Description des instructions du BASIC STAMP II

A partir de la page 3, vous trouverez la liste des instructions du
Basic Stamp 2 et du Basic Stamp 2sx, classées par ordre alphabétique.

Liste des Instructions du BASIC STAMP 2 et 2sx
Classées par type d'instructions

A) La structure d'enchaînement

B) La structure de décision

Instructions de branchement

C) La structure de boucle.

Instructions de boucle

N.B. : Les boucles vues en programmation structurées peuvent-être réalisées avec les instructions IF...THEN et GOTO.

Conventions :

Lors de la présentation de la syntaxe des instructions, les indications de Parallax ont été respectées.
Les parenthèses ( ) et les crochets [ ] doivent être utilisés tel-quels. Par contre, et comme il est d'usage dans quasiment tous les manuels, les accolades {} permettent de repérer les éléments dont la présence est facultative. Elles ne doivent jamais être reproduites dans le programme.
Les termes ports et pattes sont utilisés indifféremment lors de la présentation des instructions, étant entendu que ceux-ci ne correspondent en aucun cas au brochage réel des circuits mais bien à l'appellation normalisée des ports. Ainsi lorsqu'il est fait état de la patte 1, il faut comprendre port 1 ou port P1, soit physiquement la pin n°5 du circuit stamp2 ou stamp2sx.

BRANCH
BRANCH index, [adresse 0, adresse 1, ..., adresse N]
Branche à l'adresse spécifiée par l'index.

- Index est une constante ou une variable ou une expression, de type bit, nibble octet ou mot qui précise quelle adresse va être utilisée (0 à N).

- Adresse X est une étiquette définissant une adresse.

L'instruction fonctionne de la façon suivante : si l'index vaut 0, la première adresse de la liste est utilisée ; si l'index vaut 1, la deuxième adresse est utilisée et ainsi de suite. Dans le cas où la valeur de l'index est supérieure au nombre d'adresses proposées dans la liste, l'instruction da aucun effet et le programme continue en séquence.

BUTTON
BUTTON patte, étatbas, délai, vitesse, variable, étatcible, adresse
Lit l'état d'un poussoir et effectue un anti-rebond, l'auto-répétition, le branchement à une adresse si le poussoir est dans un état stable.
Le poussoir peut-être actif haut ou actif bas.

- Patte est une constante, une variable ou une expression, de type bit, nibble, octet ou mot, comprise entre 0 et 15. Elle indique le numéro de port d'entrée/sortie à utiliser. Cette patte est placée en entrée par l'instruction et y reste quel qu'ait pu être son état préalable.

- Etatbas est une constante ou une variable logique (0 ou 1) ou une expression, de type bit, nibble, octet ou mot. Elle précise l'état logique du poussoir lorsqu'il est appuyé.

- Délai est une constante ou une variable ou une expression, de type bit, nibble octet ou mot, comprise entre 0 et 255. Elle indique pendant combien de temps le poussoir doit être maintenu pour générer une fonction de répétition automatique. Si délai vaut 0, les fonctions d'anti-rebond et de répétition automatique sont inhibées. Si délai vaut 255, l'anti-rebond a lieu mais pas la répétition automatique.

- Vitesse est une constante ou une variable ou une expression, de type bit, nibble, octet ou mot, comprise entre 0 et 255. Elle précise la vitesse de la répétition automatique. Cette vitesse est exprimée en nombre de cycles de l' instruction BUTTON.

- Variable est la variable de travail de BUTTON, de type octet ou mot. Elle doit être mise à zéro avant la première utilisation de cette instruction et ne doit pas être utilisée ailleurs dans le programme, même par une autre instruction BUTTON, au risque d'obtenir des résultats imprévisibles.

- Etatcible est une constante, une variable logique ou une expression, de type bit, nibble, octet ou mot. Elle précise dans quel état doit être le poussoir pour que le branchement ait lieu (0 = non appuyé, 1 = appuyé).

- Adresse est une étiquette qui précise l'adresse où va brancher le programme lorsque le poussoir sera dans l'état défini par étatcible.

Cette instruction est assez complexe à utiliser correctement.

COUNT
COUNT patte, période, variable
Compte le nombre de cycles (0-1-0 ou 1-0-1) sur la patte spécifiée pendant une durée indiquée par période et place le résultat dans variable.

- Patte est une constante, une variable ou une expression, de type bit, nibble, octet ou mot, comprise entre 0 et 15, qui indique le numéro de port d'entrée/ sortie à utiliser. Cette patte est placée en entrée par l'instruction et y reste quel qu'ait pu être son état préalable.

- Période est une constante, une variable ou une expression, de type bit, nibble octet ou mot, comprise entre 1 et 65535, qui indique le nombre de millisecondes (Stamp2) ou de multiples de 400 µs (Stamp2SX) pendant lequel le comptage va avoir lieu.

- Variable est une variable, généralement de type mot, dans laquelle est placé le résultat. Si une plage de variation de cette variable de 0 à 255 suffit, une variable de type octet peut être utilisée. Un cycle est considéré par cette instruction comme la succession de changements d'états: 1-0-1 ou encore O-1-0. Le signal appliqué à la patte utilisée doit avoir des états hauts et bas d'une durée au moins égale à 4 µs, ce qui limite la fréquence maximale utilisable à 125 kHz en supposant un rapport cyclique de 50 %. Dans le cas d'un signal non symétrique, la durée du plus court des états, haut ou bas peu importe, doit être au moins égale à 4 µs.

DATA
DATA Etiquette
DATA constante, constante, ... Etiquette
DATA @adresse, constante, constante, ... Etiquette
DATA (nombre) Etiquette
DATA constante (nombre) Etiquette
DATA WORD constante
Place des constantes dans l'EEPROM avant d'y charger le programme BASIC.

Cette instruction est utile pour initialiser l'EEPROM avec des données qui seront ensuite utilisées par le programme. Les données sont stockées à partir des adresses basses de la mémoire. Comme le programme est stocké à partir des adresses hautes, un recouvrement peut être à craindre. Si tel est le cas, il est détecté lors du téléchargement du programme par le logiciel de développement qui le signale. Les constantes sont placées en EEPROM à l'adresse indiquée par un pointeur qui est initialisé par défaut à 0 et qui augmente d'une unité à chaque donnée mémorisée. L'étiquette qui précède l'instruction se voit allouer la valeur initiale de ce pointeur lors de l'exécution de l'instruction.

- Ainsi, dans la syntaxe n' 1, le pointeur part de 0 et Etiquette vaut donc 0.

- La syntaxe n' 2 permet de modifier la valeur de départ du pointeur qui est alors rendue égale à la valeur qui suit le caractère @.

- La syntaxe n' 3 permet de réserver un nombre d'octets égal à nombre mais sans y stocker aucune donnée. Cela revient en fait à augmenter le pointeur d'une valeur égale à nombre. Attention, comme aucune donnée n'est effectivement mémorisée par cette instruction dans cette situation, le contenu de l'EEPROM est alors quelconque si l'on part d'une EEPROM vierge, ou correspond à ce qui y avait été placé précédemment.

