# Chenillard au carré

## Et si nous faisions plus compliqué ?

Jean-Paul Brodier

**Le jeu de lumière de notre premier article (Chenillard E-blocks du mois de février 2007) est un peu simple. Nous allons essayer de l'étendre à un plus grand nombre de LED, mais le principe du chenillard simple ne s'y prête guère.**

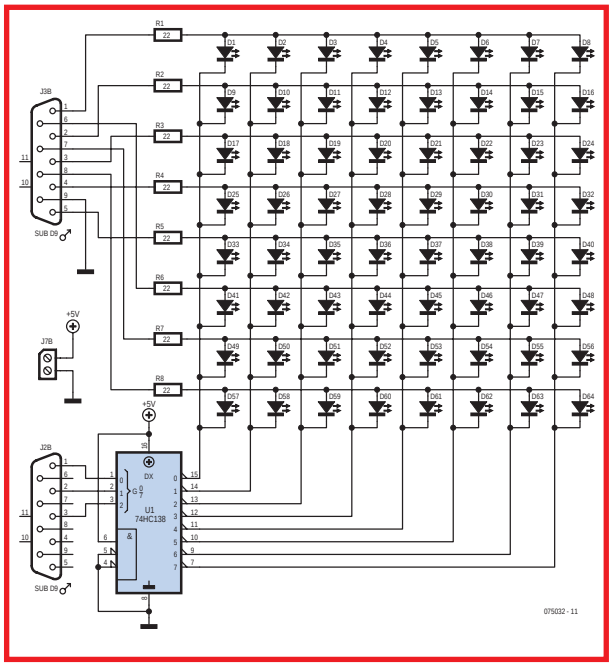


**Figure 1.** Les soixante-quatre LED sont disposées sur une matrice de 8 rangées par 8 colonnes.

Si nous voulions piloter 64 LED directement, il nous faudrait 64 lignes de sortie. Or nous avons pris le parti de nous en tenir à notre PIC16F88 à 18 broches. Nous allons donc, comme l'illustre le schéma de la **figure 1**, organiser les LED suivant une matrice de 8 rangées et 8 colonnes.

### Bataille navale

Chaque LED peut être désignée par le numéro de sa rangée et celui de sa colonne, comme dans une « bataille navale ». Si nous activons une rangée (niveau haut) et une colonne (niveau bas), la LED située à l'intersection, celle dont l'anode est connectée à cette rangée et la cathode à cette colonne, s'allume.

### Décodage

Les rangées seront alimentées par les sorties du PORT B, comme précédemment, mais nous ne disposons pas d'un nombre suffisant de lignes de sortie pour piloter directement les colonnes de notre matrice. C'est la raison d'être du circuit intégré 74HC138. Il s'agit d'un décodeur de trois lignes vers huit. Le nombre binaire appliqué aux

entrées A, B et C est représenté par le rang de la ligne de sortie active (au niveau bas).
Pour balayer les huit colonnes, nous devons faire varier de 0 à 7 le nombre COLONNE et l'appliquer en binaire sur les trois lignes de sortie du PORT A. L'augmentation est effectuée par COLONNE = COLONNE + 1, la limitation à 7 par l'opération Modulo 8.
Nous créons un nouveau fichier, appelé ici Ch2D0.fcf, avec « 2D » pour deux dimensions.
La première tâche (**figure 2**) est de créer la variable COLONNE et différentes variables pour les rangées. Pour ce faire, on traîne entre DÉBUT (*BEGIN*) et FIN (*END*) un rectangle Calcul (*Calculation*), puis on édite ses propriétés. Les variables sont ajoutées par Variables… (*Browse for variable*) puis Ajouter Variable (*Add New Variable*).
Les nouvelles variables une fois créées, il reste à les utiliser dans le programme, c'est-à-dire augmenter la valeur de COLONNE et copier cette valeur sur le PORT A pour balayer les huit colonnes d'une part, et d'autre part appliquer sur le PORT B la valeur correspondante de RANG.

