STM8eForth V3.0 référence du vocabulaire.

Conventions typographique

a Adressse 16 bits

ad Adresse 32 bits

b Adresse 16 bits d'une chaîne de caractère

c Caractère ASCII ou octet.

f Indicateur booléen 0 indique faux tout autre valeur est considérée comme vrai.

i Entier 16 bits signé {-32767...32767}, -32768 utilisé comme indicateur de débordement.

id Entier double (32 bits) signé

n Valeur 16 bits sans type défini.

n+ Entier positif.

u Entier 16 bits non signé.

ud Entier 32 bits non signé.

nx Représente un nombre quelconque d'élément sur la pile.

R: Représente la pile des retours.

(nx1 -- nx2) Commentaire indiquant la liste des paramètres à gauche et des résultats à droite.

T Indique la valeur booléenne **vrai**.

F Indique la valeur booléenne **faux**.

Index

Chaque module a une section séparé pour son vocubulaire. Cet index conduit à la section concernée.

Vocabulaire principal du système

- **!** (n a --) Dépose la valeur **n** à l'adresse **a**
- **#** (u1 -- u2) Utilisé dans la conversion d'entier en chaîne. Extrait un digit de l'entier **u1** et et ajout l'ajoute à la chaîne de sortie. u2=u1/BASE.
- **#>** (w -- b u) Complète la conversion d'un entier en chaîne. **b** est l'adresse de la chaîne résultante et **u** sa longueur. **w** est le quotient de la dernière division et donc égal à zéro.
- **#S** (u -- 0) Converti en chaîne tous les digits de u. Lorsque cette opération est complétée la valeur **0** demeure sur la pile.
- **#TIB** (-- a) Empile l'adresse de la variable système **#TIB**. Cette variable contient le nombre de caractères qu'il y a dans le TIB *(Terminal Input Buffer)*.
- **$"** (-- ; <string>) Compile la chaîne litérale **<string>**.
- **$COMPILE** (b --) Routine invoquée par **EVAL** lorsqu'en mode compilation. **b** est l'adresse de la chaîne du nom qui est recherché dans le dictionnaire. Si ce nom n'est pas dans le dictionnaire le compilateur vérifie s'il s'agit d'un entier et dans ce cas compile un entier litéral. Toute autre situation conduit à un abandon de compilation avec message d'erreur.
- **$INTERPRET** (b --) Routine invoquée par **EVAL** lorsqu'en mode interprétation. **b** représente le nom qui est recherché dans le dictionnaire. Ce nom s'il est trouvé sera interprété. S'il n'existe par une tentative de converion en entier est réalisée. Cet entier sera empilé. Toute autre situation conduit à un abandon avec message d'erreur.
- **'** (-- a ; <string>) Ce mot est suivit d'une chaîne de caractère qui doit représenter un mot du dictionnaire. Si ce mot est trouvé **a** est l'adresse d'éxécution de ce mot. Un échec résulte en un abandon avec message d'erreur.
- **'BOOT** (-- a) Retourne l'adresse du programme qui doit-être exécuté au démarrage du système. Par défaut il s'agit de **hi**.
- **'EVAL** (-- a) Retourne l'adresse de la variable système **INTER** qui contient l'adresse du code qui doit-être exécuté par **EVAL**.
- **(** (--) Ce mot introduit un commentaire délimité par **)**
- ***** (i1 i2 -- i3) Multiplication signée. i3=i1*i2.
- ***/** (i1 i2 i3 -- i4) Multiplication de i1 par i2 avec résultat conservé en entier double suivit d'une division par i3.
- ***/MOD** (i1 i2 i3 -- u i4) Multiplication de i1 par i2. Le résultat est conservé en entier double pour être divisé par i3. Le quotient i4 et le reste u sont retourné.
- **+** (i1 i2 -- i3) i3 est la somme de i1 et i2.
- **+!** (i a --) Ajoute la valeur i à l'entier situé à l'adresse a.