- La syntaxe n' 4 permet de réserver un nombre d'octets égal à nombre tout en les initialisant à la valeur indiquée par constante.

- La syntaxe n' 5 enfin permet de couper en deux la constante, qui doit être un mot de 16 bits, afin de placer l'octet de poids fort à l'adresse n et l'octet de poids faible à l'adresse n + 1.

Voiçi quelques exemples :

```
liste1    DATA      12,45,38,29     ' liste1 vaut 0, la valeur 12 est stockée à
                                    ' l'adresse 0, la valeur 45 est stockée à
                                    ' l'adresse 1, 38 est mis en 2 et 29 en 3

liste2    DATA      25,85,54        ' Le pointeur d'adresse vaut donc 4 et ce
                                    ' sera la valeur de liste2. Les valeurs 25,85
                                    ' et 54 seront mises aux adresses 4,5 et 6.

liste3    DATA      @10,78          ' Le pointeur d'adresse est mis à 10 et la
                                    ' valeur 78 y sera rangée.

liste4    DATA      16(22)          ' Liste4 vaut 11 (10+1). Les 22 octets
                                    ' suivants seront initialisés a la valeur 16.

liste5    DATA      WORD 1000       ' liste 5 vaut maintenant 33 (11+22). La
                                    ' valeur 1000 est décomposées en 2 valeurs
                                    ' "octets" soit 3 et 232 (3*256+232=1000).
                                    ' La valeur 3 sera stockée à l'adresse 33 et
                                    ' la valeur 232 sera stockée à l'adresse 34.
```

DEBUG
DEBUG variable 1 {, variable, ...}

C'est une instruction de mise au point de programmes qui envoie le contenu des variables au PC connecté au Stamp afin de les visualiser. DEBUG fonctionne un peu comme le PRINT de la majorité des BASIC mais, au lieu de faire imprimer ce qui suit, elle le fait afficher sur l'écran du PC en fonction des instructions de formatage qui peuvent être données après la ou les variables indiquées.

DEBUG peut être suivi de plusieurs variables et chaînes de caractères, séparées par des virgules. Les chaines de caractères sont affichées telles quelles, les variables sont formatées conformément à un certain nombre de préfixes. Par défaut, toute variable, dont le format numérique n'est pas explicitement précisé au moyen des préfixes présentés ci-dessous, est affichée sous la forme du caractère ASCII correspondant. Ainsi, si la variable x vaut 65 en décimal DEBUG x fait afficher la lettre A puisque son code ASCII est égal à 65.

- Un point d'interrogation (?) devant le nom d'une variable fait afficher "variable = valeur" sur l'écran du PC. Si ce point d'interrogation est utilisé conjointement à un préfixe, il doit être placé entre le préfixe et le nom de la variable. On doit ainsi écrire, par exemple, DEBUG dec ? x pour faire afficher
  "x = valeur décimale de x".

- Les différents éléments à afficher doivent être séparés par des virgules.

- DEBUG peut effectuer des calculs sur des variables sous réserve de respecter la syntaxe générale des expressions arithmétiques et logiques. Ainsi, si la variable x vaut 24, DEBUG dec 3 * (x + 1) fait afficher "3 * (x + 1) = 75".

- REP variable\n fait afficher la valeur de la variable n fois.

- DEC variable fait afficher la valeur de la variable en décimal.

- DEC1-DEC5 variable fait afficher la valeur de la variable en décimal avec 1 à 5 chiffres.

- SDEC variable fait afficher la valeur de la variable en décimal signé.

- SDEC1-SDEC5 variable fait afficher la valeur de la variable en décimal signé avec 1 à 5 chiffres.

- HEX variable fait afficher la valeur de la variable en hexadécimal.

- HEX1-HEX4 variable fait afficher la valeur de la variable en hexadécimal avec 1 à 4 chiffres.

- SHEX variable fait afficher la valeur de la variable en hexadécimal signé.

- SHEX1-SHEX4 variable fait afficher la valeur de la variable en hexadécimal signé avec 1 à 4 chiffres.

- IHEX variable fait afficher la valeur de la variable en hexadécimal, précédée du symbole $.

- IHEX1-IHEX4 variable fait afficher la valeur de la variable en hexadécimal, précédée du symbole $ et avec 1 à 4 chiffres.

- ISHEX variable fait afficher la valéur de la variable en hexadécimal signé, précédée du symbole $.

DEBUG (suite)

- ISHEX1-ISHEX4 variable fait afficher la valeur de la variable en hexadécimal signé, précédée du symbole $ et avec 1 à 4 chiffres.

- BIN variable fait afficher la valeur de la variable en binaire.

- BIN1-BIN16 variable fait afficher la valeur de la variable en binaire avec 1 à 16 chiffres.

- SBIN variable fait afficher la valeur de la variable en binaire signé.

- SBIN1-SBIN16 variable afficher la valeur de la variable en binaire signé avec 1 à 16 chiffres.

- IBIN variable fait afficher la valeur de la variable en binaire, précédée du symbole %.

- IBIN1-IBIN16 variable fait afficher la valeur de la variable en binaire, précédée du symbole % et avec 1 à 16 chiffres.

- ISBIN variable fait afficher la valeur de la variable en binaire signé, précédée du symbole %.

- ISBIN1-ISBIN16 variable fait afficher la valeur de la variable en binaire signé, précédée du symbole % et avec 1 à 16 chiffres.

- STR variable tableau d'octet fait afficher la chdine de caractères dont les codes ASCII correspondent au contenu du tableau pris dans son intégralité.

- STR variable tableau d'octets\n fait afficher la chaîne de caractères dont les codes ASCII correspondent aux n octets pris dans le tableau.

DEBUG peut également être suivi par un certain nombre de directives de formatage interprétées par le PC; ce sont :

- CLS qui efface l'écran et positionne le curseur en haut à gauche;
- HOME qui positionne le curseur en haut à gauche de l'écran mais sans l'effacer
- BELL qui génère un « bip » sur le haut-parleur du PC;
- BKSP qui déplace le curseur d'une position vers la gauche;
- TAB qui déplace le curseur de huit colonnes vers la droite;
- CR qui génère un retour chariot suivi d'un saut de ligne.

DTMFOUT
DTMFOUT patte, {on,off,}[touche, touche, ...]
Génère les tonalités de numérotation téléphonique DTMF sur la patte spécifiée.

- Patte est une constante, une variable ou une expression, de type bit, nibble, octet ou mot, comprise entre 0 et 15, qui indique le numéro de port d'entrée/sortie à utiliser. Cette patte est placée en sortie pour la durée de l'instruction, et revient en entrée à la fin de l'instruction, quel qu'ait pu être son état préalable.