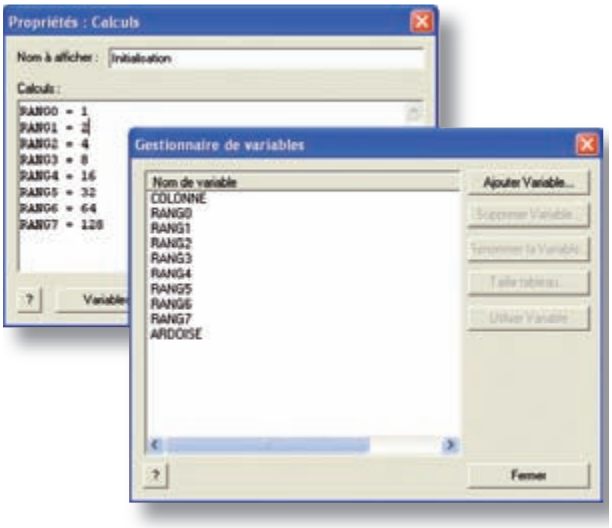


**Figure 2.** On crée de nouvelles variables et on leur donne une valeur initiale.

## Langage C

Nous avons recours au langage C quand c'est plus rapide ou plus simple que les symboles de l'organigramme, par exemple quand il n'y a qu'une opération mathématique à effectuer.
La première case C CODE du corps du programme, intitulée Modulo 8, (**figure 3**) comporte une seule instruction en langage C (clic droit, Propriétés… (*Properties*), comme d'habitude…).

```
FCV_COLONNE = FCV_COLONNE % 8;
```

Le préfixe FCV_ devant la variable COLONNE indique au compilateur qu'elle est définie dans Flowcode. Il est obligatoire dans les instructions en C.
Pour les instructions en assembleur, le préfixe obligatoire est _FCV_.

Le signe % (pourcent) est celui de l'opération *modulo*, le résultat est le reste de la division par 8.
Notre variable COLONNE augmente de 1 à chaque passage dans la boucle. Sans autre opération, la valeur croîtrait jusqu'à 255, puis repasserait à zéro, en utilisant les huit bits du registre. Cela n'est pas compatible avec le format de sortie, qui ne doit occuper que trois lignes du PORT A, donc passer de 7 à 0.
Quand la valeur de COLONNE augmente jusqu'à 7, 111 en binaire, le reste de la division par 8 est égal à COLONNE. Quand la valeur arrive à 8, le reste de la division (modulo) est 0, le compteur redémarre de 0.
Il est possible d'obtenir le même résultat avec un masque, un opérateur logique ET. On a vu que les valeurs autorisées de COLONNE vont de 000 à 111 en binaire, 0 à 7 en décimal. La ligne en langage C devient :

```
FCV_COLONNE = FCV_COLONNE & 7;
```

Le signe & (esperluette) représente en langage C l'opération logique ET.
Le report de la valeur de COLONNE sur le PORT A ne peut pas se faire simplement et brutalement. Pour ne pas bousculer l'état des lignes RA3 à RA7, il faut ordonner le masquage de la valeur à écrire. C'est fait (**figure 4**) dans la fenêtre Propriétés (*Properties*) de la première case Sortie (*OUTPUT*) (clic droit, Propriétés (*Properties*)). Cocher Masque : (*Use Masking* :) et les bits qui doivent être écrits, les autres ne seront pas affectés.
À chaque passage dans la boucle, nous appliquons sur le PORT B une nouvelle valeur de RANG, de façon à déplacer le point lumineux dans les deux directions.
C'est le rôle du deuxième fragment de langage C. La valeur d'ARDOISE est changée en fonction de celle de COLONNE, puis reportée sur le PORT B. On aura compris que l'octet ARDOISE sert à écrire temporairement une valeur à reporter plus loin.
Quand le programme est compilé sans erreur, on peut l'envoyer vers le microcontrôleur Puce – Compiler vers Puce (*PIC - Compile to PIC*), comme nous l'avons vu dans l'article précédent. Nous avons maintenant un point lumineux qui se déplace suivant une diagonale de notre carré de 8 x 8, mais on ne voit qu'un seul point à la fois. Pour voir vraiment une ligne, il faudrait allumer en même temps toutes les LED de la diagonale. La question qui se pose maintenant est : comment allumer en même temps A1 et B2 ? Si nous activons les rangées A et B en même temps que les colonnes 1 et 2, nous aurons quatre LED allumées et non deux. La réponse s'appelle multiplexage. Écrire 1 dans RANGEE et activer la colonne 1 pour allumer la LED A1 ; écrire 2 dans RANGEE et activer la colonne 2 pour allumer la LED B2, et ainsi de suite. Si les opérations se succèdent suffisamment vite, notre œil voit les LED A1, B2, etc. allumées en même temps.