- **,** (n --) Compile la valeur n
- **-** (i1 i2 -- i3) i3 est le résultat de la soustraction i1-i2.
- **-1** (-- -1) Constante numérique **-1**. Utilisée pour indiquée une valeur booléenne vrai.
- **.** (i --) Imprime l'entier au sommet de la pile.
- **."** (--) Compile une chaîne litérale pour impression. Cette chaîne est terminée par le caractère **"**.
- **.(** (--) Mot immédiat qui imprime le commentaire délimité par **)**.
- **.ID** (a --) **a** étant l'adresse d'un champ nom dans le dictionnaire **.ID** imprime ce nom.
- **.OK** (--) Imprime le messate ** ok** suivit d'un **CR**.
- **.R** (i u --) Imprime l'entier i sur une largeur de colonne de u carctère. Si la chaîne résultant de la conversion de i est plus courte que u le remplissage est fait avec des espaces.
- **.S** (--) Imprime le contenu de la pile des paramètres.
- **/** (i1 i2 -- i3) i3 est le quotient entier arrondie à l'entier le plus petit résultant de i1/i2.
- **/MOD** (i1 i2 -- u i3) i3 est le quotient et u le reste de la division entière arrondie à l'entier le plus petit de i1/i2.
- **0** (-- 0) Constante numérique **0**.
- **0<** (i -- f) Retourne vrai (-1) si i<0 sinon retourne 0.
- **1** (--) Constante numérique **1**.
- **1+** (i1 -- i2) Incrémente i1 pour donner i2.
- **1-** (i1 -- i2) Décrémente i1 pour donner i2.
- **2!** (nd a --) Dépose un entier double à l'adresse a.
- **2*** (i1 -- i2) Multiplie par i1 par 2.
- **2+** (i1 -- i2) Ajoute 2 à i1.
- **2-** (i1 -- i2) Soustrait 2 à i1.
- **2/** (i2 -- i2) divise i2 par 2.
- **2@** (a -- nd) Empile l'entier double qui est à l'adresse a.
- **2DROP** (n1 n2 --) Jette les 2 éléments au sommet de la pile.
- **2DUP** (n1 n2 -- n1 n2) Duplique les 2 éléments au sommet de la pile.
- **:** (-- ; <string>) Débute la création d'une nouvelle définition dans le dictionnaire. **<string>** est le nom de cette nouvelle définition. Passe en mode compilation.
- **;** (--) Complète la définition d'un nouveau mot et repasse en mode interprétaion.

- **<** (i1 i2 -- f) Empile la valeur booléenne i1<i2.
- **<#** (--) Débute la conversion d'un entier en chaîne de caractère.
- **=** (i1 i2 -- f) Empile la valeur booléenne indiquant si i1=i2
- **>CHAR** (c -- c) Filtre les caractères de contrôle ASCII pour les remplacer par un **_**.
- **>IN** (-- a) Empile l'adrese de la variable système **>IN** qui est le pointeur de l'analyseur lexical.
- **>NAME** (a1 -- a2|0) Retourne l'adrresse du champ nom à partir de l'adresse du champ code d'une entrée du dictionnaire. a1 est le *code address* et a2 est le *name address*. Si le champ code est invalide retourne **0**.
- **>R** (n -- R: n) Envoie n sur la pile des retours.
- **?** (a --) Imprime l'entier à l'adresse a.
- **?DUP** (n -- n n | 0) Duplique n seulement si <> 0.
- **?KEY** (-- c -1 | 0) Vérifie s'il y a un caractère de disponible en provenance du terminal. Si oui retourne le caractère **c** et **-1** sinon retourne **0**.
- **?STACK** (--) Vérifie si la pile des arguments est en état sous-vidée *(underflow)*. Un abadon avec message d'erreur se produit dans ce cas.
- **?UNIQUE** (b -- b) Vérifie si le nom pointé par **b** existe déjà dans le dictionnaire. Affiche un messag d'avertissement s'il ce nom est déjà dans le dictionnaire. Ça signifit qu'on est en train de redéfinir un mot qui est déjà dans le dictionnaire.
- **@** (a -- n) Empile l'entier qui est à l'adresse **a**.
- **@EXECUTE** (a --) **a** est un pointeur vers l'adresse d'un code exécutable. Cette adresse est empilée pour être exécutée immédiatement.
- **ABORT** (nx --) Abandon avec vidage de la pile et du TIB. Est appellé par **ABORT"**.
- **ABORT"** (f --) Si l'indicateur est vrai affiche le message litéral qui suit et appelle **ABORT**.
- **ABS** (i1 -- u) Retourne la valeur absolue de i1.
- **ACCEPT** (b u1 -- b u2) Effectue la lecture d'une ligne de texte dans le **TIB**. **b** est l'adresse du **TIB u1** est la longueur du **TIB** et **u2** est le nombre de caractères reçus dans le **TIB**.
- **AFT** (a1 -- a1 a2) Mot compilant Utilisé dans une boucle FOR..AFT..THEN..NEXT. Pendant la compilation compile un saut avant après le **THEN**.
- **AGAIN** (--) Marque la fin d'une boucle BEGIN..AGAIN.
- **AHEAD** (-- a) Compile un saut avant inconditionnel. *a* est l'adresse de la fente où sera insérée l'adresse du saut ultérieurement lors du processus de compilation.
- **ALLOT** (u --) Alloue **u** octets dans l'espace RAM. Avance le pointeur **VP** de **u** octets.