- On est optionnel et permet de spécifier la durée de génération de la tonalité choisie, exprimée en millisecondes (Stamp2) ou en multiples de 400 µs (Stamp2SX). C'est une constante, une variable ou une expression, de type bit, nibble, octet ou mot, comprise entre 0 et 65535. Sa valeur par défaut fait générer un temps de 200 ms.

- Off est optionnel et spécifie la durée de la pause après la tonalité générée, ou entre les différentes tonalités si elles sont plusieurs. Elle est exprimée en millisecondes (Stamp2) ou en multiples de 400 µs (Stamp2SX). C'est une constante, une variable ou une expression, de type bit, nibble, octet ou mot, comprise entre 0 et 65535 et sa valeur par défaut fait générer un délai de 50 ms.

- Touche est une variable, une constante ou une expression, de type bit, nibble, octet ou mot, comprise entre 0 et 15, qui indique la tonalité DTMF à générer conformément aux affectations des claviers de téléphone. Les chiffres de 0 à 9 sont représentés par la valeur correspondante, l'étoile (*) par 10, le dièse (#) par 11 et les lettres A à D par les valeurs 12 à 15.
  Comme ces tonalités sont générées de façon purement numérique et ne sont pas accessibles via un quelconque convertisseur digital/analogique, il est nécessaire de les filtrer avant de les utiliser, sur le réseau téléphonique ou dans toute autre application.

END
END
Fait passer le Stamp en mode sommeil pour une durée indéfinie.

Le retour au fonctionnement normal n'aura lieu que lors de la détection d'une demande de connexion de la part du PC ou suite à un reset matériel.

Pendant ce mode, les entrées et sorties du Stamp conservent leur état. De ce fait, si une sortie commandait une charge avant de rencontrer l'instruction END, elle continuera à le faire, même pendant la durée du mode sommeil provoqué par cette instruction. Par contre, en raison du mode de fonctionnement interne du Stamp, les sorties concernées passent en état haute impédance pendant 18 ms toutes les 2,3 s environ. Il importe donc d'en tenir compte si cela peut avoir une influence sur la ou les charges qui y sont connectées.

FOR ... NEXT
FOR variable = debut TO fin {STEP pas}
... instructions ...
NEXT

Réalise une boucle FOR - NEXT classique. La variable est rendue égale à la valeur début puis le code compris entre FOR et NEXT est exécuté. La variable est alors incrémentée de 1 (ou du pas spécifié par la directive optionnelle STEP) et si elle n'est pas égale ou supérieure à fin, les instructions comprises entre FOR et NEXT sont exécutées à nouveau, et ainsi de suite. Lorsque la variable devient égale ou supérieure à fin, le programme continue en séquence après NEXT.

Quelles que soient les valeurs de début et de fin, la boucle est toujours exécutée au moins une fois.

Un programme peut comporter autant de boucles FOR - NEXT qu'on le souhaite mais il ne faut pas imbriquer les unes dans les autres plus de 16 boucles.

- Variable est une variable de type bit, nibble, octet ou mot utilisé comme compteur de boucle. Début et fin sont limités par la capacité en taille de cette variable (0 à 1, 0 à 15, 0 à 255 ou 0 à 65535).

- Début est une constante, une variable ou une expression, de type bit, nibble, octet ou mot, qui spécifie la valeur de début adoptée par la variable de boucle.

- Fin est une constante, une variable ou une expression, de type bit, nibble, octet ou mot, qui spécifie la valeur de fin que devra atteindre ou dépasser (si l'égalité n'est pas possible) la variable de boucle.

- Pas est une constante, une variable ou une expression, optionnelle, de type bit, nibble, octet ou mot. Elle précise de combien doit être incrémentée ou décrémentée la variable de boucle. En son absence, le pas est fixé à + 1.Il ne faut pas préciser le signe de pas qui est déterminé automatiquement par l'interpréteur, suite à la comparaison des valeurs extrêmes de la variable de boucle.

- La présence du nom de la variable de boucle après le NEXT est interdite.

FREQOUT
FREQOUT patte, durée, fréq1 {,fréq2}
Génère une ou deux sinusoïdes de fréquence(s) choisies(s) pendant une durée sélectionnée sur la patte d'entrée/sortie spécifiée.

- Patte est une constante, une variable ou une expression, de type bit, nibble, octet ou mot, comprise entre 0 et 15, qui indique le numéro de port d'entrée/sortie à utiliser. Cette patte est placée en sortie dès le début de l'instruction et elle y reste ensuite quel qu'ait pu être son état préalable.

- Durée est une constante, une variable ou une expression, de type bit, nibble, octet ou mot, comprise entre 1 et 65535, qui indique la durée de génération du signal en millisecondes (Stamp2) ou en multiples de 400 µs (Stamp2SX).

- Fréq1 est une constante, une variable ou une expression, de type bit, nibble, octet ou mot, comprise entre 0 et 32767, qui indique la première fréquence à générer en hertz (Stamp2) ou en multiples de 2,5 Hz (Stamp2SX).

- Fréq2 est une constante, une variable ou une expression, optionnelle, de type bit, nibble, octet ou mot, comprise également entre 0 et 32767, qui indique la deuxième fréquence à générer en hertz (Stamp2) ou en multiples de 2,5 Hz (Stamp2SX). Comme ces fréquences sont générées de façon purement numérique et ne sont pas accessibles via un quelconque convertisseur digital/analogique, il peut être nécessaire de les filtrer avant de les utiliser.

GET
GET adresse, variable
Lit une donnée à l'adresse spécifiée dans la mémoire bloc-notes.

Cette instruction est spécifique du Stamp2SX

- Adresse est une constante ou une variable, comprise entre 0 et 63, qui indique l'adresse de la mémoire bloc-notes qui doit être lue

- Variable contient la donnée lue à la fin de l'exécution de l'instruction.

GOSUB
GOSUB adresse
Exécute un saut au sous-programme dont l'adresse est spécifiée.

Lorsque le sous- programme est terminé, l'instruction RETURN (voir ce mot) fait poursuivre l'exécution à l'instruction qui suit GOSUB.

Il est possible d'utiliser jusquà 255 GOSUB dans un seul et même programme.

Il ne faut pas imbriquer plus de quatre GOSUB.

- Adresse est une étiquette qui précise l'adresse de début du sous-programme.

GOTO
GOTO adresse
Exécute un saut à l'adresse spécifiée et fait poursuivre l'exécution du programme à cette adresse.

Une utilisation courante de GOTO est la réalisation de boucles de programmes sans fin, en renvoyant l'exécution de la fin sur le début.

- Adresse est une étiquette qui précise l'adresse où poursuivre l'exécution du programme.

HIGH
HIGH patte
Fait passer la patte spécifiée au niveau logique haut.

Si la patte a été préalablement définie comme étant une entrée, elle est automatiquement mise en sortie par cette instruction qui modifie en conséquence le bit correspondant du registre DIRS.

