

Liste complète des commandes possibles.

Toutes les commandes commencent par '\$' peuvent être saisies librement en MAJuscules ou en Minuscules. Par exemple pour obtenir l'aide à la détermination des origines on peut indiquer "\$?" ou son équivalent en caractère minuscule soit "\$,".

Commande Effet de la commande

- \$, ou \$? Affiche la liste des commandes possibles.
\$a ou \$A Demande confirmation puis passe en mode "Analyse". Commence par imposer les paramètres optimisés.
- \$b ou \$B Affiche la taille logicielle des deux Buffers TX et TY.
\$c ou \$C Affiche les Codes ASCII des caractères saisis.
\$d ou \$D Option "Encadrement Double" OUI / NON. Si l'option "Encadrer" n'est pas active, valide cette dernière.
\$e ou \$E Option "Encadrer" OUI / NON.
\$gn ou \$G Impose la Grandeur de la police. *n* respectera [1 à 9].
\$in ou \$I Choix du "nom" du fichier Image. *n* sera entre [0 à 9].
\$h ou \$H (HELP) Aide à la rédaction du préambule d'un texte.
- \$ln ou \$L Listage des 255 codes ASCII affichables.
- \$mn ou \$M Comme pour \$s : Mémorise les paramètres actuels.
\$pn ou \$P Purge la fenêtre d'écran du Moniteur de l'IDE.
\$rn ou \$R Restitue les paramètres Mémorisés. (Sauvegardés.)
\$sn ou \$S Comme pour \$m : Sauvegarde les paramètres actuels.
\$t... ou \$T "Titre" du fichier.gco avec 16 caractères maximum.
- \$u... ou \$U Bascule d'affichage des limites Utilisées actuellement.
\$v... ou \$V Valider des valeurs optimisées pour pouvoir graver un maximum de texte en fonction des paramètres actuels.
\$w ou \$W (Word) Affiche la liste les caractères de substitution.
\$xnn ou \$X Impose sur X'X la position de l'origine du texte. @
\$yyn ou \$y Définit sur Y'Y la position de l'origine du texte. @
\$z ou \$Z Remise à Zéro des paramètres. (RESET des options.)

Repérées en jaune, quatre commandes dédiées aux développeurs logiciels. Elles ne sont pas pertinentes en utilisation courante du compilateur, ce ne sont que des aides pour modifier le programme.

@ : Cette origine correspond à l'angle inférieur gauche du premier caractère supposé être une lettre Majuscule. Sera utile pour décentrer notamment la position de gravure sur le plateau de la machine.

Protocole pour rédiger un texte.

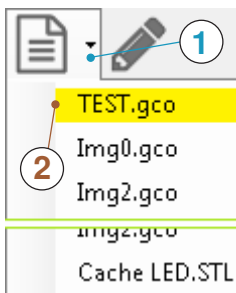
- 1) Dans Bloc-notes par exemple rédiger le texte tel qu'il est désiré avec les interlignages et les centrages éventuels des lignes avec des espaces. Les passages à la ligne seront remplacés en ② par le caractère '\$'. Le caractère 'ç' ne générera une cédille *que si celui qui le précède est un 'c'* majuscule ou minuscule.
- 2) Éliminer les interlignages et les remplacer par des '\$'.
- 3) Changer les caractères spécifiques par ceux de substitution :
 - Pour 'œ' prendre '2' et pour 'Œ' le caractère '2' suivi de 'o'.
 - Pour 'π' utiliser "'" et pour '±' remplacer par "'."
 - Pour obtenir '≠' on substituera '_'.
 - Pour avoir 'α' on frappera '{' et pour 'β' on prendra '}'.
 - Le symbole 'Ø' est obtenu avec 'α'.
 - La résistance ohmique 'Ω' s'obtient avec '\$'.
 - Enfin '~' s'obtient directement avec la touche [Alt Gr ~] mais uniquement s'il est suivi d'un autre caractère.
 - '←' avec '\ ' de [ALT 7], '↑' avec '^' de [ALT 9] et '→' avec '\$'.