- **AND** (n1 n2 -- n3) Opération bit à bit ET.
- **AUTORUN** (-- ; <string>) Enregistre dans la variable système persistante **APP_RUN** l'adresse d'exécution du programme qui doit-être exécuté au démarrage. Par défaut il s'agit du *ca* du mot **hi**.
- **BASE** (--) Variable système qui contient la base numérique utilisée pour la conversion des entiers en chaîne de caractères.
- **BEGIN** (-- a) Compile le début d'une boucle BEGIN..UNTIL|AGAIN. *a* indique l'adresse où doit se faire le saut arrière pour répéter la boucle.
- **BL** (-- c) Empile le caractère ASCII *space* i.e. 32.
- **BRANCH** (--) Compile un saut inconditionnel avec une adresse litérale. En *runtime* ce saut est toujours effectué.
- **BYE** (--) Exécute l'instruction machine **HALT** pour mettre le MCU en mode suspendu. Dans ce mode l'oscillateur et arrêter et le MCU dépense un minimum d'énergie. Seul un *reset* ou une interruption externe peut réactivé le MCU. Si le MCU est réanimé par une interruption après l'exécution de celle-ci l'exécution se poursuit après l'instruction **HALT**.
- **C!** (c a --) Dépose le caractère *c* à l'adresse *a*. Sur la pile *c* occupe 2 octets mais en mémoire il n'occupe qu'un octete.
- **C,** (c --) Compile le caractère qui est au sommet de la pile.
- **C@** (a -- c) Empile l'octet qui se trouve à l'adresse *a*.
- **CALL,** (a --) Compile un appel de sous-routine dans la liste d'une définition. *a* est l'adresse de la sous-routine.
- **CMOVE** (a1 a2 u --) Copie *u* octets de *a1* vers *a2*.
- **COLD** (--) Réinitialisation du système. Toute la mémoire RAM est remise à **0** et les pointeurs de piles sont réinitialisés ainsi que les variables système.
- **COMPILE** (--) Compile un appel de sous-routine avec adresse litérale.
- **CONSTANT** (n -- ; <string>) Compile une constante dans le dictionnaire. *n* est la valeur de la constante dont le nom est **<string>**.
- **CONTEXT** (-- a) Empile l'adresse de la variable système CNTXT. Cette variable contient l'adresse du point d'entré du dictionnaire.
- **COUNT** (b -- b u) Empile la longueur de la chaîne comptée *b* et incrémente *b*.
- **CP** (-- a) Empile l'adresse de la variable système **CP** qui contient l'adresse du début de l'espace libre en mémoire flash.
- **CR** (--) Envoie le caractère ASCII **CR** au terminal.
- **CREATE** (-- ; <string>) Compile le nom d'une nouvelle variable dans le dictionnaire. **<string>** est le nom de la nouvelle variable. Les variables sont initialisées à **0**.