- Patte est une constante, une variable, une expression, de type bit, nibble, octet ou mot, comprise entre 0 et 15. Elle indique le numéro de port d'entrée/sortie à utiliser.

IF ... THEN
IF expression conditionnelle THEN étiquette
Evalue une expression conditionnelle et effectue un branchement à l'adresse spécifiée si le résultat de l'expression est vrai.

Contrairement à ce qui est permis avec certains Basic, l'étiquette qui suit le THEN ne peut pas être remplacée par une instruction.

- ?? est un des opérateurs de relation suivants : =, <>, >, >, >=, <=.

- Expression conditionnelle est une expression conditionnelle valide telle qu' elles ont été définies en début de chapitre. Tous les opérateurs de relation peuvent y être utilisés, de même que les opérateurs arithmétiques.

- Étiquette indique l'adresse à laquelle se continue l'exécution du programme si le résultat de la comparaison est vrai.

INPUT
INPUT patte
Place la patte spécifiée en entrée et met le bit correspondant du registre DIRS à 0.

Attention ! Cette instruction ne lit pas l'état de la patte correspondante.

- Patte est une constante, une variable, une expression, de type bit, nibble, octet ou mot comprise entre 0 et 15. Elle indique le numéro de port d'entrée/sortie à utiliser.

LOOKDOWN
LOOKDOWN donnée,{??,}[valeurO,valeur1,.... valeurN],variable
Compare une donnée avec une liste de valeurs.

Si une égalité est trouvée (ou si la comparaison est vérifiée), cette instruction place le numéro de rang de cette valeur (de 0 à N) dans la variable. Si aucune valeur ne concorde, la variable n'est pas affectée. Le décompte du rang des valeurs commence à zéro. Si, par exemple, la liste des valeurs est: 4, 13, 15, 28, 8, et que la donnée est égale à 15, la variable contiendra 2 car 15 est la troisième valeur de la liste. Par défaut, l'instruction ne recherche que l'égalité de la donnée avec les valeurs contenues dans la liste. Mais il est possible de rechercher d'autres types de conditions (égal, différent, supérieur, inférieur, supérieur ou égal et inférieur ou égal).

- Donnée est une constante ou une variable, une expression dont la présence est comparée dans la liste des valeurs.

- ValeurO à valeurN sont des constantes,variables ou expressions de type bit, nibble, octet ou mot.

- Variable est une variable de type bit, nibble, octet ou mot qui contient le résultat de la recherche ou n'est pas affectée si la recherche est négative.

- ?? est un opérateur relationnel à choisir parmi les suivants <>, >, <, <=, =>. S'il n'est pas précisé, l'égalité est choisie par défaut.

LOOKUP
LOOKUP index, [valeurO, valeurl,.... valeurN], variable
Recherche la valeur spécifiée par l'index dans la liste des valeurs et place celle-ci dans la variable.

Comme pour l'instruction LOOKDOWN, l'index commence son décompte à zéro. Si, par exemple, la liste des valeurs est 2, 13, 15, 28 et 8 et que l'index vaut 1, la valeur placée dans la variable sera 13. Si l'index est plus grand que le nombre de valeurs disponibles, la variable n'est pas affectée.

- Index est une constante ou une variable de type bit, nibble, octet ou mot, qui spécifie la position de la valeur dans la liste qui doit être placée dans la variable.

- ValeurO à valeurN sont des constantes, variables ou expressions de type bit, nibble, octet ou mot.

- Variable est une variable de type bit, nibble, octet ou mot qui contient la valeur sélectionnée par l'index ou n'est pas affectée si l'index est trop grand.

LOW
LOW patte
Fait passer la patte spécifiée au niveau logique bas.

Si la patte a été préalablement définie comme étant une entrée, elle est automatiquement mise en sortie par cette instruction qui modifie en conséquence le bit correspondant du registre DIRS.

- Patte est une constante, une variable ou une expression de type bit, nibble, octet ou mot, comprise entre 0 et 15. Elle indique le numéro de port d'entrée/sortie à utiliser.

NAP
NAP periode
Place le Stamp en mode sommeil pour une courte durée.

Dans ce mode, la consommation est de 100 µA (Stamp2) et 200 µA (Stamp2SX), hors consommation de la circuiterie d'entrées/sorties.

- Période est une constante, une variable ou une expression de type bit, nibble, octet ou mot, comprise entre 0 et 7, qui spécifie la durée de ce mode selon la relation approximative suivante : durée = 18 X 2^periode (en millisecondes)

Compte tenu du fait que période est comprise entre 0 et 7, la plage de durée permise varie de 18 ms à 2,3 s environ.

Pendant ce mode, les entrées et sorties du Stamp conservent leur état. De ce fait, si une sortie commandait une charge avant de rencontrer l'instruction NAP, elle continuera à le faire même pendant la durée du mode sommeil provoqué par cette instruction. Par contre, en raison du mode de fonctionnement interne du Stamp, toutes les lignes d'entrées/sorties passeront en entrée pendant 18 ms lors de la sortie du mode NAP. Il importe donc d'en tenir compte si cela peut avoir une influence sur la ou les charges qui y sont connectées.

Notez en outre que le décompte du temps réalisé par cette instruction utilise le timer chien de garde contenu dans le Stamp sans compensation de temps. La durée effective de l'instruction NAP subit des variations relativement importantes en fonction de la température et de la tension d'alimentation. Dans le pire des cas des variations de - 50 à + 100 % sont envisageables. Ainsi, un NAP 0 peut voir sa durée varier de 9 à 36 ms. A température ambiante et avec une alimentation stabilisée, cette variation peut cependant être estimée à ± 10 % seulement.

Pour des durées relativement longues ou plus précises, l'instruction SLEEP est à préférer car elle utilise un processus de compensation des variations de fréquence du timer chien de garde interne et atteint alors une précision de l' ordre de 1%.

OUTPUT
OUTPUT patte
Place la patte spécifiée en sortie et met le bit correspondant du registre DIRS à 1.

Attention! Cette instruction ne définit pas l'état de la patte correspondante qui prend alors immédiatement celui de la valeur préalablement contenue dans le registre de sortie (OUTS).

- Patte est une constante, une variable ou une expression de type bit, nibble, octet ou mot, comprise entre 0 et 15. Elle indique le numéro de port d'entrée/sortie à utiliser.

PAUSE
PAUSE millisecondes
Suspend l'exécution du programme pendant un certain nombre de millisecondes.

La précision de cette durée ne dépend que de celle de l'oscillateur d'horloge du Stamp. Cependant, il faut noter qu'un peu de temps (typiquement, 1 ms) peut être perdu à exécuter les instructions adjacentes. L'erreur ainsi introduite est d'autant plus négligeable que la pause est longue.

Un timer relativement précis peut être réalisé grâce à cette instruction en effectuant des pauses de « longue » durée, 100 ms par exemple, dans une boucle et en incrémentant un compteur à chaque fois. Tant que le compteur ne déborde pas, la boucle contenant la pause se répète.