Dans le compilateur :

- 4) Définir les options :
 - '\$iN' : Numéro de ImgN. (N de 0 à 9.)
 - '\$tTitre' : Titre de l'image. (Maximum 16 caractères.)
 - '\$gN' : Grandeur des caractères. (N de 1 à 9.)
 - '\$xN' : Origine gauche du premier caractères.
 - '\$yN' : Origine haute du premier caractères.
 - '\$e' : Encadrement OUI/NON. • '\$d' : Double OUI/NON.
 - '\$s' : Sauvegarder les paramètres est très conseillé.
- 5) Sur le moniteur de l'IDE imposer une vitesse de 115200 baud.
(Valeur maximale actuellement imposée dans void setup().)
- 6) Dans Bloc-notes copier le texte formaté et le coller dans la fenêtre de saisie du Moniteur puis valider.
- 7) Le résultat attendu est affiché suivi de la demande confirmation. Répondre par 'n' fait reprendre la saisie sans modifier les options. Répondre par 'o' déclenche la compilation. Si une erreur est détectée, un avertissement est généré et le Moniteur repasse en attente d'une saisie : Reprendre en ⑤. Si le texte est correct, il est compilé et les options de base sont réinitialisées sauf pour le Titre du fichier.

Méthode pour tester les "images texte".

- 1) Ouvrir Bloc-notes et sauvegarder une première fois avec un nom de fichier comme **TEST.gco** par exemple.
- 2) Ouvrir Repetier-Host V1.6.2 et charger une première fois **TEST.gco** puis valider les options et .
- 3) Copier dans la fenêtre du Moniteur l'intégralité du code **compilé** et délimité par les deux lignes
=====.
- 4) Dans Bloc-notes avec TEST.gco frapper : (1) **[CTRL A]** puis **[CTRL V]** suivi de **[CTRL S]**.
- 5) Valider la fenêtre de Repetier-Host V1.6.2 puis cliquer sur en 1, et un peu à gauche sur le nom du fichier en 2 pour voir le résultat.



Chaque fois que le traitement avec le compilateur d'un nouveau texte aura été effectué, reprendre la procédure en ③.

(1) : Vérifier en bas à droite du Bloc-notes le nombre de lignes.

Grandeur police	Nb CAR par ligne	Nb MAX de lignes	Origines		
			X MIN	Y MAX	
1	45 45	25 24 24	2 4	197	195 193
2	22 22	12 12 12	4 8	192	188 184
3	14 14	8 8 7	6 12	187	181 175
4	10 10	6 6 5	8 16	182	174 166
5	8 7	5 4 4	10 20	177	167 157
6	7 6	4 4 3	12 24	172	160 148
7	6 5	3 3 2	14 28	167	153 139
8	5 4	3 3 2	16 32	162	146 130
9	4 3	3 2 2	18 36	157	139 121

Ordre de grandeur des valeurs maximales, car dépend de la largeur des caractères *et si la dernière ligne comporte des lettres minuscule avec queue.* (Enlever une ligne si la dernière est avec minuscule à queue.) Présenté en noir les estimations indiquées sans demande d'encadrement, et **en rouge les limites si encadrement** désiré. En marron l'**option Cadre DOUBLE** est prise en compte par "\$?" et par "\$V". **En jaune** les valeurs conseillées avec "\$?" ou validées avec "\$V". **En violet** les limites à respecter pour le texte.

Choix des caractères pour mémoriser aisément.

Toutes les commandes peuvent être saisies librement en MAJuscules ou en Minuscules. Par exemple pour obtenir la liste des commandes possibles, on peut indiquer "\$?" ou son équivalent en caractère minuscule soit "\$,".

- '\$' pour préciser une commande ou imposer un passage à la ligne dans les textes est choisi car (*) et voisin de "Valider".
- 'g' est choisi pour imposer la **G**randeur des caractères. Mais on peut aussi utiliser 'd' si on pense à **D**imensions.
- 's' est possible pour imposer **S**auvegarder les paramètres actuels. Mais on peut aussi le remplacer par 'm' pour **M**émoriser.

