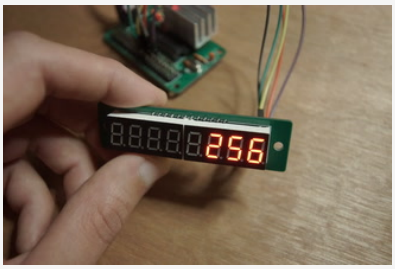
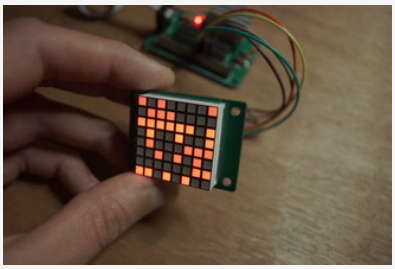
Afficher des caractères avec Max7219.h

Max7219.h est une classe qui permet d'utiliser des afficheurs à diodes électroluminescences (digits et matrice) pilotées par le composant **MAX7219**, ce dernier étant connecté en **SPI** avec le microcontrôleur (l'automate programmable):





Ports des automates programmables concernés par le SPI:

> ***Automate programmable MODULABLE M20:***
> *- Port 11 (PB2) = SS (slave select)*
> *- Port 12 (PB3) = MOSI (master output slave input)*
> *- Port 13 (PB4) = MISO (master input slave output)*
> *- Port 14 (PB5) = SCK (serial clock)*
>
> ***Automate programmable MODULABLE M32:***
> *- Port 5 (PB4) = SS (slave select)*
> *- Port 6 (PB5) = MOSI (master output slave input)*
> *- Port 7 (PB6) = MISO (master input slave output)*
> *- Port 8 (PB7) = SCK (serial clock)*

Exemple pour afficher un nombre entier:

```
#include "../module/1284p/Max7219.h"

int main()
{
        Max7219 myDisplay = Max7219 (5, 1);

        myDisplay.integer (256, 1, 8, true);

        return 0;
}
```

Dans cet exemple, un objet **myDisplay** de type **Max7219** est déclaré:
- Le 1er paramètre indique l'utilisation du port numéro **5** de l'automate programmable pour sélectionner l'esclave (slave select) sur le bus **SPI**.
- Le 2ème paramètre **1** signifie que l'afficheur est en **position 1 en cascade** sur le bus SPI.

> *Par habitude en SPI, je connecte la sélection de l'esclave au port de l'automate programmable relié à la fonction SS (slave select) du microcontrôleur quand je le peux, **ce qui permet de regrouper tout le câblage relatif au SPI**.*
>
> *Mais en réalité, cette entrée/sortie n'est en fait un port SS à la fonction spécifique que lorsque le SPI est configuré en **mode esclave**, ce qui n'arrive jamais avec MODULE. Vous pouvez donc choisir l'esclave sur la sortie SS du microcontrôleur, ou tout autre port d'entrée/sortie disponible (sauf MISO qui même inutilisé doit rester comme une entrée libre lorsque le SPI est en fonctionnement).*

Puis la fonction **integer** de l'objet **myDisplay** est appelée:
- Le 1er paramètre **256** est le nombre entier à afficher (sur 32 bits maximum).
- Le 2ème paramètre **1** indique la position ou commence l'affichage de ce nombre (position 1).
- Le 3ème paramètre **8** indique la position ou termine l'affichage de ce nombre (position 8).
- Le 4ème paramètre **true** permet de caler le nombre affiché à droite (false pour le caler à gauche) dans l'intervalle de positions sélectionnée (positions 1 à 8).

> *Dans cet exemple, le nombre à afficher (256) comporte seulement **3 digits**, l'intervalle de positions qui lui est accordé pour être affiché comporte **8 digits** (positions 1 à 8) c'est-à-dire **la totalité de l'afficheur**. En conséquence, le 4ème paramètre (calage) est effectif lorsque le nombre à afficher prend moins de place que l'intervalle sélectionnée (ceci permet de caler le nombre sur la partie gauche ou droite de l'intervalle choisie).*

Autres fonctions utiles:

D'autres fonctions existent, comme l'affichage des nombres décimaux, l'affichage de texte, de points ou lignes sur une matrice à dels, ou encore le réglage de la luminosité (par PWM), etc...