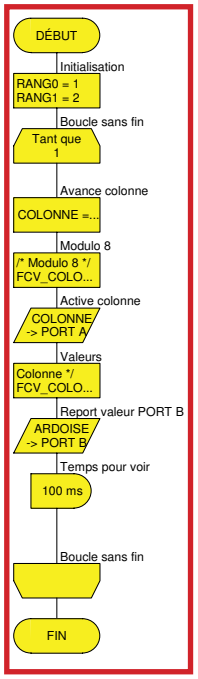


**Figure 3.** Le programme de test de la matrice d'affichage.

Nous allons modifier le programme dans ce sens, non sans l'avoir renommé, par exemple Ch2D1 (Fichier =- En registrer sous…) (*File – Save as…*)

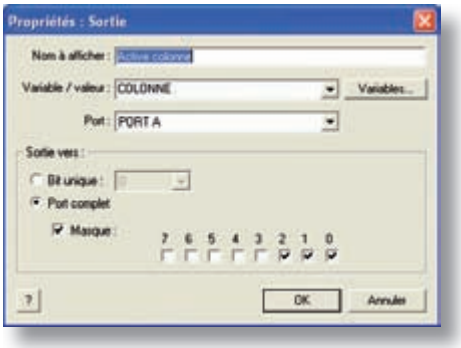


**Figure 4.** Le report de la variable COLONNE sur le PORT A.

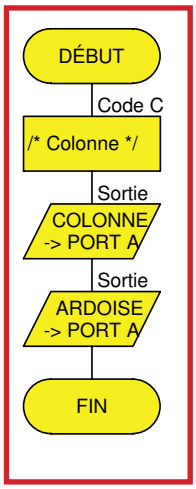


**Figure 7.** La routine d'interruption dans toute sa simplicité.

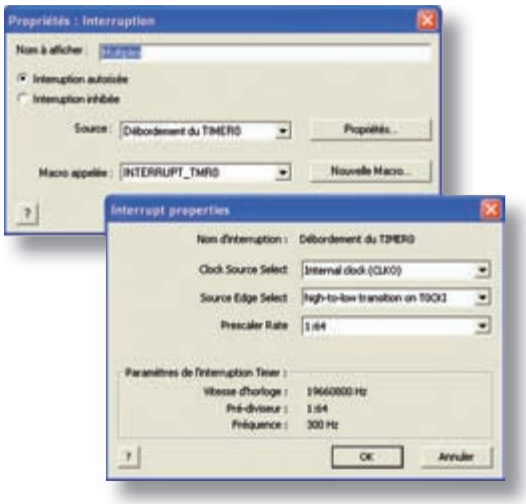


**Figure 5.** Configuration de l'interruption.

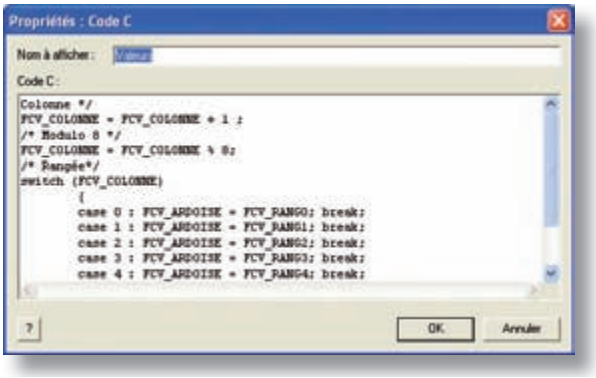


**Figure 6.** Le code C inclus dans la routine d'interruption.

## Interruptions

L'opération de balayage doit être extrêmement rapide et elle ne nécessite pas d'intelligence. C'est une tâche de bulbe rachidien plutôt que de cerveau. Nous allons donc la confier à une interruption déclenchée par un temporisateur du microcontrôleur.

On traîne un hexagone INT sous l'initialisation des variables et on configure l'interruption comme le montre la **figure 5**.
Chaque fois que le compteur TMR0 passe par zéro, soit 300 fois par seconde, il provoque l'exécution de la routine (dite MACRO en langage FlowCode) INTERRUPT_TMR0. À nous de reporter dans cette routine les instructions qui se trouvaient jusqu'ici dans le corps du programme. On clique sur Macro dans la barre de menu, puis sur (*Edit/Delete*), puis sur INTERRUPT_TMR0. On traîne une boîte C CODE au début et on y copie les instructions du programme précédent.