Dans les textes :

- Pour le caractère 'Ç' ou 'ç' **on doit impérativement saisir un premier caractère** 'C' ou 'c', puis le faire suivre par la directive 'ç'.
- Pour obtenir le symbole 'œ' on frappera '2' qui suggère la fusion de deux entités. Pour 'Œ' le caractère '2' sera suivi de 'o'. (*)
- Si le symbole 'π' est désiré, utiliser '"' qui sous le 3 rappelle PI. (*)
- Si l'on désire '±', il sera obtenu en saisissant ';' qui suggère deux entités situées l'une au dessus de l'autre. (**)
- Pour obtenir '≠' on substituera '_'. (*)
- Pour avoir 'α' on frappera '{'. (**)
- Pour le caractère 'β' on prendra '}'. (**)
- Le symbole 'Ø' est souvent employé pour désigner "diamètre". Aussi, il sera obtenu avec '⌘' qui suggère un trou percé vu de dessus.
- Pour la résistance ohmique le 'Ω' s'obtient avec '\$'. (**)
- Pour le symbole "alternatif" noté '~' il s'obtient directement avec la touche [Alt Gr~] mais uniquement s'il est suivi d'un autre caractère. Si on le désire "seul", **le faire suivre d'un caractère quelconque pour qu'il soit visualisé**, effacer ce caractère "parasite".

Enfin les derniers caractères disponibles permettent d'obtenir :

- La flèche gauche '←' avec '^' de [ALT 7]. (**)
- La flèche vers le haut '↑' avec '^' de [ALT 9].
- La flèche droite '→' avec '£'. (**)
- Noter que 'l' deviendra 'I'.

(*) : Caractère choisi car n'impose pas [MAJ] simplifiant la frappe.

(**) : Caractère généralement peu utilisé en phrases banales.

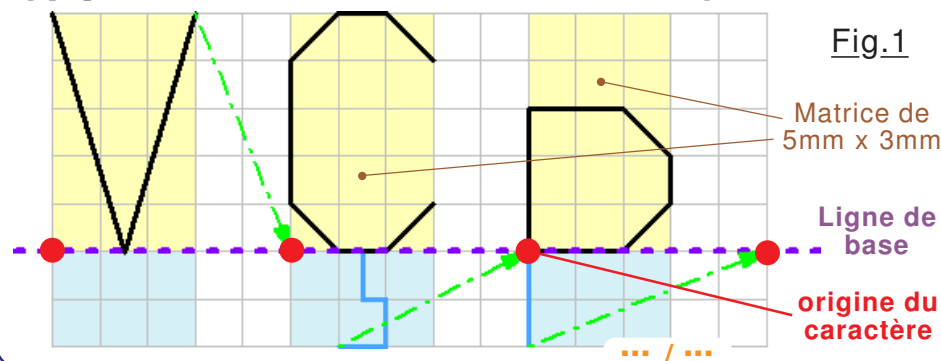
Modifier un caractère dans la table. 1/3

L'intégralité des caractères qui donnent une valeur différente de zéro est actuellement utilisée. On peut toutefois désirer changer la forme d'une matrice que l'on utilise peu par un autre personnalisé. Les caractères logés en EEPROM constituent l'alphabet de base, c'est donc probablement dans la liste des caractères "spéciaux" que l'on cherchera à modifier la police. Obligatoirement logés dans **Complement_de_police[NNN]** il faudra terminer en optimisant la taille **NNN** du tableau. Dans ce but prendre une valeur volontairement trop faible et l'augmenter par dichotomie jusqu'à ne plus avoir le message d'erreur **Compil ... error: too many initializers for ... Erreur lors de la compilation.** (Prendre une bonne marge par sécurité vu que la zone mémoire réservée au programme n'est utilisée actuellement qu'à environ 77%)

NOTE : Pour faciliter le travail, la fonction "\$c" permet au programmeur d'obtenir en décimal les valeurs ASCII des Codes pour les caractères frappés au clavier, et la commande "\$I" Liste des 255 caractères affichables sur le moniteur de l'IDE.

► Géométrie des caractères et du texte.

