

1

Représentation numérique de l'information

L'informatique en général peut se définir comme la science du traitement de l'information. Préalablement au traitement de cette information, encore faut-il commencer par la représenter. Quelle que soit la teneur des données' (texte, graphique, son...), celles-ci sont codées sous forme numérique, de même que le programme qui les traite (suite d'instructions en langage machine). Cette représentation numérique ne s'appuie pas sur le système décimal, mais sur le système binaire (0 et 1). Cette logique à deux états découle directement des propriétés physiques et électriques des composants employés (processeurs, mémoires, etc.) : le courant passe, ou ne passe pas. Pour comprendre le langage MIDI, il faut donc déjà comprendre le vocabulaire de base, c'est-à-dire le système binaire et sa représentation hexadécimale.

1.1 Le système binaire

Le système de représentation numérique que nous exploitons couramment se nomme système décimal. Le terme "décimal" vient du fait que nous utilisons dix chiffres (de 0 à 9) pour représenter les nombres. Cependant, nous pourrions fort bien en utiliser moins (par exemple 2, comme le 0 et le 1), ou plus (à condition de créer de nouveaux symboles s'ajoutant aux dix que nous employons déjà). Ou encore, pourquoi ne pas remplacer les chiffres par des lettres (ou par n'importe quelle autre famille de symboles). Avec le système décimal, en substituant par exemple les lettres A à J aux chiffres 0 à 9, l'opération $9 + 6 = 15$ se transformerait en $J + G = BF$. Tout cela n'est qu'une question de convention.

La notion de base

Le nombre de symboles (chiffres) dont nous disposons pour représenter des nombres détermine le type de la base. Ainsi, le système décimal est également appelé système à base 10, le système binaire système à base 2, etc. D'une manière générale, pour une base X, un nombre à Y chiffres (par exemple 4 chiffres, en les notant A, B, C, D) se traduit en décimal de la manière suivante :

$$ABCD = AxX^3 + BxX^2 + CxX^1 + DxX^0$$

Bien évidemment, les chiffres A, B, C et D seront toujours inférieurs à X, puisque le propre d'une base est d'utiliser X symboles pour représenter les nombres. En décimal, A, B, C et D seront compris entre 0 et 9.

Pour mémoire, porter un nombre à une certaine puissance (appelée "exposant") consiste à le multiplier par lui-même autant de fois que l'indique cet exposant :

$$10^4 = 10 \times 10 \times 10 \times 10 = 10\ 000$$

$$2^7 = 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 128$$

Lorsque l'exposant est égal à zéro, le résultat est égal à un, indépendamment du nombre élevé à cette puissance. Cette particularité se démontre facilement. En effet :

$$\begin{aligned}
 X^n \times X^m &= (XxXxX\dots xX)x(XxXxX\dots xX) \\
 &\quad \leftarrow n \text{ termes } \rightarrow \leftarrow m \text{ termes } \rightarrow \\
 &= (XxXxX\dots xX) \\
 &\quad \leftarrow n + m \text{ termes } \rightarrow \\
 &= X^{n+m}
 \end{aligned}$$

En posant $m = 0$, nous obtenons :

$$X^n \times X^0 = X^{n+0} = X^n$$

D'où nous déduisons que $X^0 = 1$. Tout comme 0 est l'élément neutre de l'addition, 1 est l'élément neutre de la multiplication.

Ainsi, en décimal, nous pouvons poser les égalités suivantes :

$$\begin{aligned}
 5\ 412 &= 5 \times 10^3 & + & 4 \times 10^2 & + & 1 \times 10^1 & + & 2 \times 10^0 \\
 &= 5 \times 10 \times 10 \times 10 & + & 4 \times 10 \times 10 & + & 1 \times 10 & + & 2 \times 1 \\
 &= 5 \times 1000 & + & 4 \times 100 & + & 1 \times 10 & + & 2 \times 1
 \end{aligned}$$

Le chiffre 2 correspond au rang des unités (10^0), 1 à celui des dizaines (10^1), 4 à celui des centaines (10^2), et 5 à celui des milliers (10^3).

