de la résistance : les 5 LEDs devant absorber chacune 3 Volt, il reste 16 – (5*3) = 1 Volt au borne de la résistance. Pour un courant nominal, il faut R = U / I = 1V / 10mA = 100 Ohm.

En pratique, on réalisera un montage de test avec les 5 LED et une résistance, sans oublier le transistor, qui a aussi une chute de tension. On mesurera la tension au borne de la résistance, on calculera le courant qui la traverse (I = U / R). On corrigera ensuite la valeur de la résistance pour se rapprocher du courant souhaité (par exemple 10 mA) et on reprendra le test.

Logiciel de commande d'une enseigne

Une enseigne est un dispositif qui ne dispose que de sorties. Sa programmation correspond donc à une suite d'affectations de sorties et d'attentes : c'est un séquenceur.

Il est important d'utiliser des techniques de programmation qui évitent d'interminables suites de commande, rendant l'écriture et le dépannage des programmes fastidieux. Parfois, il faut aussi prendre soin d'utiliser des techniques économes en place mémoire, pour pouvoir implémenter des animations variées sur de petits processeurs. Pour cela, on peut inventer un langage spécialisé pour décrire les animations, qui sera ensuite interprété par un petit programme en C.

D'autre part, l'utilisation de PWM pour avoir des variations d'intensité permet des produire des enseignes plus attrayantes. Par exemple, on pourra faire « bouger » les lettres successives d'un mot, tout en maintenant le mot toujours lisible en demi-intensité.

Il est aussi intéressant de pouvoir séparer plusieurs parties de l'enseigne et d'appliquer à chaque partie un programme indépendant (multi-tâches).

Un programme pour AVR ATtiny2313, intégrant ces trois techniques, est disponible dans un document séparé.

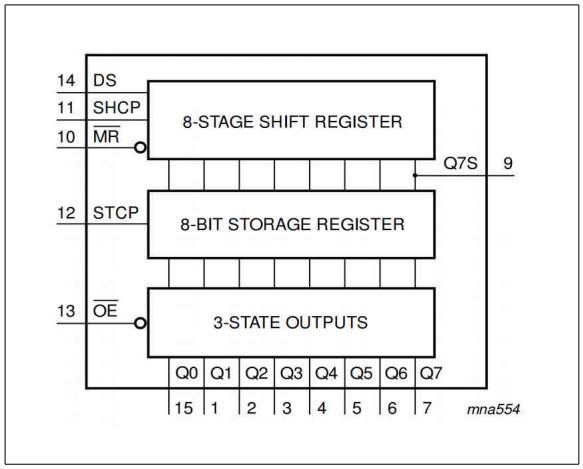
Extensions des entrées-sorties

En augmentant le nombre de motifs sur une enseigne, la richesse des effets visuels

possibles augmente. Par exemple, le mot « Restaurant » peut constituer un seul motif, qu'il est possible d'allumer, d'éteindre, de faire clignoter, etc. Mais si chaque lettre est un motif indépendant, beaucoup d'autres animations qui attirent l'œil sont possibles.

Faut-il alors choisit simplement un microcontrôleur qui offre davantage de pattes ? Ce n'est pas toujours possible, ni pratique, ni simplement avantageux.

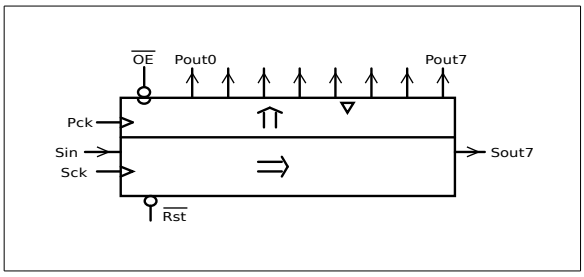
On utilise souvent des circuits intégrés permettant l'extension du nombre de sorties. Il existe des circuits spécialisés, basés sur des standards de communication comme SPI ou I2C, qui seront présentés plus loin dans le cours. Mais on peut aussi utiliser de simples registres, comme par exemple le 74HC595.



Ce circuit contient un registre série 8 bits, dont les sorties sont branchées sur un registre parallèle de 8 bits également. Le registre série a une entrée série. Le registre parallèle a une sortie parallèle. Chaque registre a son horloge (Clock).