- **DCONST** (d -- ; <string>) Création d'une constante de type entier double.
- **DECIMAL** (--) Affecte la valeur **10** à la variable système **BASE**.
- **DEPTH** (-- u) retourne le nombre d'élément qu'il y a sur la pile.
- **DI** (--) Désactive les interruptions en exécutant l'instruction machine **SIM**.
- **DIGIT** (u -- c) Convertie le chiffre *u* en caractère ASCII.
- **DIGIT?** (c base -- u f) Converti le caractère *c* en chiffre correspondant dans la *base*. L'indicateur *f* indique si *c* est bien dans l'intervalle {0..base-1}.
- **DNEGATE** (d1 -- d2) Négation arithmétique de l'entier double *d1*.
- **DROP** (n --) Jette l'élément qui est au sommet de la pile.
- **DUMP** (a u --) Affiche en hexadécimal le contenu de la mémoire débutant à l'adresse *a*. *u* octets arrondie au multiple de 16 supérieur sont affichés. Chaque ligne affiche 16 octets suivit de leur représentation ASCII.
- **DUP** (n -- n n) Empile une copie de l'élément au sommet de la pile.
- **EI** (--) Active les interruptions en exécutant l'instruction machine **RIM**.
- **ELSE** (a1 -- a2) Compile l'adresse du saut avant dans la fente *a1* laissée sur la pile par le **IF** qui indique où doit se faire le saut avant pour exécuter une condition *fausse*. Laisse *a2* sur la pile qui est l'adresse de la fente qui doit-être comblée par le **THEN** et qui permet un saut avant après le **THEN** lors que la condition *vrai* est exécutée.
- **EMIT** (c --) Envoie vers le terminal le caractère *c*.
- **ERASE** (b u --) Met à zéro *u* octets à partir de l'adresse *b*.
- **EVAL** (--) Interprète le texte d'entrée.
- **EXECUTE** (a --) Exécute le code à l'adresse *a*.
- **EXTRACT** (n1 base -- n2 c) Extrait le chiffre le moins significatif de *n* et le converti en caractère ASCII *c*. *n2=n1/base*.
- **FC-XOFF** (--) Envoie du caractère ASCII XOFF (19) au terminal. Il s'agit du caractère de contrôle de flux logiciel selon le protocole XON/XOFF. Lorsque le terminal reçoit ce caractère il doit cesser de transmettre jusqu'à ce qu'il reçoive un caractère XON.
- **FC-XON** (--) Envoie du caractère ASCII XON (17) au terminal. Il s'agit du caractère de contrôle de flux logiciel selon le protocole XON/XOFF. Indique au terminal qu'il peut reprendre la transmission.
- **FILL** (b u c --) Remplie *u* octets de la mémoire RAM à partir de l'adresse *b* avec le caractère *c*.
- **FIND** (a va -- ca na | a 0) Recherche le nom pointé par *a* dans le dictionnaire à partir de l'entrée indiquée par *va*. Si trouvé retourne *ca* l'adresse d'exécution. *na* l'adresse du champ nom. En cas d'échec retourne *a* et *0*.

- **FOR** (n+ --) Initialise une boucle FOR..NEXT. *n+* est un entier positif. La boucle se répète *n+1* fois.
- **FORGET** (-- ; <string>) Supprime du dictionnaire la définition **<string>** ainsi que toutes celles qui ont étées créées après celle-ci. Ne supprime que les définitions en mémoire FLASH. Pour les définitions en mémoire RAM il faut faire un **REBOOT**.
- **FREEVAR** (na --) *na* étant l'adresse du champ nom d'une variable **FEEVAR** réinitialise le pointeur **VP** à cette adresse. Toute allocation de mémoire RAM qui suit cette adresse est perdu.
- **HERE** (-- a) Retourne la valeur de la variable système **VP**.
- **HEX** (--) Sélectionne la base numérique hexadécimal. Dépose la valeur **16** dans la variable système **BASE**.
- **HLD** (-- a) Empile l'adresse de la variable système **UHLD**
- **HOLD** (c --) Insère le caractère *c* dans la chaîne de sortie. HOLD est utilisé dans la conversion des entiere en chaîne.
- **I** (-- n+) Empile le compteur d'une boucle **FOR..NEXT**.
- **I:** (--) Débute la compilation d'une routine d'interruption. Les routines d'interruptions n'ont pas de nom et ne sont pas inscrite dans le dictionnaire.
- **I;** (-- ad) Termine la compilation d'une routine d'interruption. *ad* est l'adresse de la routine d'interruption tel qu'elle doit-être inscrite dans le vecteur d'interruption.
- **IF** (f --) Vérifie la valeur de l'indicateur booléen *f* et exécute le code qui suis le **IF** si cette indicateur est *vrai* sinon saute après le **ELSE** ou le **THEN**.
- **IMMEDIATE** (--) Active l'indicateur **IMMED** dans l'entête de dictionnaire du dernier mot qui a été compilé. Habituellement invoqué juste après le **;**.
- **INIT-OFS** (--) Initialise la variable système **OFFSET** au début d'une nouvelle compilation. L'offset est la distance entre les valeurs des variables **CP** et **VP** Lorsque la variable système **TFLASH** est à zéro **OFFSET** est initialisé à zéro. **OFFSET** est utilisé par le compilateur pour déterminer les adresses absolues à utiliser dans les instructions de saut **BRANCH** et **?BRANCH**.
- **KEY** (-- c) Attend la réception d'un caractère du terminal. Empile le caractère *c*.
- **KTAP** (c --) Utilisé par **ACCEPT** pour Traiter les caractères de contrôles reçu du terminal. Les les caractères ASCII **CR** et **BS** sont traités les autres sont remplacés par un **BLANK**.
- **LAST** (-- a) Empile l'adresse de la variable système **LAST**.
- **LITERAL** (n --) Compile *n* comme entier litéral. En *runtime* **DOLIT** est invoqué pour remettre sur la pile la valeur *n*.
- **LSHIFT** (i1 n+ --) Décalager vers la gauche de *i1 n+* bits. Les bits à droites sont mis à zéro.
- **M*** (n1 n2 -- d) Multiplication *n1*n2* conservé en entier double *d*.