Systématiquement le tracé d'un caractère se fait à partir d'une **origine relative**, le LASER étant **initialement positionné** sur cette dernière sur la **ligne de base**, c'est à dire en bas à gauche ; la **Matrice** étant celle d'une lettre majuscule. Pour les caractères de **Grandeur** unitaire la taille de la matrice fait **5mm** de haut, de **3mm** de large avec un inter-caractère de **2mm**. La définition est de 0,5mm pour les coordonnées du "matriçage". **Cédille ou queue de minuscules** (*g,p,q etc*) sont dans une zone de **2mm** sous la **ligne de base**.



```
void Traiter_le_Caractere() {
```

```
.....
```

```
//----- Si minuscule avec queue décaler vers le bas ----
```

```
if ((Clone == 'g') || (Clone == 'j') || (Clone == 'p')  
    || (Clone == 'q') || (Clone == 'y') || (Clone == ','))  
    Decale_vers_le_bas();
```

```
.....
```

```
//----- Pointer la position suivante -----
```

```
if ((Clone == 'g') || (Clone == 'j') || (Clone == 'p')  
    || (Clone == 'q') || (Clone == 'y') || (Clone == ','))  
    Origine_Caractere_Y = SAV_Origine_Y; // Recale Y ...
```

Recaler l'origine du caractère sur la ligne de base pour passer correctement à l'origine du suivant impose un déplacement sans pyrograver sur la machine, donc une ligne de code en plus. Il faut de ce fait corriger le calcul de la taille du fichier :

```
void Tester_en_memoire_programme() {
```

```
.....
```

```
if ((Clone == 'g') || (Clone == 'j') || (Clone == 'p')  
    || (Clone == 'q') || (Clone == 'y') || (Clone == ','))  
    Taille_du_fichier_produit++;}
```

Et surtout pour un nouveau caractère qui logiquement remplacera un individu dans la table des codes complémentaires. C'est par exemple le cas de la virgule dont une partie est sous la ligne de base :

```
void Tester_en_memoire_programme() {
```

```
.....
```

```
if (char(Clone) == Caractere) {  
    if (Clone == ',') Taille_du_fichier_produit++;
```

```
.....
```

Ce type de caractères occupent plus de place verticalement sur la dernière ligne du texte. Donc pour encadrer correctement et positionner le LASER avant gravure il importe de recalculer la limite verticale :

```
void Tester_derniere_ligne() {  
    boolean Queue_presente = false;
```

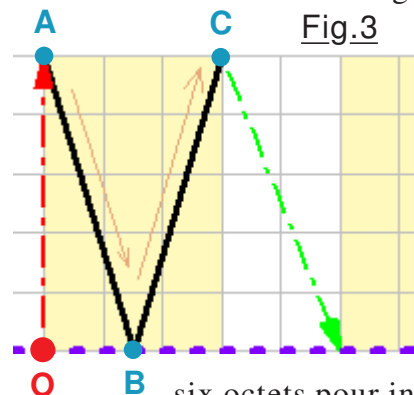
```
.....
```

```
if ((Clone == 'g') || (Clone == 'j') || (Clone == 'p') || (Clone ==  
    'q') || (Clone == 'y') || (Clone == ',') || (Clone == char(231)))  
    Queue_presente = true;}
```

```
.....
```

Modifier un caractère dans la table. 2/3

Pour illustrer le principe de codage des caractères, nous allons prendre en exemple le cas de la lettre 'V' majuscule, les valeurs étudiées se retrouvant dans le programme d'initialisation de l'EEPROM. Afin de minimiser les mouvements sur la machine, on va comme montré sur la Fig.3 effectuer trois déplacements.