- Millisecondes est une constante, une variable ou une expression, de type bit, nibble, octet ou mot, comprise entre 0 et 65535, qui spécifie la durée de la pause.

PULSIN
PULSIN patte, état, variable
Mesure la durée d'une impulsion d'entrée avec une résolution de 2 µs (Stamp2) et 800 ns (Stamp2SX).

- Patte est une constante, une variable ou une expression, de type bit, nibble, octet ou mot, comprise entre 0 et 15, spécifiant le numéro de port d'entrée/ sortie à utiliser. Cette patte est placée en entrée dès le début de l' instruction et elle y reste ensuite quel qu'ait pu être son état antérieur.

- État est une constante, une variable ou une expression, de type bit, nibble, octet ou mot, égale à 0 ou 1, qui indique si l'impulsion à mesurer commence par un front montant (1) ou par un front descendant (0).

- Variable est une variable de type bit, nibble, octet ou mot, utilisée pour stocker le résultat de la mesure compris entre 1 et 65536. Si la variable est de taille insuffisante pour le résultat de la mesure, elle contiendra un résultat faux (en principe, les bits de poids faibles du compteur interne utilisé pour réaliser cette mesure).

PULSOUT
PULSOUT patte, temps
Génère une impulsion en inversant le niveau d'une patte pendant le laps de temps spécifié.

- Patte est une constante, une variable ou une expression, de type bit, nibble, octet ou mot, comprise entre 0 et 15, spécifiant le numéro de port d'entrée/ sortie à utiliser. Elle est placée en sortie, dès le début de l'instruction, et elle y reste ensuite quel qdait pu être son état antérieur.

- Temps est une constante, une variable ou une expression, de type bit, nibble, octet ou mot comprise entre 0 et 65535, qui spécifie la durée de l'impulsion par pas de 2 µs (Stamp2) et de 800 ns (Stamp2SX).

PUT
PUT adresse, variable
Écrit une donnée à l'adresse spécifiée dans la mémoire bloc-notes.

Cette instruction est spécifique du Stamp2SX.

- Adresse est une constante ou une variable, comprise entre 0 et 62, qui indique l'adresse de la mémoire bloc-notes où doit être écrite la donnée contenue dans la variable.

- Variable contient la donnée à écrire en mémoire.

Notez que toute la mémoire bloc-notes est initialisée à zéro lors d'une mise sous tension ou d'un reset du Stamp. Par ailleurs, la mémoire d'adresse 63 contient le numéro du programme en cours d'exécution (compris entre 0 et 7). Enfin, les adresses de la mémoire bloc-notes « tournent en rond » modulo 63. Ainsi, l'adresse 64 est-elle l'adresse 0, l'adresse 65 est l'adresse 1 et ainsi de suite.

PWM
PWM patte, rapport cyclique, cycles
Génère un signal impulsionnel modulé en durée sur la patte spécifiée et place ensuite celle-ci en entrée.

Cette instruction peut être utilisée pour produire facilement une tension analogique grâce à une résistance et un condensateur. Compte tenu de la décharge inévitable du condensateur au fil du temps, l'instruction doit être répétée régulièrement pour mettre à jour ou rafraîchir cette charge.

- Patte est une constante, une variable ou une expression, de type bit, nibble, octet ou mot comprise entre 0 et 15, spécifiant le numéro de port d'entrée/ sortie à utiliser.

- Rapport cyclique est une constante, une variable ou une expression, de type bit, nibble, octet ou mot, comprise entre 0 et 255, qui précise le niveau analogique désiré de 0 à 5 V. 0 correspond à une tension nulle et 255 à une tension de 5 V; la progression entre ces deux valeurs est linéaire.

- Cycles est une constante, une variable ou une expression, de type bit, nibble octet ou mot, comprise entre 0 et 255, qui spécifie la durée du signal à générer en multiples de 1 ms (Stamp2) et 400 µs (Stamp2SX).

REMARQUE Toute consommation de courant éventuelle sur le condensateur a pour effet de le décharger prématurément. Il est donc conseillé de le faire suivre par un montage à haute impédance tel qu'un suiveur à amplificateur opération-nel, par exemple.

RANDOM
RANDOM variable
Génère un nombre pseudo-aléatoire et place celui-ci dans variable.

Le Stamp utilise une suite de 65536 nombres pseudoaléatoires pour exécuter cette instruction. Lors de cette exécution, la valeur initialement contenue dans la variable est utilisée pour choisir un de ces nombres. De ce fait, si la même valeur de départ est utilisée à chaque fois, la même séquence de nombres pseudo-aléatoires sera générée. Pour obtenir une séquence vraiment aléatoire, il faut ajouter un élément incertain au processus comme, par exemple, lire une entrée série ou l'horloge temps réel et s'en servir comme valeur initiale pour cette instruction.

- Variable est une variable de type octet ou mot, utilisée à la fois comme variable de travail et pour récupérer le nombre pseudoaléatoire généré par l' instruction.

RCTIME
RCTIME patte, état, variable
Mesure le temps pendant lequel une patte reste dans un état logique déterminé.

En pratique, cette instruction est utilisée pour mesurer le temps de charge ou de décharge d'un condensateur.

- Patte est une constante, une variable ou une expression, de type bit, nibble, octet ou mot, comprise entre 0 et 15, spécifiant le numéro de port d'entrée/sortie à utiliser. Cette patte est placée en entrée par l'instruction et y reste ensuite quel qu'ait pu être son état préalable.

- État est une constante, une variable ou une expression, de type bit, nibble, octet ou mot, égale à 0 ou à 1, qui indique l'état logique dans lequel doit être la patte pendant la mesure.

- Variable est une variable de type bit, nibble, octet ou mot qui contient le résultat de la mesure. La résolution est de 2 µs (Stamp2) ou de 800 ns (Stamp2SX) et la variable peut évoluer de 0 à 65535. Si un débordement se produit, elle est mise à zéro. De même, si l'entrée choisie est déjà dans un autre état que celui spécifié lors de l'exécution de l'instruction, la variable est rendue égale à 1 (soit 2 µs en Stamp2 et 800 ns en Stamp 2SX) car il faut au moins un cycle de mesure au Stamp pour s'en rendre compte. Cette instruction est utile pour mesurer le temps de charge ou décharge d'un condensateur connecté.

READ
READ adresse, variable
Lit la mémoire EEPROM à l'adresse spécifiée et place la valeur lue dans la variable.

- Adresse est une constante, une variable ou une expression, de type bit, nibble octet ou mot, comprise entre 0 et 2047. Elle spécifie l'adresse à lire.

- Variable est une variable de type bit, nibble, octet ou mot qui reçoit la valeur lue dans la mémoire EEPROM.

Notez que l'EEPROM est utilisée à la fois pour les données et pour le programme. Le programme est toujours placé en descendant à partir de l'adresse la plus haute (2047), tandis que les données sont placées en montant à partir de l' adresse la plus basse (0).