Pour s'éviter beaucoup de peine et de risques d'erreur, il suffira d'ouvrir (avec Bloc-notes ou un autre éditeur simple) le fichier Ch2D0.c créé par la dernière compilation. On marque à la souris le bloc à recopier, on appuie sur Ctrl-C (pour copier), on repasse dans la fenêtre de l'éditeur Flowcode et on appuie sur Ctrl-V (pour coller) (**figure 6**).
Après le code C, il reste à placer les symboles de sortie pour PORT A et PORT B (**figure 7**). Le programme de démonstration du multiplexage (**figure 8**) se résume à la routine d'interruption.

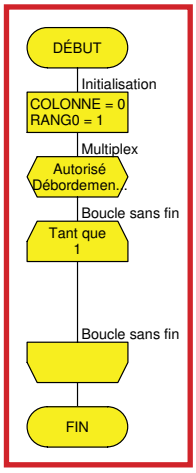


**Figure 8.** La démonstration du multiplexage. On affiche maintenant une diagonale fixe.

## Norme ANSI et langage C à la MPLAB

Le langage C compris par MPLAB (l'outil de développement Microchip) semble ne pas correspondre exactement à la norme ANSI.
L'opération de masquage peut s'écrire :

```
FCV_COLONNE &= 7;
```

Ces formes comprimées au maximum font la joie des programmeurs en C, principalement parce qu'elles sont difficiles à lire, y compris par eux-mêmes, déroutantes pour les novices et propres à susciter chez eux le respect pour les maîtres.
Curieusement, l'expression :

```
FCV_COLONNE %= 8;
```

correcte selon la norme ANSI, est refusée par le compilateur.

Les opérations copier-coller sont devenues tellement faciles avec les ordinateurs et les éditeurs modernes qu'on peut se dispenser de ces économies de quelques frappes au clavier. Dans les deux cas, le résultat est le même, comme on peut le vérifier en consultant à chaque fois le fichier .LST produit par le compilateur et l'assembleur. Quand il arrive que le compilateur refuse une ligne de langage C ou d'assembleur, il se contente de le dire, sans donner d'explication sur la cause. On se reportera donc toujours au fichier .LST, qui contient tous les détails du programme en assembleur.

Autre petite différence entre ANSI et MPLAB : les noms de variables sont obligatoirement en majuscules dans MPLAB, alors que la norme ANSI reconnaît les deux casses et les distingue.

## On se bouge !

Puisque notre but était d'avoir un jeu de lumière animé, nous allons faire défiler la diagonale, et même en ajouter une deuxième, à quatre pas d'écart. Ce sera le programme Ch2D2.
Tout d'abord, les valeurs initiales de RANG sont remplacées de façon à allumer deux points, ensuite la boucle principale décale le contenu de tous les registres.
Les valeurs hexadécimales successives sont 0x11, 0x22, 0x44, 0x88 pour RANG0 à RANG3. Ces quatre valeurs se répètent pour RANG4 à RANG7. Il faut les écrire en décimal parce que FlowCode ne comprend ni le binaire, ni l'hexadécimal, à la mode C (0x11) ou à la mode Intel (11h) : 17, 34, 68, 136.
Pendant le déroulement du programme, le décalage se fait dans chaque registre au moyen d'une multiplication par 2. Lorsque le bit à décaler est le plus à gauche, pour la valeur 136, 10001000 en binaire, on saute à la valeur initiale 17, ou 00010001.
Nous n'avons pas deux lignes, mais trois ou quatre suivant le moment (**figure 9**).
Il est possible maintenant de changer les valeurs initiales, pour produire par exemple une seule diagonale (1, 2, 4, 8, 16, 32, 64, 128), des chevrons (1, 2, 4, 8, 4, 2, 1, 2), ou une ligne brisée en plusieurs points. Le programme lui-même peut compter les balayages et, après un nombre donné, changer les valeurs pour produire un autre dessin, chargé à partir de blocs de constantes définies dans la mémoire de programme.
Bien que ce micro-contrôleur soit parmi les plus petits, la dernière version de ce programme n'occupe qu'une petite partie de la mémoire Flash disponible : 391 mots de programme sur 4 096.