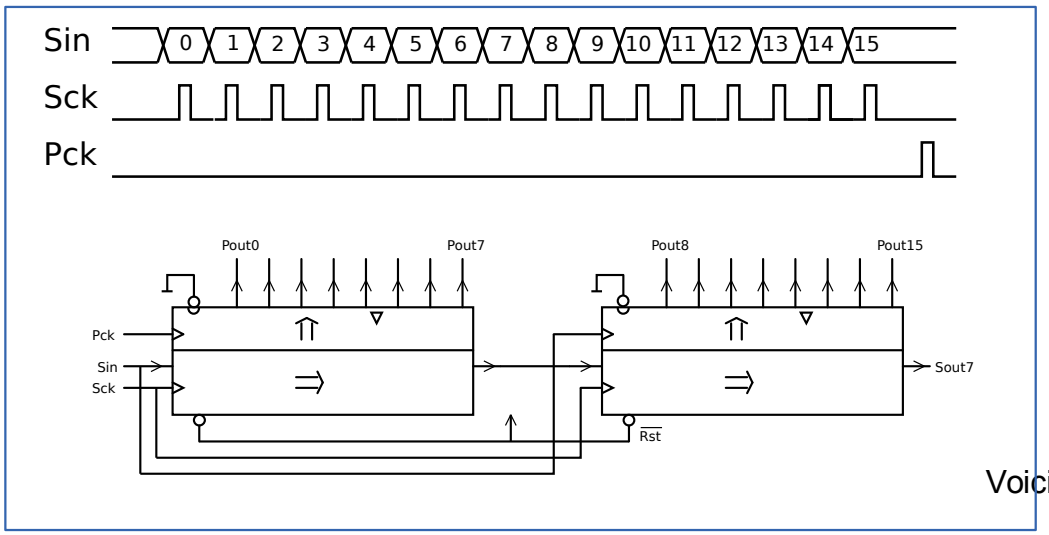
La figure ci-contre montre le symbole donné par un fabricant. On lui préférera un symbole plus compact.

L'horloge série **Sck** permet d'enregistrer l'état de l'entrée série **Sin**, dans la première bascule du registre série et de le décaler. La sortie **Sout7** permet de cascader plusieurs registres. L'entrée **$\overline{\text{Rst}}$** (active à zéro) remet le registre série à zéro.



L'horloge parallèle **Pck** permet de transférer le contenu du registre série dans le registre parallèle, relié aux sorties **Pout0** à **POut7**. Ces sorties sont à trois-états : elles peuvent être désactivées par l'entrée **$\overline{\text{OE}}$** (Output Enable), active à zéro. Sur des enseignes, cette entrée permet de faire une variation globale d'intensité, par PWM.

La figure ci-dessous montre le digramme des temps correspondant à l'envoi de 16 bits sur les sorties. On voit que seuls trois signaux de commande sont nécessaires, même pour un registre très long (en cascadant **Sout7** sur **Sin** du registre suivant et en connectant ensemble les entrées **Sck** de tous les registres, de même pour les **Pck**).



Voici la procédure

pour l'envoi des 16 bits :

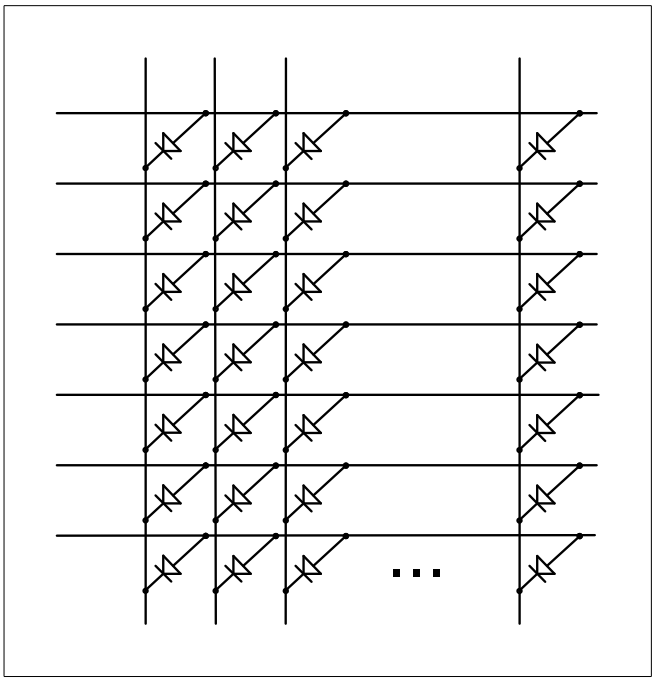
```
void Envoie16 (valeur: integer) {
    int i;
    for (i=0; i<16; i++) {
        if (valeur & 1) { SetSin; }  else {ClearSin; }
        SetSck; ClearSck;
        valeur = valeur >> 1;
    }
    SetPck; ClearPck;
}
```

Matrices de LED

Le principe des afficheurs matrices est de disposer de pixels indépendants sur une géométrie orthogonale, généralement même orthonormée. Autrement dit, on place des LEDs sur une grille régulière. Chaque LED doit pouvoir être commandée séparément.