RETURN
RETURN
Retour de sous-programme.

Fait poursuivre l'exécution du programme à l'instruction qui suit immédiatement le GOSUB ayant appelé le sous-programme correspondant.

RETURN n'utilise aucun paramètre.

REVERSE
REVERSE patte
Inverse le sens de fonctionnement de la patte spécifiée et modifie en conséquence le bit du registre DIRS correspondant.

Si la patte était entrée, elle devient sortie et vice versa.

- Patte est une constante, une variable ou une expression, de type bit, nibble, octet ou mot, comprise entre 0 et 15, spécifiant le numéro de port d'entrée/sortie à utiliser.

RUN
RUN numéro
Cette instruction permet de lancer l'exécution d'un des huit programmes que peut contenir la mémoire du Stamp2SX.

Cette mémoire de 16 Ko est en effet découpée artificiellement par l'interpréteur en huit blocs de 2 Ko, chacun étant affecté à un programme repéré par un numéro évoluant de 0 à 7. Numéro est une constante ou une variable, comprise entre 0 et 7, qui spécifie le numéro de programme à exécuter.

A la mise sous tension ou suite à un reset, le Stamp exécute par défaut le premier programme ou programme 0. Lorsqu'il rencontre une instruction RUN, il poursuit l'exécution avec le programme dont le numéro est spécifié.

Lors du passage d'un programme à un autre, les entrées/sorties gardent leurs états et niveaux; les variables en mémoire RAM de travail et en mémoire bloc-notes ne sont pas affectées. Ceci permet de passer facilement des paramètres entre programmes. Il importe cependant de définir correctement, et dans le même ordre, les noms de variables dans les différents programmes. En effet, comme l' interpréteur procède seul à l'affectation des variables en mémoire, des décalages entre variables peuvent se produire si leur ordre de définition n'est pas respecté.

SERIN
SERIN patte r {\patte f},mode, {errp,}{timeout,etiqt,}[données]
Initialise une liaison série sur l'une des pattes du Stamp et configure cette liaison.

Si plusieurs variables sont indiquées, elles se verront affecter les données successives reçues.

- Patte r est une constante, une variable ou une expression, de type bit, nibble octet ou mot, comprise entre 0 et 16, spécifiant le numéro de port d'entrée/ sortie à utiliser pour la réception des données. Elle peut prendre la valeur de 0 à 15 pour un port normal et recevoir alors des niveaux TTL ou prendre la valeur 16 et recevoir des pseudo-niveaux RS 232 via l'entrée RX (patte 2) du Stamp2 ou Stamp2SX. Quelle que soit la patte choisie, elle est placée en entrée par l'instruction et y reste ensuite quel qu'ait pu être son état préalable.

- Patte f est une constante, une variable ou une expression optionnelle, de type bit, nibble, octet ou mot, comprise entre 0 et 15, spécifiant le numéro de port d'entrée/sortie à utiliser pour le protocole de dialogue éventuel. Elle est placée en sortie par l'instruction et y reste ensuite quel qu'ait pu être son état préalable.

- Mode est une constante, une variable ou une expression, de type bit, nibble, octet ou mot, spécifiant le mode de fonctionnement de la liaison série conformément aux indications ci-dessous :

    - les bits 0 à 12 de mode correspondent à la période de l'horloge bit exprimée en microsecondes et diminuée de 20;

    - le bit 13 est à zéro pour une transmission sur 8 bits sans parité et à un pour une transmission sur 7 bits avec parité paire;

    - le bit 14 est à zéro pour une émission de données « vraies » et à un pour une émission de données « inversées »;

    - le bit 15 n'est pas utilisé par SERIN.

      Une formule simple permet de calculer la période bit de la transmission codée sur les bits 1 à 12 :

        période bit = int (1 000 000/vitesse en bauds) - 20

- Errp est une étiquette optionnelle qui indique où doit se brancher le programme en cas d'erreur de parité. Cette étiquette doit uniquement être indiquée si mode sélectionne un fonctionnement avec parité (bit 13 à un).

- Timeout est une constante, une variable ou une expression optionnelle, de type bit, nibble, octet ou mot, comprise entre 0 et 65535, qui indique combien de temps attendre avant de brancher à l'étiquette Etiqt. L'unité de mesure utilisée est la milliseconde.

- Etiqt est une étiquette optionnelle (mais qui doit être présente si timeout est présent) qui indique où doit se poursuivre le programme en cas de timeout, c'est-à-dire d'attente supérieure à celle spécifiée au moyen de la variable de même nom, sans recevoir de caractère.

- Donnée est un ensemble de constantes, d'expressions et de noms de variables, séparées par des virgules et précédées éventuellement par des "formateurs". Les différents "formateurs" utilisables sont indiqués dans le tableau suivant.

SERIN (suite)

Formateur	Fonction
Variable	Place la donnée reçue dans la variable
STR tableau\L{\E}	Reçoit une suite de données placées dans le tableau d'octets de longueur L avec le caractère de fin optionnel E
SKIP L	Reçoit mais ignore L octets
WAITSTR tableau	Attend un tableau d'octets terminé par un zéro
WAITSTR tobleau/L	Attend un tableau d'octets de longueur L
WAIT (val,val,..)	Attend jusqu'à une séquence de six octets
DEC variable	Reçoit une donnée en décimal
DEC1-DECS variable	Reçoit une donnée en décimal de 1 à 5 chiffres
SDEC variable	Reçoit une donnée en décimal signé
SDEC1-SDEC5 var	Reçoit une donnée en décimal signé de 1 à 5 chiffres
HEX variable	Reçoit une donnée en hexadécimal
HEX1-HEX4 variable	Reçoit une donnée en hexadécimal de 1 à 4 chiffres
SHEX variable	Reçoit une donnée en hexadécimal signé
SHEX1-SHEX4 var	Reçoit une donnée en hexadécimal signé de 1 à 4 chiffres
IHEX variable	Reçoit une donnée en hexadécimal précédée de S
IHEX1-IHEX4 var	Reçoit une donnée en hexadécimal de 1 à 4 chiffres, précédée de S
ISHEX variable	Reçoit une donnée en hexadécimal signé précédée de S
ISHEX1-ISHEX4 var	Reçoit une donnée en hexadécimal signé de 1 à 4 chiffres, précédée de S
BIN variable	Reçoit une donnée en binaire
BIN1-BIN16 variable	Reçoit une donnée en binaire de 1 à 16 chiffres
SBIN variable	Reçoit une donnée en binaire signé
SBIN1-SBIN16 var	Reçoit une donnée en binaire signé de 1 à 16 chiffres
IBIN variable	Reçoit une donnée en binaire précédée de %
IBIN1-IBIN16 var	Reçoit une donnée en binaire de 1 à 16 chiffres,précédée de
SBIN variable	Reçoit une donnée en binaire signé précédée de %
ISBIN1-ISBIN16 var	Reçoit une donnée en binaire signé de 1 à 16 chiffres, précédée de %

SEROUT
SEROUT patte t, mode, {tempo,}[donnée] ou
SEROUT patte t {\patte f},mode, {timeout,etiqt,}[donnée]

Initialise une liaison série sur l'une des pattes du Stamp, configure cette liaison et émet un ou plusieurs caractères sous forme série asynchrone.