- **M/MOD** (d n -- r q) Division de l'entier double *d* par l'entier simple *n*. Empile le reste et le quotient. Le quotient est arrondie à l'entier le plus petit.
- **MAX** (n1 n2 -- n) Empile le plus grand des 2 entiers.
- **MIN** (n1 n2 -- n) Empile le plus petit des 2 entiers.
- **MOD** (n1 n2 -- n) Retourne le reste de la division entière arrondie au plus petit entier. *n* est toujours **≥0**.
- **MSEC** (-- u) Retourne la valeur du compteur de millisecondes. Il s'agit d'un compteur qui est incrémenté chaque milliseconde par une interruption du TIMER4.
- **NAME>** (na -- ca) Retourne l'adresse du **code** correspondant à l'entrée du dictionnaire avec le *champ nom* **ca**. Donne une valeur erronnée si **na** n'est pas une entrée valide dans le dictionnaire.
- **NAME?** (b -- ca na | b 0) Recherche le nom **b** dans le dictionnaire. Si ce nom existe retourne l'adresse du code **ca** et l'adresse du champ nom **na**. Si le nom n'est pas trouvé retourne **b** et **0**.
- **NEGATE** (i1 -- i2) Empile la négation arithmétique de **i1**.
- **NEXT** (a --) Mot immédiat qui compile la fin d'une boucle **FOR-NEXT**. **a** est l'adresse du début de la boucle et est compilée comme saut arrière.
- **NOT** (i1 -- i2) **i2** est le complément unaire de **i1**. Autrement dit tous les bits de **i1** sont inversés.
- **NUF?** (-- f) Vérifie si un caractère a été reçu du terminal. Si aucun caractère reçu retourne **F**. Si un caractère a été reçu jette ce caractère et appel **KEY** pour attendre le prochain caractère. Si le prochain caractère reçu est **CR** retourne **T** sinon retourn **0**. Est utilisé pour faire une pause dans un défilement d'écran.
- **NUMBER?** (b -- i T | b F) Essaie de convertir la chaîne *b* en entier. Si la convertion réussie l'entier **i** et **T** sont retournés. Sinon **b** et **F** sont retournés.
- **OFFSET** (-- a) Variable système indiquant la distance enter **CP** et **VP**. Utilisé pour calculer les adresses de saut lors de la compilation.
- **OR** (n1 n2 -- n3) **n3** est le résultat d'un OU bit à bit entre **n1** et **n2**.
- **OVER** (n1 n2 -- n1 n2 n1) Copie le second élémente de la pile au sommet.
- **OVERT** (--) Ajoute le dernier mot compilé au début de la liste chaîné du dictionnaire.
- **PACK0** (b u a -- a) Construit une chaîne comptée à partir de *b* et de *u* qui est le nombre de caractères à copier dans *a*. Garde l'adresse de la nouvelle chaîne.
- **PAD** (-- a) Empile l'adresse du tampon de travail **PAD**.
- **PARSE** (c -- b u ;) Analyseur lexical. parcourt le flux d'entrée à la recherche de la prochaîne unité lexicale. *c* est le caractère délimiteur. *b* est l'adresse de la chaîne trouvée et *u* sa longueur.
- **PAUSE** (u --) Suspend l'exécution pour une durée de *u* millisecondes.