Brancher chaque LED sur une patte du microcontrôleur est évidemment impossible. Des registres est souvent utilisés.

A titre d'exemple, j'ai réalisé un circuit imprimé comportant 16 fois 16 LEDs, soit 256 au total. Des registres 74HC595 au nombre de 32 commandent chaque LED. Un microcontrôleur MSP430G2302 envoie les signaux aux registres. Il peut être programmé pour générer des animations. On l'utilise ainsi pour de petites enseignes publicitaire de vitrine. Le microcontrôleur peut aussi être programmé pour recevoir des images à partir d'un bus de communication, comportant 4 bits de données et 3 bits de commande. Il est alors possible de fabriquer de grands journaux lumineux, en alignant plusieurs circuit. Des enseignes spécifiques, telles que des croix de pharmacies, ont aussi été réalisée avec cette technique. Un processeur plus puissant (un ARM dans ce cas) est utilisé pour transmettre les images à tous les carrés, chacun possédant un numéro spécifique (ligne, colonne).



Matrices a balayage

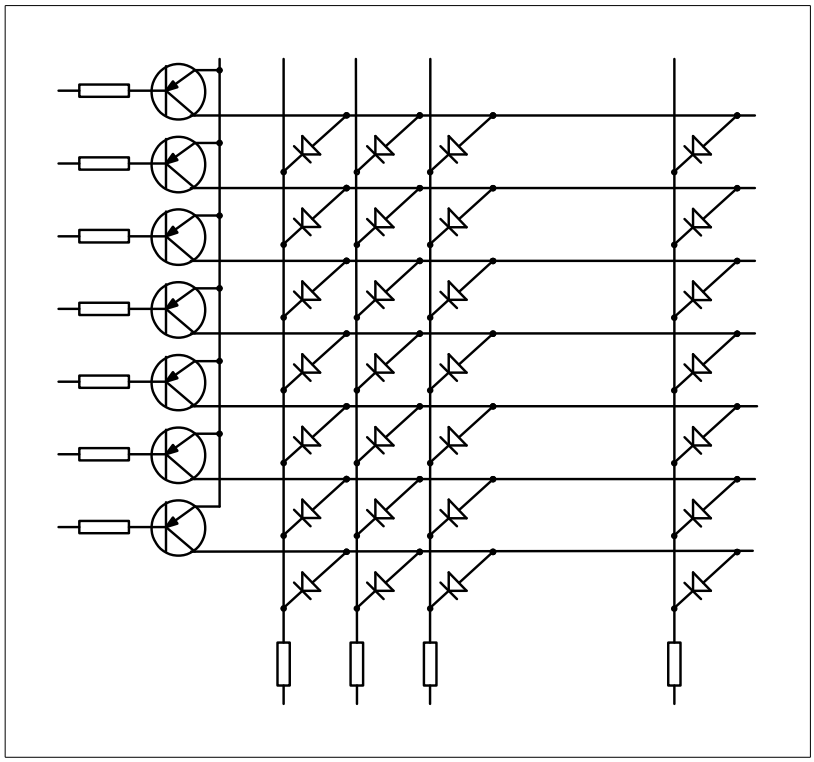
Il existe une technique plus astucieuse pour fabriquer des afficheurs matriciels avec moins de composants électroniques. La figure ci-contre montre le principe : les LEDs sont connectées à toutes les intersections d'un certain nombre de lignes et de colonnes.

Avec cette topologie, il n'est pas possible d'allumer en même temps deux LEDs se trouvant sur des lignes et des colonnes différentes, sans allumer d'autres LEDs parasites. Il nécessaire d'utiliser la technique du balayage, où les lignes sont accédées successivement.

Si ce balayage est suffisamment rapide, l'œil ne perçoit plus l'allumage successif des lignes et voit l'ensemble de l'image formée sur la matrice de LEDs. Une fréquence supérieure à 100 Hz, ou mieux 150 Hz, enlève l'impression de clignotement.