- Patte t est une constante, une variable ou une expression, de type bit, nibble octet ou mot, comprise entre 0 et 16, spécifiant le numéro de port d'entrée/ sortie à utiliser pour l'émission des données. Elle peut prendre une valeur entre 0 et 15 pour une patte normale et émettre alors des niveaux TTL ou prendre la valeur 16 et émettre des pseudo-niveaux RS 232 via la sortie TX (patte 1) du Stamp2 ou Stamp2SX.

- Patte f est une constante, une variable ou une expression optionnelle, de type bit, nibble, octet ou mot, comprise entre 0 et 15, spécifiant le numéro de port d'entrée/sortie à utiliser pour le protocole de dialogue éventuel. Cette patte est placée en entrée par l'instruction et y reste ensuite quel qu'ait pu être son état préalable.

- Mode est une constante, une variable ou une expression, de type bit, nibble, octet ou mot, spécifiant le mode de fonctionnement de la liaison série conformément aux indications ci-dessous :

    - les bits 0 à 12 de mode correspondent à la période de l'horloge bit exprimée en microsecondes et diminuée de 20;

    - le bit 13 est à zéro pour une transmission sur 8 bits sans parité et à un pour une transmission sur 7 bits avec parité paire;

    - le bit 14 est à zéro pour une émission de données « vraies » et à un pour une émission de données « inversées »;

    - le bit 15 est à zéro pour une sortie normale et à un pour une sortie de type drain ouvert/source ouverte.

      Une formule simple permet de calculer la période bit de la transmission codée sur les bits de 1 à 12 :

          période bit = int (1000 000/vitesse en bauds) - 20

      Si la patte TX (patte 1) est spécifiée pour patte t, les bits 14 et 15 de mode sont ignorés sauf pour patte f

- Tempo est une constante, une variable ou une expression optionnelle, de type bit, nibble, octet ou mot, comprise entre 1 et 65535, qui indique combien de temps attendre entre chaque octet transmis. Le temps est indiqué en millisecondes. Tempo ne doit être employée que si le protocole de dialogue n' est pas utilisé.

- Timeout est une constante, une variable ou une expression optionnelle, de type bit, nibble, octet ou mot, comprise entre 0 et 65535, qui indique combien de temps attendre avant de brancher à l'étiquette Etiqt. L'unité de mesure utilisée est la milliseconde. Le temps d'attente est celui imposé par la patte de dialogue patte f qui doit donc être inférieur à la valeur choisie pour timeout.

- Etiqt est une étiquette optionnelle (mais qui doit être présente si timeout est présent) qui indique où doit se poursuivre le programme en cas de timeout, c'est-à-dire d'attente supérieure à celle spécifiée au moyen de la variable de même nom, sans pouvoir émettre de caractère.

SEROUT (suite)

- Donnée est un ensemble de constantes, d'expressions et de noms de variables, séparées par des virgules et précédées éventuellement par des "formateurs". Les différents "formateurs" utilisables sont indiqués dans le tableau suivant.

Formateur	Fonction
Variable	Emet la donnée contenue dans la variable
STR tableau\n	Emet une suite de n données contenues dans le tableau spécifié
REP variable\n	Emet n fois la donnée spécifiée
DEC variable	Emet la donnée en décimal
DEC1-DEC5 variable	Emet la donnée en décimal de 1 à 5 chiffres
SDEC variable	Emet la donnée en décimal signé
SDEC1-SDEC5 var	Emet la donnée en décimal signé de 1 à 5 chiffres
HEX variable	Emet la donnée en hexadécimal
HEX1-HEX4 variable	Emet la donnée en hexadécimal de 1 à 4 chiffres
SHEX variable	Emet la donnée en hexadécimal signé
SHEX1-SHEX4 var	Emet la donnée en hexadécimal signé de 1 à 4 chiffres
IHEX variable	Emet la donnée en hexadécimal précédée de S
IHEX1-IHEX4 var	Emet la donnée en hexadécimal de 1 à 4 chiffres, précédée de S
ISHEX variable	Emet Io donnée en hexadécimal signé précédée de S
ISHEX1-ISHEX4 var	Emet la donnée en hexadécimal signé de 1 à 4 chiffres, précédée de S
BIN variable	Emet la donnée en binaire
BIN1-BIN16 variable	Emet la donnée en binaire de 1 à 16 chiffres
SBIN variable	Emet la donnée en binaire signé
SBIN1-SBIN16 var	Emet la donnée en binaire signé de 1 à 16 chiffres
IBIN variable	Emet la donnée en binaire précédée de %
IBIN1-IBIN16 var	Emet la donnée en binaire de 1 à 16 chiffres, précédée de %
ISBIN variable	Emet la donnée en binaire signé précédée de %
ISBIN1-ISBIN16 var	Emet la donnée en binaire signé de 1 à 16 chiffres, précédée de %

SHIFTIN
SHIFTIN patte d, patte c, mode, [variable {\bits}...]
Reçoit des données sous forme série au rythme d'une horloge de transmission fournie par le Stamp.

- Patte d est une constante, une variable ou une expression, de type bit, nibble octet ou mot, comprise entre 0 et 15, spécifiant le numéro de port d'entrée/ sortie à utiliser pour la réception des données série. Cette patte est placée en entrée dès le début de l'instruction et elle y reste ensuite quel qu'ait pu être son état antérieur.

- Patte c est une constante, une variable ou une expression, de type bit, nibble octet ou mot, comprise entre 0 et 15, spécifiant le numéro de port d'entrée/ sortie à utiliser pour l'émission de l'horloge. Cette patte est placée en sortie dès le début de l'instruction et elle y reste ensuite quel qu'ait pu être son état antérieur. L'horloge générée par le Stamp est un signal rectangulaire dont l'état haut dure environ 14 µs (Stamp2) et 12 µs (Stamp2SX) et l'état bas 46 µs (Stamp2) et 12 µs (Stamp2).

- Mode est une constante, une variable ou une expression, de type bit, nibble, octet ou mot, comprise entre 0 et 3, spécifiant le mode de fonctionnement de la réception selon le format indiqué dans le tableau suivant. Mode peut aussi être défini au moyen des symboles indiqués dans ce même tableau, qui sont connus de l'interpréteur Basic du Stamp.

- Variable est une variable de type bit, nibble, octet ou mot, qui contient la donnée reçue.

- Bits est une constante, une variable ou une expression optionnelle, de type bit, nibble, octet ou mot, comprise entre 1 et 16, qui spécifie le nombre de bits à recevoir. Si bits n'est pas spécifié, le nombre 8 est pris par défaut.