Exemple pour afficher un nombre décimal:

```
#include "../module/1284p/Max7219.h"

int main()
{
	Max7219 myDisplay = Max7219 (5, 1);

	myDisplay.decimal (2.56, 3, 7, 1, false);

	return 0;
}
```

Paramètres de la fonction **decimal** de l'objet **myDisplay**:
- Le 1er paramètre **2.56** est le nombre décimal à afficher (en virgule flottante 32 bits).
- Le 2ème paramètre **3** indique la position ou commence l'affichage de ce nombre (position 3).
- Le 3ème paramètre **7** indique la position ou termine l'affichage de ce nombre (position 7).
- Le 4ème paramètre **1** indique le nombre de chiffres à afficher après la virgule.
- Le 5ème paramètre **false** permet de caler le nombre affiché à gauche (true pour le caler à droite) dans l'intervalle de positions sélectionnée (positions 3 à 7).

Exemple pour afficher un nombre entier et décimal l'un à coté de l'autre:

```
#include "../module/1284p/Max7219.h"

int main()
{
	Max7219 myDisplay = Max7219 (5, 1);

	myDisplay.integer (256, 1, 4, true);
	myDisplay.decimal (1.28, 5, 8, 1, true);

	return 0;
}
```

Dans cet exemple, le nombre entier **256** sera affiché dans l'intervalle de positions **1** à **4** et **calé à droite** dans son intervalle de positions.

Le nombre décimal qu'en à lui sera affiché dans l'intervalle de positions **5** à **8**, avec **1 chiffre après la virgule**, et à l'instar du nombre décimal il sera **calé à droite** dans son intervalle de positions sur l'afficheur.

> *Le nombre entier affiché en premier ne sera pas effacé tant que des caractères ne viendront pas écraser les positions (intervalles) qu'il occupe. Si vous souhaitez effacer tout ce qui est affiché sur l'afficheur, utilisez la fonction **clearDevice**.*

Exemple pour afficher du texte:

www.sylvainmahe.xyz

LE BLOG
de Sylvain Mahé
contact@sylvainmahe.xyz

```
#include "../module/1284p/Max7219.h"

int main()
{
        Max7219 myDisplay = Max7219 (5, 1);

        myDisplay.word ("hello   ");

        return 0;
}
```

La fonction **word** de l'objet **myDisplay** ne prend qu'un seul paramètre, ici c'est le mot **hello** qui est affiché (sur un afficheur à digits).

Il est également possible d'afficher des nombres avec cette fonctions, pour cela il suffit d'écrire en paramètre par exemple **"123.45 "**, ou bien encore d'afficher les lettres avec leurs points (virgules) correspondants (en suffixe) en écrivant **"..hel.l.o "**. Ici les deux **l.** s'afficheront avec un point en dessous.

> *Un caractère à la fonction spécifique existe, c'est la barre oblique /. Celle-ci permet de terminer le rafraîchissement de l'affichage à cet endroit, et par conséquent de ne pas écraser les digits suivants, car en effet avec la fonction word, **par défaut c'est l'intégralité de l'affichage qui est mis à jour**.*

Exemple pour connecter plusieurs afficheurs en cascade et/ou en parallèle sur le bus SPI:

```
#include "../module/1284p/Max7219.h"

int main()
{
        Max7219 myDigitDisplay = Max7219 (5, 1);
        Max7219 myMatrixDisplay = Max7219 (5, 2);
        Max7219 myWordDisplay = Max7219 (4, 1);

        myDigitDisplay.integer (256, 1, 8, true);

        myMatrixDisplay.dot (7, 2);
        myMatrixDisplay.dot (8, 1);

        myWordDisplay.word ("hello   ");

        return 0;
}
```

Explication des connexions:
- La broche SS (slave select) de l'afficheur relatif à l'objet **myDigitDisplay** est connectée au port **5** de l'automate programmable, il est le **1er périphérique en cascade** sur le bus SPI.
- La broche SS de l'afficheur relatif à l'objet **myMatrixDisplay** est également connectée au port **5** de l'automate programmable, mais cet afficheur est défini comme étant le **2ème périphérique en cascade** sur le bus SPI.
- Le 3ème afficheur relatif à l'objet **myWordDisplay** voit sa broche SS connectée au port **4** de l'automate programmable (il est donc en parallèle par rapport aux autres) et est indiqué comme étant le seul périphérique sur cette partie du bus SPI.

> *Si votre alimentation le permet, ce principe de connexions en cascade (série) et/ou en parallèle vous permet de faire fonctionner un très grand nombre d'afficheurs sur un même automate programmable.*

Exemple pour régler la luminosité de l'affichage:

www.sylvainmahe.xyz

LE BLOG
de Sylvain Mahé
contact@sylvainmahe.xyz

```
#include "../module/1284p/Max7219.h"

int main()
{
        Max7219 myDisplay = Max7219 (5, 1);

        myDisplay.brightness (4);

        return 0;
}
```

La fonction **brightness** est appelée et prend en paramètre un nombre entier de **1** à **16** (16 étant la luminosité maximale).