Bien entendu, on perd de la luminosité. Lorsqu'il y a par exemple 7 lignes, une LED ne pourra être allumée dans le meilleur des cas qu'un septième du temps. On pourra toutefois faire passer dans les LED leur courant maximum : c'est le courant qu'il est possible d'atteindre sans abîmer le composant, mais de manière non continue (contrairement au courant nominal, qu'il est possible d'imposer en continu sans risque). Or ce courant maximal est souvent le double du courant nominal, par exemple 20 mA contre 10 mA. On peut donc gagner un peu sur la luminosité perdue par le balayage.

Comme chaque ligne doit être accédée successivement, le courant total maximum dans une ligne sera le courant maximum d'une LED multiplié par le nombre de LEDs. La sortie d'un microcontrôleur ne va donc pas suffire. Il faut amplifier le courant avec un transistor, qui peut être un transistor bipolaire ou MOS. Si les lignes correspondent aux anodes, il faut un transistor PNP (ou un MOS à canal P) : l'émetteur doit être relié au +.



Lorsque du courant passe dans une colonne, ce sera le courant d'une seule LED, vu que les lignes sont accédées successivement. C'est donc au niveau des colonnes qu'il faut mettre la résistance de limitation. En la plaçant sur les lignes, la présence d'un courant variable (selon le nombre de LED allumées dans la ligne) entraînera une chute de tension variable (par la loi d'Ohm) et donc un courant variable dans chaque LED, qui donnerait des différences de luminosité selon les lignes.

Le schéma ci-contre respecte ces principes.

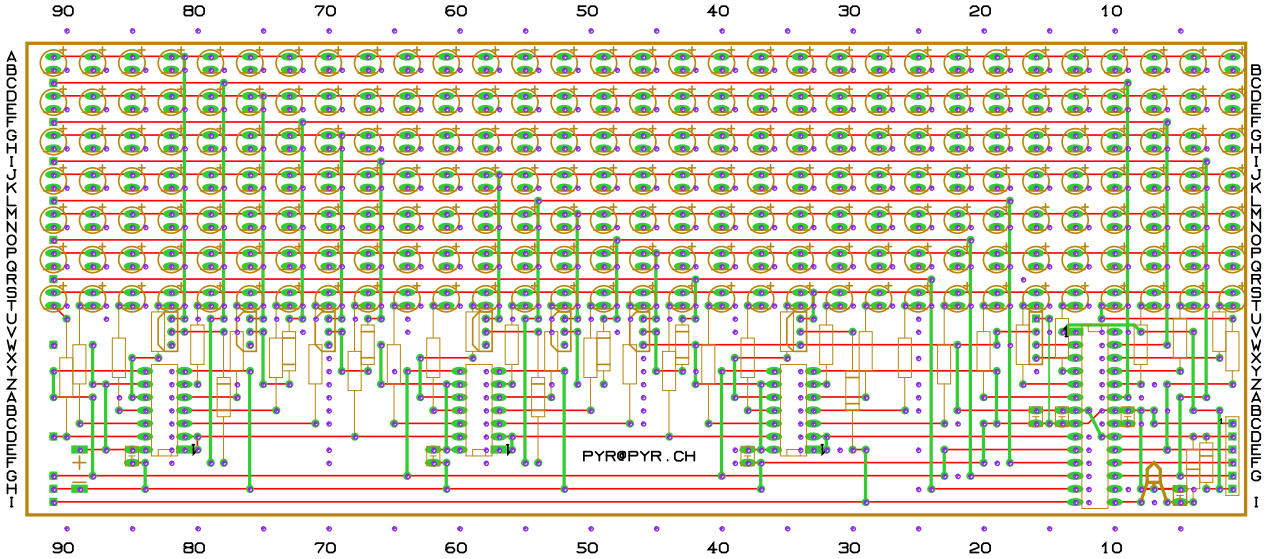
On remarque que ce montage fonctionne en « logique négative ». En effet, c'est un 0 logique qui active une ligne, en faisant conduire le transistor. C'est aussi un 0 sur une colonne qui permet au courant de passer dans une LED.

L'avantage de ce schéma est le suivant : comme les colonnes sont souvent nombreuses, on utilise des registres d'extension. Or beaucoup de ces registres sont capables de tirer un courant important lorsqu'une sortie est à 0, alors qu'ils ne sont capables de fournir qu'un petit courant lorsqu'ils sont à 1. C'est le cas du 74HC595.