Conversion de binaire en décimal

En binaire, le principe est similaire, sauf qu'il n'existe que deux chiffres pour codifier un nombre : le 0 et le 1. Les généralités précédentes sont donc en partie erronées, puisqu'elles s'appuient sur quatre valeurs symboles A, B, C et D, alors que nous sommes ici limités à deux. Remplaçons par exemple ABCD par BABB :

$$BABB = B \times 2^3 + A \times 2^2 + B \times 2^1 + B \times 2^0$$

Remplaçons ensuite les symboles A et B par les chiffres 0 et 1 :

$$\begin{aligned} 1011 &= 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\ &= 1 \times 2 \times 2 \times 2 + 0 \times 2 \times 2 + 1 \times 2 + 1 \times 1 \\ &= 1 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1 \\ &= 11 \text{ (en décimal)} \end{aligned}$$

N.B. : Pour éviter toute confusion, un nombre binaire est parfois précédé du symbole %, par exemple %11.

Conversion de décimal en binaire

Pour convertir un nombre décimal en binaire, il suffit d'effectuer une suite de divisions par 2 de la manière suivante : diviser ce nombre par 2, en noter le reste, diviser le quotient par 2, en noter le reste, et ainsi de suite, jusqu'à obtention d'un quotient égal à 0 et d'un reste égal à 1. Pour une division par X, le reste est fatalement inférieur à X (en binaire, il sera donc égal à 0 ou à 1). En regroupant de gauche à droite les restes de toutes les divisions (de la dernière à la première), nous aboutissons à l'équivalent binaire du nombre décimal. Ainsi, avec l'exemple précédent (11 en décimal), nous obtenons quatre divisions successives :

$$\begin{aligned} 11:2 &= 5 \text{ reste } 1 \\ 5:2 &= 2 \text{ reste } 1 \\ 2:2 &= 1 \text{ reste } 0 \\ 1:2 &= 0 \text{ reste } 1 \end{aligned}$$

D'où nous déduisons que :

11 décimal = 1011 binaire.

Avec 242, nous obtenons huit divisions successives :

$$242:2 = 121 \text{ reste } 0$$

```

121:2 = 60  reste 1
60:2  = 30  reste 0
30:2  = 15  reste 0
15:2  = 7   reste 1
7:2   = 3   reste 1
3:2   = 1   reste 1
1:2   = 0   reste 1

```

D'où nous déduisons que :

242 décimal = 11110010 binaire.

Exemples de calculs en binaire

Quelle que soit la base utilisée, ajouter 1 à un nombre dont le chiffre de droite est le plus élevé de cette base (9 en décimal, 1 en binaire, etc.), équivaut :

- A poser une retenue, c'est-à-dire, soit si le nombre est constitué de plusieurs chiffres, à additionner 1 au deuxième chiffre en partant de la droite, soit, si le nombre n'est constitué que d'un chiffre, à inscrire le chiffre 1 à sa gauche.

- A "forcer" à zéro (la valeur la plus faible de la base) le chiffre de droite.

Voici deux exemples en décimal :

```

59 + 1 = 60
9  + 1 = 10

```

Voici deux exemples en binaire :

```

1001 + 1 = 1010 (10 en décimal)

```

```

1 + 1 = 10 (2 en décimal)

```

En admettant que le deuxième chiffre en partant de la droite soit lui aussi le plus élevé de la base, la retenue se propagera d'une position vers la gauche.

Voici un exemple de propagation de la retenue en décimal :

```

  9999
+     0
-----
10.000

```

Voici un exemple de propagation de la retenue en binaire :

```

  1111 (15 en décimal)
+     1 (1)
-----
10000 (16)

```

Ces explications concernant le dépassement, lors d'une addition, du chiffre le plus élevé d'une base, ainsi que la propagation de la retenue, devraient vous suffire à adapter au binaire les mécanismes de l'addition, de la soustraction, de la multiplication et de la division en base décimale. On pourra s'exercer avec les opérations suivantes :

$$\begin{array}{r}
 1011 \text{ (11)} \\
 + 1100 \text{ (12)} \\
 \hline
 10111 \text{ (23)}
 \end{array}
 \qquad
 \begin{array}{r}
 101 \text{ (5)} \\
 + 1011 \text{ (11)} \\
 + 111 \text{ (7)} \\
 \hline
 = 10111 \text{ (23)}
 \end{array}$$

$$\begin{array}{r}
 111 \text{ (7)} \\
 \times 101 \text{ (5)} \\
 \hline
 111 \\
 000 \\
 111 \\
 \hline
 = 100011 \text{ (35)}
 \end{array}
 \qquad
 \begin{array}{r}
 100101 \text{ (37)} \\
 - 1111 \text{ (15)} \\
 \hline
 = 10110 \text{ (22)}
 \end{array}$$