- **PICK** (nx j -- nx nj) Copie au sommet de la pile le jième élément de la pile. *j* est d'abord retiré de la pile ensuit les éléments sont compté à partir du sommet vers le fond de la pile. l'Élément au sommet est l'élément *0*. Donc **0 PICK** est l'équivalent de **DUP** et **1 PICK** est l'équivalent de **OVER**. Le nombre d'éléments sur la pile doit-être ≥j+1.
- **PRESET** (--) Vide la pile des arguments et le TIB avant d'invoquer **QUIT**.
- **QUERY** (--) Lecture d'une ligne de texte du terminal dans le TIB. La lecture se termine à la réception d'un caractère **CR**. Le nombre de caractères dans le TIB est dans la variable systèmE **#TIB**.
- **QUIT** (--) Il s'agit de l'interpréteur de texte, c'est à dire l'interface entre l'utilisateur et le système. **QUIT** appel en boucle **QUEYR** et **EVAL**.
- **R>** (n -- R: -- n) La valeur au sommet de la pile des arguments est transférée sur la pile des retours.
- **R@** (-- n) La valeur au sommet de la pile des retours est copié sur la pile des arguments.
- **RAMLAST** (-- a) Empile l'adresse de la variable système **RAMLAST**.
- **RANDOM** (u1 -- u2) Retourne un entier pseudo aléatoire dans l'intervalle **0≤u2<u1**.
- **REBOOT** (--) Rédamarrage du MCU. A le même effet que d'enfoncer le bouton *RESET* sur la carte.
- **REPEAT** (--) Termine un boucle de la forme **BEGIN-WHILE-REPEAT**. Le branchement s'effectue après le **BEGIN**.
- **ROT** (n1 n2 n3 -- n2 n3 n1) Rotation des 2 éléments supérieurs de la pile.
- **RP!** (n --) Initialise le pointeur de la pile des retours avec la valeur **n**.
- **RP@** (-- n) Empile la valeur du pointeur de la pile des retours.
- **RSHIFT** (n1 u -- n2) Décale *n1* de *u* bits vers la droite. Les bits à gauche sont remplacés par **0**. Même effet que l'opérateur **C >>**.
- **S>D** (i -- d) Convertie un entier simple en entier double.
- **SAME?** (a1 a2 u -- a1 a2 f) Compare *a1* et *a2* sur *u* octets et retourne **0** s'ils sont identiques sinon retourne u-1.
- **SEED** (u --) Initialise le générateur pseudo-aléatoire.
- **SET-ISP** (u1 u2 --) Fixe le niveau de priorité logicielle d'interruption du vecteur **u2** avec la valeur **u1** {1,2,3}. Le niveau maximal est **3** et c'est la valeur par défaut.
- **SIGN** (i --) Si i<0 alors préfixe la chaîne numérique du caractère **-**. Est utilisé lors de la conversion d'un entier en chaîne de caractères.
- **SP!** (u --) Initialise le pointeur de la pile des arguments.
- **SP@** (-- u) Empile la valeur du pointeur des arguments.
- **SPACE** (--) Envoie un caractère ASCII *espace* au terminal.

- **SPACES** (n+ --) Envoie **n+** caractères ASCII *espace* au terminal.
- **STR** (i -- b u) Converti en chaîne de caractère l'entier **i**. **b** est la chaîne résultatnte et **u** la longueur de cette chaîne.
- **SWAP** (n1 n2 -- n2 n1) Inverse l'ordre des 2 éléments au sommet de la pile.
- **TAP** (a1 c -- a2) Envoie le caractère **c** au terminal et le dépose dans le tampon à l'adresse **a1**. Incrémente **a1** pour donner **a2**.
- **TBUF** (-- a) Empile l'adresse du tampon de 128 octets qui sert à l'écriture par bloc dans la mémoire persistante.
- **TFLASH** (-- a) Empile l'adresse de la variable système **TFLASH** qui indique la destination de la compilation. Cette variable est modifiée par **TO-FLASH** et **TO-RAM**.
- **THEN** (--) Termine une boucle **IF-ELSE-THEN.
- **TIB** (-- a) Empile l'adresse du *Transaction Input Buffer* qui est le tampon qui accumule les caractes lus par **ACCEPT**.
- **TIMEOUT?** (-- f) Vérifie l'état du compteur à rebour **TMER** et retourne *vrai* s'il est à zéro sinon retourne *faux*.
- **TIMER** (u --) Initialise le compteur à rebour. Ce compteur est décrémenter à chaque milliseconde jusqu'à ce qu'il atteigne zéro.
- **TMP** (-- a) Empile l'adresse de la variable système **TMP**.
- **TO-FLASH** (--) Met à **-1** la valeur de la variable système **TFLASH**. Indiquant ainsi que les mots doivent-être compilés dans la mémoire flash.
- **TO-RAM** (--) Met à **0** la valeur de la variable sytème **TFLASH**. Indiquant ainsi que les mots doivent-être compilés dans la mémoire RAM. C'est mots sont perdus lors d'une réinitialisaton du système avec **COLD**,**REBOOT**,un *RESET* ou une perte d'alimentation de la carte.
- **TOKEN** (-- a ; <string>) Extrait le prochain mot du TIB.
- **TYPE** (b u --) Envoie *u* caractère au terminal à partir de l'adresse *b*.
- **U.** (u --) Imprime l'entier non signé *u*.
- **U.R** (u n+ --) Imprime l'entier non signé *u* sur *n+* colonnes aligné à droite avec remplissage par *espace*.
- **U<** (u1 u2 -- f) Comparaison de 2 entiers non signés. Retourne *vrai* si *u1<u2*. Sinon retourne *faux*.
- **UM*** (u1 u2 -- ud) Multiplication de 2 entiers non signés et retourne le résultat comme entier non signé double.
- **UM+** (u1 u1 -- ud) Additionne 2 entiers non signés et retourne la somme comme entier non signé double.