Déplcmnt	sur X	sur Y	code X	code Y
O > A	0	5	100	110
A > B	1,5	0	3	0
B > C	3	5	6	10

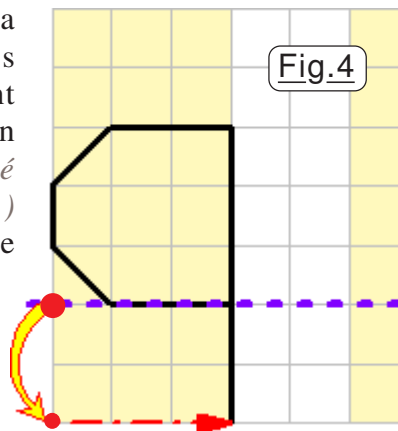
Le premier octet du groupe sera le code du caractère 'V'. Comme on le voit sur la Fig.3 **le nombre de déplacements est de 3**, ce qui exigera

six octets pour indiquer les valeurs de **X** et de **Y** dont les valeurs sont résumées dans la zone violette du tableau ci-dessus.

```
Ecrire_un_OCTET('V'); Ecrire_un_OCTET(3);
Ecrire_un_OCTET(100); Ecrire_un_OCTET(110);
Ecrire_un_OCTET(3); Ecrire_un_OCTET(0);
Ecrire_un_OCTET(6); Ecrire_un_OCTET(10);
```

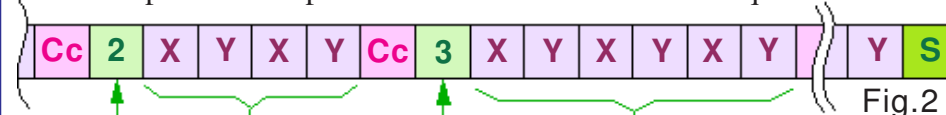
Particularité des caractères avec queue.

Générés de façon particulière ces caractères s'inscrivent dans une matrice qui fait 7mm de haut au lieu de cinq. L'origine du caractère reste en bas à gauche de la matrice de description. Aussi, les coordonnées des points sur X'X restent à 3mm de large, mais en hauteur on passe à 7mm au lieu de cinq. (*Multiplié ensuite par deux pour avoir les 1/2mm.*) Au moment de pyrograver le caractère il faut effectuer un changement de la position de l'Origine comme montré sur la Fig.4, puis, en fin de traçage replacer cette dernière en ligne de base.



Coder un caractère dans la table.

Optimiser l'utilisation de la machine revient à minimiser les déplacements, et de façon générale à terminer le plus proche possible de la prochaine origine. La police de caractère sera définie dans un tableau de **bytes**, et constituée d'une série d'individus, chacun étant codé, comme précisé sur la Fig.2 par une suite ordonnée d'octets représentatifs. La fin de la table est indiquée par une "Sentinelle" constituée de l'octet de valeur zéro. Les deux tableaux de description de la police sont de structures identiques.



Chaque descripteur commence par un octet de type **char** contenant le **Code ASCII** du caractère défini. L'octet qui suit précise le **nombre des déplacements** à effectuer pour tracer l'individu. **Chaque déplacement** sera effectué **depuis la position actuelle** jusqu'à la position indiquée par les **coordonnées relatives X et Y** du point à atteindre exprimées dans le repère XOY du plateau de la machine. (*Coordonnées relatives / à l'origine du caractère.*)

Coder les coordonnées relatives X,Y.

Dans le but de réduire à outrance la taille de la table de description de la police de caractères, les coordonnées relatives **X** et **Y** du prochain point à atteindre intègrent dans l'octet la nature du déplacement, c'est à dire si le LASER doit graver ou non. **Si le déplacement doit se faire sans pyrograver, on ajoute 100** à la valeur de **X et de Y**, cet artifice étant traité par logiciel au décodage.

ATTENTION : Pour pouvoir préciser les coordonnées avec la précision de la machine, c'est à dire le demi millimètre, sans pour autant exiger l'utilisation d'un **float**, on multiplie par deux les valeurs des déplacements relatifs.

Lorsque le tracé du caractère est terminé, le déplacement **représenté en vert** sur la Fig.1 **pour atteindre la prochaine origine** est généré par le programme. Ce déplacement n'est donc pas codé dans les tables descriptives de la police. Si la largeur de l'élément est différente de 3mm, le cas est traité par logiciel.

Modifier un caractère dans la table. 3/3

Autre spécificité : La largeur d'un caractère non standard engendre une ligne dont l'étendue ne sera pas conforme au nombre d'individus qui la composent. En particulier les éléments plus larges que 5mm vont engendrer des débordements par la droite s'ils ne sont pas pris en compte. La largeur du texte doit s'adapter.