Codification binaire : le bit, l'octet et le nybble

En informatique, la plus petite unité numérique (le chiffre) se nomme "bit" (contraction de *binary digit*, ou chiffre binaire). Un bit prend donc les valeurs 0 ou 1. Les composants (processeurs, mémoires) sont capables de traiter simultanément plusieurs bits (un nombre de plusieurs chiffres). Le nombre de ces bits dépend du nombre de "lignes" parallèles que possède ce composant. L'ensemble de ces lignes est regroupé sous l'appellation de "bus". Le nombre de lignes est toujours un multiple de 8 (8, 16, 24, 32), par souci de compatibilité avec l'ensemble des systèmes informatiques, 8 bits représentant un octet, ou *byte* (nombre binaire de huit chiffres). Un groupe de 16 bits est généralement appelé "mot" (*word*), et un groupe de 32 bits "mot long" (*long word*). Pour ces groupes de 8, 16 et 32 bits, on emploie fréquemment les abréviations B, W et L. L'octet est l'unité élémentaire utilisée par l'interface MIDI pour transmettre des messages. Les nombres qu'il représente s'échelonnent entre une valeur minimale de 00000000 (0) et une valeur maximale de 11111111 (255) :

$$\begin{aligned}
 11111111 &= 1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\
 &= 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 \\
 &= 255
 \end{aligned}$$

Tout comme un nombre décimal de quatre chiffres est limité à la représentation de 10 000 valeurs différentes (de 0 à 9 999), un nombre binaire de huit chiffres est limité à la représentation de 256 valeurs différentes (de 0 à 255).

Dans la norme MIDI, seuls les 7 bits de droite sont utilisés pour représenter une information. Nous le verrons par la suite, le bit de gauche est réservé à un usage spécifique. Cela débouche sur une représentation de 128 valeurs par octet (de 0 à 127) :

$$\begin{aligned}
 x1111111 &= 1x2^6 + 1x2^5 + 1x2^4 + 1x2^3 + 1x2^2 + 1x2^1 + 1x2^0 \\
 &= 64 + 32 + 16 + 8 + 4 + 2 + 1 \\
 &= 127
 \end{aligned}$$

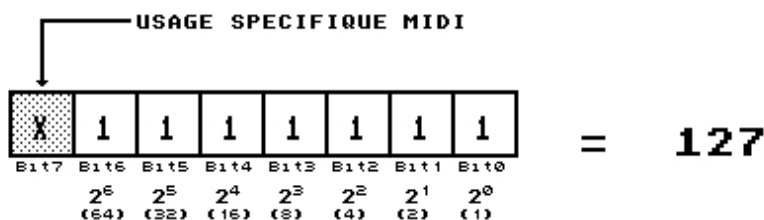


Figure 1.1 : L'octet MIDI n'utilise que les 7 bits de droite pour représenter une valeur. Le bit le plus à gauche est réservé à un usage spécifique.

De droite à gauche, les bits d'un octet sont numérotés de 0 à 7. Par convention, le bit 0 se nomme bit de poids faible, ou LSB (Least Significant Bit), car il correspond à la puissance de 2 la plus petite. Par opposition, le bit 7 se nomme bit de poids fort, ou MSB (Most Significant Bit). De même, lorsqu'une donnée est représentée par plusieurs octets, celui de droite est appelé LSB (Least Significant Byte), et celui de gauche MSB (Most Significant Byte). L'interprétation des termes LSB et MSB prête à confusion, et dépendra donc uniquement du contexte.

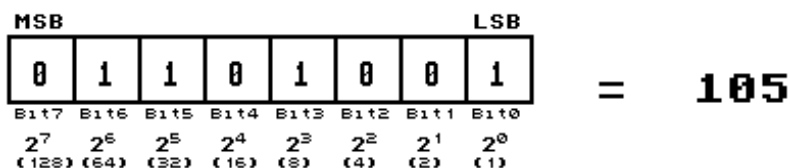


Figure 1.2 : Dans un octet, les bits sont numérotés de 0 (LSB) à 7 (MSB).