- **UM/MOD** (ud u -- ur uq) Division non signé de l'entier double *ud* par l'entier simple non signé *u*. Retourne le reste *ur* et le quotient *uq*.
- **UNTIL** (f --) Termine une boucle **BEGIN - UNTIL**. La boucle se termine quand **f** est *vrai*.
- **VARIABLE** (-- <string>) Crée une nouvelle variable de nom **<string>**. Cette variable est initialisée à zéro.
- **VP** (-- a) Empile l'adresse de la variable système **VP**.
- **WHILE** (f --) Condition de contrôle d'une boucle **BEGIN - WHILE - REPEAT**. la boucle se poursuit tant que **f** est *vrai*.
- **WITHIN** (u ul uh -- f) Retourne **f** indiquant si **ul ≤ u <uh**
- **WORD** (c -- a) Extrait le prochain mot du **TIB** et le copie à la fin du dictionnaire. **c** est le caractère séparateur de mot et **a** est l'adresse où le mot a été copié.
- **WORDS** (--) Imprime la liste de tous les mots du dictionnaire.
- **XOR** (n1 n2 -- n3) **n3** est le résultat d'un ou exclusif bit à bit entre **n1** et **n2**.
- **[** (--) Initialise le vecteur EVAL en mode *interprétation*.
- **[COMPILE]** (-- <string>) Ce mot est utilisé à l'intérieur d'une définition pour compiler le mot suivant qui est un mot *immédiat* donc serait exécuté plutôt que compilé.
- **[N]?** (n+ - n T | a F) Affiche **[n+]?** puis attend la saisie d'un entier. Si un entier a été entré au terminal retourne l'entier et **T** sinon retounre l'adresse du token **a** et **F**. Ce mot est utilisé par **CTFILL** et **WTFILL**.
- **____** (--) Introduit un commentaire qui se termine à la fin de la ligne.
- **]** (--) Initialise le vecteur EVAL en mode *compilation*.
- **^H** (--) Envoie le caractère ASCII DEL (8) au terminal.
- **dm+** (a u -- a+u) Affiche l'adresse **a** suivit de **u** octets de mémoire à partir de **a**. Retourne l'adresse incrémenté. Invoqué par **DUMP**.
- **hi** (--) Application par défaut appellée par **COLD** et qui imprime le message *stm8eForth v3.0*.
- **parse** (b1 u1 c -- b2 u2 delta ;) **c** étant le séparateur de mots saute par dessus les **c** jusqu'au premier caractère différent de **c** ensuite avance jusqu'au premier caractère **c**. **b2** est le début du mot, **u2** sa longueur et **delta** est la distance **b2-b1**.

Index

Module flash.asm

Ce module définit le vocabulaire nécessaire pour écrire dans la mémoire persistante FLASH,EEPROM et OPTION. Il y a aussi des mots pour modifier les vecteurs d'interruptions.