N°	Sens du transfert	Validité des données	Symbole
0	MSB en premier	Données valides sur front montant	MSSPRE
1	LSB en premier	Données valides sur front montant	LSBPRE
2	MSB en premier	Données valides sur front descendant	MSBPOST
3	LSB en premier	Données valides sur front descendant	LSBPOST

SHIFTOUT
SHIFTOUT patte d, patte c, mode, [variable {\bits}, ...]
Émet des données sous forme série au rythme d'une horloge de transmission fournie par le Stamp.

- Patte d est une constante, une variable ou une expression, de type bit, nibble octet ou mot, comprise entre 0 et 15, spécifiant le numéro de port d'entrée/ sortie à utiliser pour l'émission des données série. Cette patte est placée en sortie dès le début de l'instruction et elle y reste ensuite quel qdait pu être son état antérieur.

- Patte c est une constante, une variable ou une expression, de type bit, nibble octet ou mot, comprise entre 0 et 15, spécifiant le numéro de port d'entrée/ sortie à utiliser pour l'émission de l'horloge. Cette patte est placée en sortie dès le début de l'instruction et elle y reste ensuite quel qu'ait pu être son état antérieur. L'horloge générée par le Stamp est un signal rectangulaire dont l'état haut dure environ 14 µs (Stamp2) et 12 µs (Stamp2SX) et l'état bas 46 µs (Stamp2) et 12 µs (Stamp2SX).

- Mode est une constante, une variable ou une expression, de type bit, nibble, octet ou mot, comprise entre 0 et 1, spécifiant le mode de fonctionnement de l'émission selon le format indiqué dans le tableau suivant. Mode peut aussi être défini au moyen de l'un des deux symboles visibles dans ce même tableau.

N°	Sens du transfert	Symbole
0	LSB en premier	LSBFIRST
1	MSB en pfemier	MSBFIRST

- Variable est une variable de type bit, nibble, octet ou mot, qui contient la donnée à émettre.

- Bits est une constante, une variable ou une expression optionnelle, de type bit, nibble, octet ou mot, comprise entre 1 et 16, qui spécifie le nombre de bits à émettre. Si bits n'est pas spécifié, le nombre 8 est pris par défaut.

SLEEP
SLEEP secondes
Met le Stamp en mode sommeil pendant un certain nombre de secondes.

La résolution de cette instruction est de 2,304 s et sa précision de l'ordre de 99,9 %. Dans ce mode, la consommation est de 100 µA(Stamp2) ou 200 µA(Stamp2SX), hors consommation de la circuiterie d'entrées/sorties.

- Secondes est une constante, une variable ou une expression, de type bit, nibble, octet ou mot, comprise entre 1 et 65535. Elle indique la durée du mode sommeil en multiples de 2,304 s.

Pendant ce mode, les entrées et sorties du Stamp conservent leur état. De ce fait, si une sortie commandait une charge avant de rencontrer l'instruction SLEEP, elle continuera à le faire même pendant la durée du mode sommeil provoqué par cette instruction. Par contre, en raison du mode de fonctionnement interne du Stamp, les sorties concernées passent en état de haute impédance pendant 18 ms toutes les 2,3 s environ. Il importe donc d'en tenir compte si cela peut avoir une influence sur la ou les charges qui y sont connectées.

STOP
STOP
Arrête l'exécution du programme en cours.

L'exécution du programme est arrêtée mais le mode basse consommation n'est pas activé. Cette instruction est donc identique à END mais maintient les entrées/sorties à leurs niveaux sans jamais les faire passer en haute impédance. Seul un reset matériel permet de quitter le mode STOP.

TOGGLE
TOGGLE patte
Met la patte spécifiée en sortie si elle ne l'était pas déjà et change son état (haut vers bas ou vice versa).

- Patte est une constante, une variable ou une expression, de type bit, nibble, octet ou mot, comprise entre 0 et 15, qui indique le numéro de port d'entrée/sortie à utiliser.

WRITE
WRITE adresse, variable
Place la donnée contenue dans la variable dans la mémoire EEPROM à l'adresse spécifiée.

- Adresse est une constante, une variable ou une expression, de type bit, nibble octet ou mot, comprise entre 0 et 2047. Elle spécifie l'adresse.

- Variable est une variable de type bit, nibble, octet ou mot, qui contient la donnée à écrire dans la mémoire EEPROM.

Notez que l'EEPROM est utilisée à la fois pour les données et pour le programme. Le programme est toujours placé en descendant à partir de l'adresse la plus haute (2047). Il est donc conseillé de placer les données en montant à partir de l'adresse la plus basse (0) pour éviter tout écrasement du programme par cette instruction. Une bonne pratique consiste à réserver de la place en mémoire EEPROM en utilisant l'instruction DATA et à se tenir ensuite à cette zone lors de l'utilisation de l'instruction WRITE. En effet, lors du téléchargement du programme dans la mémoire du Stamp, l'interpréteur vérifie que les zones de données réservées par DATA n'entrent pas en conflit avec la mémoire utilisée par le programme. Tout risque d'écrasement du programme par les données est ainsi écarté.

XOUT
XOUT patte m, patte z, [maison\touche {\cycles, ...}]
Génère des codes compatibles du système de télécommande par courant porteur X10.

- Patte m est une constante, une variable ou une expression, de type bit, nibble octet ou mot, comprise entre 0 et 15, spécifiant le numéro de port d'entrée/ sortie à utiliser pour la commande de modulation. Cette patte est placée en sortie dès le début de l'instruction et elle y reste ensuite quel qu'ait pu être son état antérieur.

- Patte z est une constante, une variable ou une expression, de type bit, nibble octet ou mot, comprise entre 0 et 15, spécifiant le numéro de port d'entrée/ sortie à utiliser pour l'entrée de l'information de détection de passage par zéro. Cette patte est placée en entrée dès le début de l'instruction et elle y reste ensuite quel qu'ait pu être son état antérieur.

- Maison est une constante, une variable ou une expression, de type bit, nibble, octet ou mot, spécifiant le code « maison » du système X10 où les lettres A à P sont remplacées respectivement par les chiffres 0 à 15.

- Touche est une constante, une variable ou une expression, de type bit, nibble, octet ou mot, représentant le numéro des touches des claviers X10 avec la correspondance entre les touches 1 à 16 et les nombres 0 à 15 (la touche 1 correspond à la valeur 0 et ainsi de suite). Un des codes de commande indiqués dans le tableau suivant peut également être utilisé.

Symbole	Valeur numérique
UNITON	%10010
UNITOFF	%11010
UNITSOFF	%11100
LIGHTSON	%10100
DIM	%11110
BRIGHT	%10110

- Cycles est une constante, une variable ou une expression optionnelle, de type bit, nibble, octet ou mot, qui remplace la valeur 2 prise par défaut. Elle ne doit être utilisée que pour les commandes « dim » et « bright ».