La réunion de deux octets (MSB, LSB) représente un nombre de 16 bits pour un total de 65 536 valeurs possibles (de 0 à 65 535) :

$$\begin{aligned}
 11111111 \ 11111111 &= 1x2^{15} + 1x2^{14} + 1x2^{13} + 1x2^{12} + 1x2^{11} \\
 &\quad + 1x2^{10} + 1x2^9 + 1x2^8 + 1x2^7 + 1x2^6 + 1x2^5 \\
 &\quad + 1x2^4 + 1x2^3 + 1x2^2 + 1x2^1 + 1x2^0 \\
 &= 32\ 768 + 16\ 384 + 8\ 192 + 4\ 096 + 2\ 048 \\
 &\quad + 1\ 024 + 512 + 256 + 128 + 64 + 32 + 16 \\
 &\quad + 8 + 4 + 2 + 1 \\
 &= 65\ 535
 \end{aligned}$$

Attention, sur deux octets MIDI, dont seuls les bits 0 à 6 sont utiles (symbolisés par les lettres "a" à "n", et les bits 7 par X), la conversion s'opère comme suit :

Xabcdefg Xhijklmn

qui équivaut à :

abcdef ghijklmn (14 bits utiles)

En effet, les deux bits 7 n'ont pas de poids binaire en tant que tel, car ils ne sont pas utilisés pour représenter une valeur. Il est nécessaire d'opérer une "réduction" sur 14 bits, puisque en réalité le bit 0 du MSB se retrouve à la place du bit 7 du LSB, tandis que les bits 1 à 6 de ce même MSB sont décalés d'un bit vers la droite. On obtiendra donc une représentation de 16 384 valeurs (de 0 à 16 383) :

$$\begin{aligned}
 11111111 \ 11111111 &= 1 \times 2^{13} + 1 \times 2^{12} + 1 \times 2^{11} + 1 \times 2^{10} + 1 \times 2^9 + 1 \times 2^8 + 1 \times 2^7 \\
 &\quad + 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\
 &= 8\ 192 + 4\ 096 + 2\ 048 + 1\ 024 + 512 + 256 + 128 + 64 \\
 &\quad + 32 + 16 + 8 + 4 + 2 + 1 \\
 &= 16\ 383
 \end{aligned}$$

128 et 16 384 sont des nombres que l'on rencontrera souvent, lors de l'étude du langage MIDI. Il est donc indispensable de bien assimiler leur origine.

Il est fréquent de ne traiter qu'un demi-octet, ou groupe de 4 bits : c'est le *nybble* (ou nibble). Un nybble représente 16 valeurs possibles (de 0 à 15) :

$$\begin{aligned}
 1111 &= 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\
 &= 8 + 4 + 2 + 1 \\
 &= 15
 \end{aligned}$$

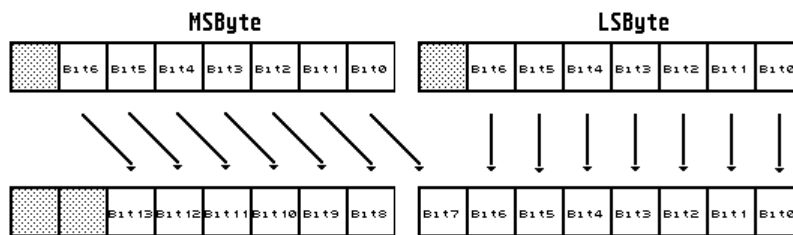


Figure 1.3 : Le principe de conversion de deux octets MIDI en un nombre de 14 bits.

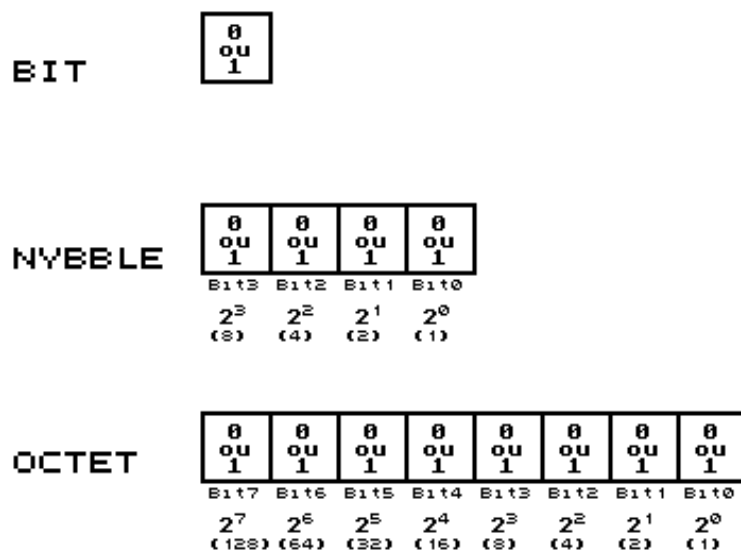


Figure 1.4 : L'octet est constitué de 8 bits, le nybble (demi-octet) est constitué de 4 bits.