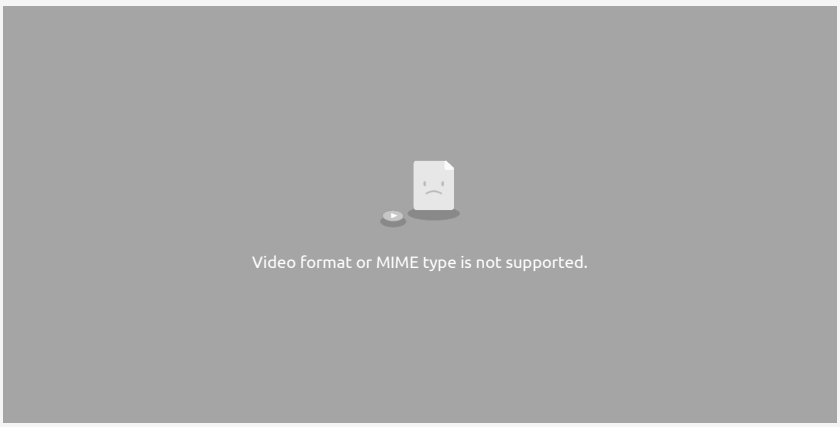
> *Vous pouvez réaliser des effets complexes avec toutes les fonctions qui vous sont proposées dans cette classe, libre à vous de composer votre affichage d'une manière harmonieuse !*

Récapitulatif des fonctions de cette classe:

```
Max7219 (const unsigned char PIN_SS, const unsigned char DEVICE);
void brightness (const unsigned char BRIGHTNESS);
void digit (const unsigned char DIGIT, const bool COMMA, const unsigned char POSITION);
void integer (const signed long INTEGER, const unsigned char POSITION_MIN, const unsign
void decimal (const float DECIMAL, const unsigned char POSITION_MIN, const unsigned cha
void character (const char *CHARACTER, const unsigned char POSITION);
void word (const char *WORD);
void byte (const unsigned char BYTE);
void draw (const unsigned char DRAW, const unsigned char POSITION);
void drawDevice (const unsigned char DRAW1, const unsigned char DRAW2, const unsigned c
void dot (const unsigned char X, const unsigned char Y);
void clearLine (const unsigned char LINE);
void clearDevice();
```

Un effet facile à réaliser avec MODULE:

Avec MODULE, il vous est possible de combiner les fonctionnalités d'une ou de toutes les classes sans problèmes d'interactions si vous le souhaitez, il devient alors très aisé de réaliser vos propres effets en utilisant plusieurs classes dédiées à la réalisation de certaines tâches simples ou complexes:



Le programme:

www.sylvainmahe.xyz

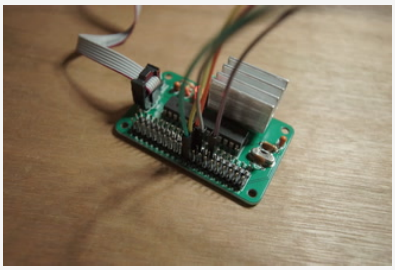
LE BLOG
de Sylvain Mahé
contact@sylvainmahe.xyz

```
#include "../module/1284p/Max7219.h"
#include "../module/1284p/Random.h"
#include "../module/1284p/Timer.h"

int main()
{
	unsigned char line = 1;
	Max7219 matrix = Max7219 (5, 1);

	Random::seed (25);

	while (true)
	{
		for (line = 1; line <= 8; line++)
		{
			matrix.draw (Random::integer (0, 255), line);
		}

		Timer::pause (100);
	}

	return 0;
}
```

Dans cet exemple, l'utilisation de la classe **Random.h** permet la génération de **positions aléatoires** ce qui a pour conséquence d'afficher les pixels au hasard sur la matrice.

Connexions (afficheur sur automates programmables):

> *- Broche +5V sur broche +5V disponible.*
> *- Broche GND sur broche GND disponible.*
> *- Broche SS (slave select) sur port SS ou tout autre port d'entrée/sortie disponible (sauf MISO qui doit rester libre sauf si il est utilisé par un périphérique également SPI).*
> *- Broche MOSI (master output slave input) sur port MOSI.*
> *- Broche SCK (serial clock) sur port SCK.*



> *Les plans de fabrication de mes afficheurs (digits et matrice) sont disponibles dans les sections "Téléchargements" et "Fabrications et divers réalisations" en page d'accueil.*