► Particularité des caractères de largeur non standard.

Que ce soit parce qu'ils sont moins larges comme le 'I', le 'l' etc, ou plus larges que le standard de 3mm comme pour le 'W' par exemple, le logiciel doit impérativement en tenir compte :

```
void Calcule_les_limites() {  
    switch (Caractere) {  
        .....  
        case 'n' : {MAX_X = MAX_X + 4; break;}  
        case 'l' : {MAX_X = MAX_X + 2; break;}  
        case 'w' : {MAX_X = MAX_X + 6; break;}  
        case 'W' : {MAX_X = MAX_X + 6; break;}  
        case '' : {MAX_X = MAX_X + 6; break;}  
        .....  
    }
```

Noter également que les caractères dont la largeur est supérieure au standard de 3mm influencent le travail effectué par la procédure qui traite la commande "\$A". Ces éléments doivent se voir intégrés dans le test de la procédure dédiée:

```
void Transcode_le_texte() {  
    .....  
    if ((Caractere == 'm') || (Caractere == 'w') ||  
        (Caractere == 'W') || (Caractere == '')) ||  
        (Caractere == '$')) {  
        .....  
    }
```

► Particularité des caractères de substitution.

Les afficher correctement à l'écran et dans le programme.gco facilitera considérablement l'utilisation du compilateur. Deux procédures sont concernées par de la substitution. En particulier l'affichage du texte avant compilation pour attente de confirmation sera effectué de façon à présenter le caractère qui sera gravé. Par exemple Ω sera affiché par 'R' car l'oméga n'est pas disponible dans

... / ...

Problème de collision de PILE.

Survenu en cours de développement du programme, le manque de place en mémoire dynamique a obligé de changer de stratégie. Du reste, lors de la compilation l'IDE s'est rendu compte de ce problème et a affiché le texte d'alerte :

La mémoire disponible faible, des problèmes
de stabilité pourraient survenir.

Fig.1

Initialement, la table de la POLICE de caractères était en mémoire EEPROM. Puis, pour tenter d'obtenir un programme "autonome" elle a été logée directement dans le croquis, provoquant la collision de PILE. Pour dégager de l'espace en mémoire dynamique, il a été envisagé de placer les textes en EEPROM, mais ils ne totalisaient que 368 octets ce qui n'est pas "rentable". Aussi, pour optimiser l'usage de l'EEPROM la table des caractères est repassée en EEPROM, quitte à loger dans le corps du programme une table complémentaire car la place n'y est pas suffisante. Actuellement l'EEPROM est saturée. Il ne reste qu'un seul octet non utilisé. La table de génération des caractères "Complement_de_police[815]" est placée en mémoire de programme pour ne pas empiéter sur l'espace de mémoire dynamique. Bien qu'actuellement le code n'avoisine que 77% on doit par principe minimiser la *réserve de place*.

► Optimiser l'utilisation de la mémoire dynamique.

Actuellement, après une optimisation poussée du code, l'alerte de la Fig.1 ne se joue qu'à quelques octets près. Deux causes principales engendrent de la consommation de mémoire dynamique :

- La taille réservée aux deux zones de saisies des informations par la ligne série du moniteur. Mais LGR_chaine 185 ne doit pas dépasser la taille maximale du "buteur" du Moniteur série. (Cette valeur est affectée à Chaine_Memorisee et à Tampon[LGR_chaine+1].)
- La taille consommée par les chaînes de caractères à afficher. Chaque texte augmente de sa "longueur" à la fois la taille du programme et la zone consommée en mémoire dynamique.

C'est la raison pour laquelle les textes affichés sont optimisés. Si deux ou plusieurs textes affichés sont strictement identiques, le compilateur optimise en créant intrinsèquement une seule procédure. Par contre, si un seul caractère est différent, il y aura plusieurs séquences différentes. Pour optimiser les textes, les séquences communes sont extraites et placées en procédures séparées.

Informations diverses.