## Réalisation

Le montage peut être réalisé sur une platine perforée ou sur la platine proposée ci-contre (figure 10). Elle est dotée de 2 embases Sub-D mâles, J2B et J3B nommées à dessein ainsi parce qu'elles viennent s'enficher sur les prises DB9 femelles J2 (PORT A) et J3 (PORT B) de la platine du Multiprogrammateur. L'alimentation sera prélevée sur les bornes +V du bornier à vis J7 côté +5 V, la masse est déjà raccordée par la broche 9 des prises DB9.
Les résistances de 22 Ω sont plutôt symboliques, car l'intensité dans les LED est limitée (horreur, hérésie !) au débit maximal des sorties du micro-contrôleur.
Le courant total débité par les huit broches du PORT B est de 100 mA.
En tout état de cause, on a intérêt à monter des LED à haut rendement.
Le circuit double face est loin d'être nécessaire. Les connexions transversales peuvent être réalisées très simplement avec du fil à wrapper. Commencer par étamer assez généreusement les pastilles libres au-dessus des anodes des LED. Ensuite pousser le fil à wrapper dans la goutte de soudure avec la panne du fer à souder et une force suffisante. L'isolant plastique s'écrase, fond et se rétracte, juste assez pour que le cuivre vienne se souder à la pastille. Vérifier par une traction modérée que la soudure tient et passer à la suivante. Couper le fil après la dernière soudure de la rangée.

(075032-I)

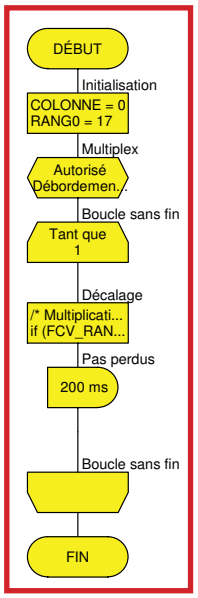


**Figure 9.**
**La simplicité de l'organigramme cache de nombreuses déclarations et instructions en C.**

## Liste des composants

**Résistances:**
R1 à R8 = 22 Ω

**Semi-conducteurs:**
D1 à D64 = LED (3 ou 5 mm)
IC1 =74HC138

**Divers:**
J2B,J3B = embase sub-D 9 points mâle encartable
J7B = bornier à vis 2 points encartable
platine via ThePCBShop (EPS075032-1)

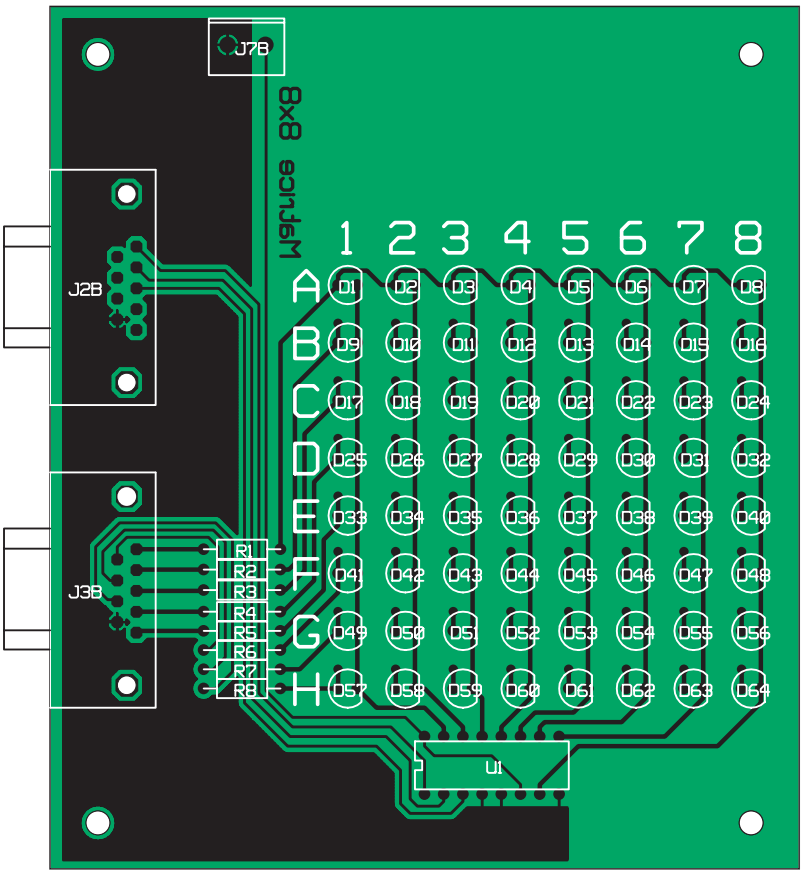


**Figure 10.**
**La platine peut être réalisée en simple face, avec une série de ponts pour la mise en parallèle des 8 LED de chaque rangée. L'implantation des composants réclame un peu d'attention pour l'orientation des LED. Le dessin des pistes (recto-verso) de cette platine est disponible au téléchargement sur le site Elektor.**