Il existe une technologie, datant des années 1990, qui offrent un courant maximum supérieur à celui offert par la technologie de la série 74HC : c'est la série 74F (Fast, produite encore chez les fabricants Fairchild et NXP). J'ai souvent utilisé le 74F164. Il ne dispose pas du registre parallèle, comme sur le 74xx595. Mais comme les lignes sont commandées par des transistors, il suffit de ne sélectionner aucune ligne pendant le décalage série des données destinées aux colonnes.

Vous verrez à la page suivante le schéma complet d'un afficheur à 217 LED, soit 7 lignes de 31 LEDs. Le choix de 7 lignes a été dicté par le fait que c'est le nombre minimum de lignes qui permet l'affichage bien lisible des lettre majuscules (il en faudrait au moins 9, mais si possible 11 pour avoir de jolies minuscules).

Mais pourquoi alors des lignes de 31 LEDs ? Ce n'est pas un chiffre qu'un informaticien va aimer ! C'était le nombre maximal de LED qu'il est possible de placer sur un modèle très populaire de plaques du type Veroboard...



Logiciel de commande d'un affichage matriciel

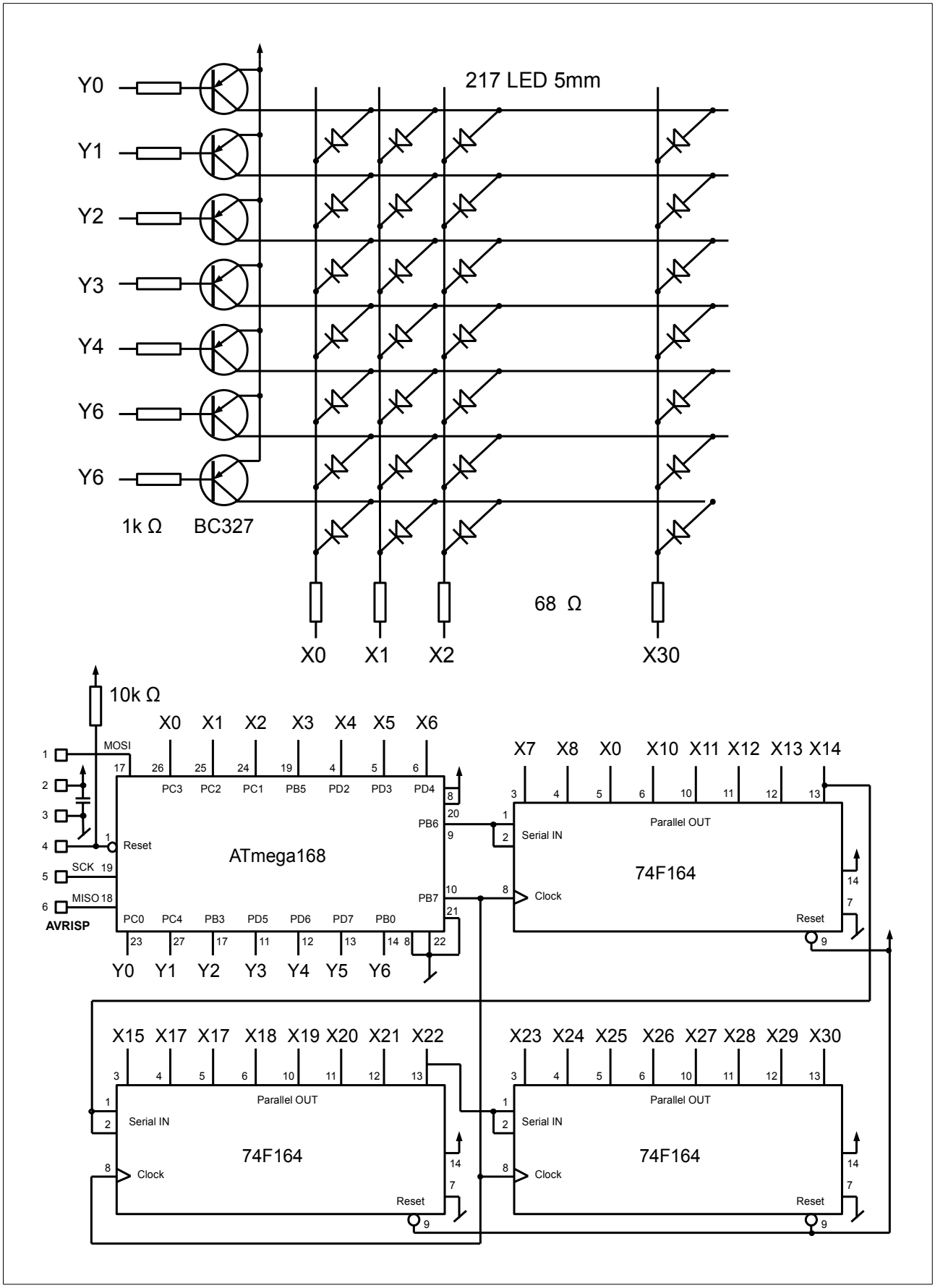
Bien qu'un journal lumineux, avec la gestion des caractères, est un programme un peu trop long pour être détaillé dans ce document, gérer un afficheur matriciel n'est pas si compliqué. Voici un programme destiné à un afficheur de 7 lignes de 8 LEDs, piloté par un MSP430G.