Quand on valide un texte, après compilation on considère que celui qui suivra n'a rien à voir. Donc, les paramètres initiaux sont réinitialisés mis à part le TITRE de l'image qui est conservé. Si on désire plusieurs textes avec les mêmes paramètres, il suffit de les sauvegarder avant la première saisie avec les commandes '\$s' ou '\$m', puis de les restituer avec '\$r' à chaque nouvelle image.

- **Le LASER sera toujours positionné à l'angle inférieur gauche du texte ou de l'encadrement lors de la phase initiale de pyrogravure.**
- "BUG" connu : Si le texte contient un 'Œ' majuscule, sous certaines conditions l'encadrement ne sera pas correct en largeur.
- **Note** : Le texte le plus grand que l'on peut saisir dans la fenêtre dédiée du Moniteur est de 185 caractères, les '\$' pour passer à la ligne étant compris. Cette restriction conduit déjà à un fichier image d'environ 1800 lignes. Si l'on désire un texte de plus grande ampleur, il suffit de modifier l'origine pour décaler la zone de travail vers le bas de la hauteur du nouveau texte, et si l'encadrement est désiré, de le tracer à l'aide d'un programme rédigé à la main.

➤ Changement des origines.

- Si on donne une valeur négative ou différente d'un nombre, l'origine sera forcée à zéro.
- Si on donne une valeur d'origine trop importante le paramètre sera réajusté aux limites matérielles de la machine et un message d'erreur préviendra l'opérateur.

➤ Cas particulier du symbole '~'.

Il n'apparaît dans la fenêtre de saisie que s'il est suivi d'un caractère quelconque. Donc, **si ce caractère '~' doit terminer une ligne** :

- Le frapper deux fois suivi de B.S. pour enlever le doublon.

➤ La commande analyse 'A'.

\$a ou \$A demande confirmation puis passe en mode "Analyse". L'option commence par imposer **les paramètres optimisés** qui **positionnent le texte au maximum en haut et à gauche** de la table de la pyrograveuse. La saisie sera alors formatée pour respecter l'optimisation, c'est à dire imposer des passages à la ligne.

la police d'écran du Moniteur, alors que β pour '}' et \pm pour ';' peuvent être présentés directement sans substitution intermédiaire.

```
void Affiche_le_texte_qui_sera_traduit_en_gco() {  
    .....  
    switch (Caractere) {  
        case '{' : {Serial.print('A'); LGR++; break;}  
        case '}' : {Serial.print(char(223)); LGR++; break;}  
        case '"' : {Serial.print('P'); LGR++; break;}  
        case ';' : {Serial.print(char(177)); LGR++; break;}  
        .....  
    }
```

Dans le programme ImgN.gco chaque caractère généré sera précisé par la remarque qui précède les lignes de code. Quand c'est possible comme pour ';' c'est \pm qui est indiqué, et si le symbole à visualiser n'est pas dans la police "affichable", un texte comme "PI" remplacera la forme de π qui sera gravé sur la machine :

```
void Generer_le_code_gco() {  
    .....  
    switch (Caractere) {  
        case ' ' : {Serial.print(" ESPACE"); break;}  
        case '"' : {.....; Serial.print("PI"); break;}  
        case ';' : {.....; Serial.print(char(177)); break;}  
        case '{' : {.....; Serial.print("Alpha"); break;}  
        case '}' : {.....; Serial.print(char(223)); break;}  
        case char(95) : {.....; Serial.print(char(208)); break;}  
        .....  
        case char(167) : {.....; Serial.print("Ohm"); break;}  
    }
```

Influence des arrondis en déplacements vectoriels.

Compte tenu des "résidus de calculs" lors des déplacements vectoriels, certains caractères comme "Différent de" ou -> sont codés légèrement dissymétriques pour optimiser le tracé de la petite police de 5mm de haut. Une fois agrandis on observe ces dissymétries, car c'est la plus petite police qui a été privilégiée. Par ailleurs les caractères en police unitaire sont parfois déformés bien que codés symétriques. Cette déformation disparaît à la gravure dès la valeur 2 pour **Grandeur**.