La procédure `Affiche` sélectionne successivement les lignes et allume les LEDs correspondantes. Une attente est faite pour rentre visibles les LEDs. Elle doit être réglée pour que l'impression de clignotement disparaisse. Il est nécessaire d'appeler en permanence cette procédure `Affiche`.

Entre les appels, on peut modifier les pixels à afficher, en agissant sur le tableau `Matrice`. C'est ce que font les procédures `SetPoint` et `ClearPoint`, auxquelle on donne les coordonnées du point à afficher.

Le programme principal initialise les sorties et quelques vraiables. Ensuite, il affiche en boucle la procédure `Ping`, qui réalise une animation qui simule une balle de ping-pong qui rebondit sans frottement sur les bords de l'afficheur.

```
#include <msp430g2553.h>

int matrice[7] = {0,0,0,0,0,0,0}; // mémorisation des pixels

#define LIGNES P1OUT
#define COLONNES P2OUT
void Affiche (int nb) { // Affiche la matrice pendant 10ms * nb
     int n; int anode; volatile int t;
     for (n=0; n<nb; n++){ // boucle de répétion du cycle
          for (anode=0; anode<7; anode++){ // boucle des lignes
               COLONNES = ~(1<<(anode+1)); // valeurs des LED
               LIGNES = ~matrice[anode];  // sélectionne la ligne
               for (t=0; t<100; t++) { // petite attente
               }
          }
     }
}

void SetPoint (int x, int y) { // allume un pixel
     matrice [y]|= (1<<x);
}
void ClearPoint (int x, int y) { // éteint un pixel
     matrice [y]&=~(1<<x);
}

void Ping () { // Joue une balle qui rebondit sur les bords
     int x = 1; int y = 1;
     int dx = 1; int dy = 1;
     for (x=0; x<8; x++) { SetPoint (x, 0); SetPoint (x, 6); }
     for (y=0; y<7; y++) { SetPoint (0, y); SetPoint (7, y); }
     x = 1; y = 1;
     while (1) {
          SetPoint (x, y);
          Affiche (20);
          ClearPoint (x, y);  x = x + dx; y = y + dy;
          if (x==6) {dx=-1;}
          if (x==1) {dx=1; }
          if (y==5) {dy=-1;}
          if (y==1) {dy=1;}
     }
}

void main () { // programme principal
     WDTCTL = WDTPW + WDTHOLD; // Stop watchdog timer
     P2SEL = 0; // usage normal de P2.6 et P2.7

     LIGNES = 0xFE; // tout en sortie
     COLONNES = 0xFF;  // tout en sortie

     while (1) { // boucle principale
          Ping () ;
     }
}
```


217 LED 5mm
Y0
Y1
Y2
Y3
Y4
Y6
Y6
1k Ω
BC327
68 Ω
X0
X1
X2
X30
10k Ω
X0 X1 X2 X3 X4 X5 X6
MOSI
PC3 PC2 PC1 PB5 PD2 PD3 PD4
PB6
PB7
Reset
ATmega168
SCK
MISO
AVRISP
PC0 PC4 PB3 PD5 PD6 PD7 PB0
Y0 Y1 Y2 Y3 Y4 Y5 Y6
X7 X8 X0 X10 X11 X12 X13 X14
Parallel OUT
Serial IN
74F164
Clock
Reset
X15 X17 X17 X18 X19 X20 X21 X22
X23 X24 X25 X26 X27 X28 X29 X30