- **BUF>ROW** (ud --) Écris le contenu du tampon **ROWBUFF** dans la mémoire flash en utilisant l'opération d'écriture par bloc du MCU.
- **CHKIVEC** (a --) Toutes les adresses de destination des vecteurs d'interruptions sont comparés à *a*. Tous les vecteurs qui pointent vers une adresse <=*a* sont réinitialisés à la valeur par défaut. Ce mot est invoqué par **PRISTINE**.
- **EE!** (n ud --) Écriture en mémoire persistante (FLASH|EEPROM) de la valeur *n*. *ud* et un entier double non signé représentant l'adresse destination.
- **EE,** (n --) Compile en mémoire FLASH l'entier *n*.
- **EEC!** (c ud --) Écris en mémoire persistante le caractère *c*. *ud* est l'adresse destination sous-forme d'entier double non signé.
- **EEC,** (c --) Compile en mémoire FLASH le caractère *c*.
- **EE-CREAD** (-- c) Empile le caractère à l'adresse pointé par **FPTR** et incrément le pointeur.
- **EE-READ** (-- n) Empile l'entier pointé par **FPTR** et incrément le pointeur de 2.
- **EEP-CP** (-- ud) Empile l'adresse de la variable système persistante **APP_CP** . *ud* est un entier double non signé.
- **EEP-LAST** (-- ud) Empile l'adresse de la variable système persistante **APP_LAST**.
- **EEP-RUN** (-- ud) Empile l'adresse de la variable système persistante **APP_RUN**.
- **EEP-VP** (-- ud) Empile l'adresse de la variable système persitante **APP_VP**
- **EEPROM** (-- ud) Empile l'adresse de base de l'EEPROM.
- **FAR@** (ad -- n) Empile l'entier qui se trouve à l'adresse étendue *ad*. Utile pour lire la mémoire flash au delà de 65535. Sur la version **NUCLEO** seulement.
- **FADDR** (a -- ad) Convertie l'adresse 16 bits *a* en adresse 32 bits *ad*. Sur la version **NUCLEO** seulement.
- **FARC@** (ad --) Empile l'octet qui se trouve à l'adresse étendue *ad*. Utile pour lire à mémoire flash au delà de 65535. Sur la version **NUCLEO** seulement.
- **FMOVE** (-- a) Déplace le dernier mot compilé de la mémoire RAM vers la mémoire FLASH. Retourne le pointeur de code mis à jour **a**.
- **FP!** (ad --) Initialize la variable système **FPTR** avec la valeur **ad**. Le far pointer est utilisé pour les opérations d'écriture en mémoire persistante.
- **IFMOVE** (-- a) Transfert routine d'interruption qui vient d'être compilée vers la mémoire flash. **a** est la valeur mise à jour du pointeur de code **CP**.
- **INC-FPTR** (--) Incrémente la variable système **FPTR**.
- **LOCK** (--) Verrouille l'écrire dans la mémoire persistante (FLASH et EEPROM).

- **PRISTINE** (--) Nettoie le système de toutes les modifications effectuées par l'utilisateur. Le système Forth se retrouve alors dans son état initial avant toute intervention de l'utilisateur.
- **PTR+** (u --) Incrémente **FPTR** d'une valeur arbitraire **u**.
- **RAM>EE** (ud a u1 -- u2) Écris dans la mémoire persistance **u1** octets de la mérmoire RAM à partir de l'adresse **a** vers l'adresse **ud**. Cependant l'écriture est limitée aux limites du bloc 128 octets qui contient l'adresse **ud**. Si **ud+u1** dépasse la limite l'écriture s'arrête à la fin du bloc. Retourne **u2** le nombre d'octets réellement écris.
- **RFREE** (a -- u) **u** est le nombre d'octets libres dans le bloc qui contient l'adresse **a**. En fait u=128-a%128. 128 étant la longueur d'un bloc FLASH pour les MCU **STM8S105C6** et **STM8S208RB.**
- **ROW-ERASE** (ud --) Efface le bloc de mémoire persistante contentant l'adresse **ud**.
- **ROW>BUF** (ud --) Copie le bloc de mémoire persistante contenant l'adresse **ud** vers le tampon système **TBUF**.
- **RST-IVEC** (u --) Réinitialise le vecteur d'interruption #**u** à sa valeur par défaut.
- **SET-IVEC** (ud u --) Initialise le vecteur d'interruption **u** avec l'adresse **ud** qui et l'adresse d'une routine de service d'interruption.
- **SET-OPT** (c u --) Écris le caractère **c** dans le registre d'OPTION **u**.
- **UNLKEE** (--) Déverouille pour l'écriture la mémoire EEPROM.
- **UNLKFL** (--) Déverouille pour l'écriture la mémoire FLASH.
- **UNLOCK** (--) Selon l'adresse contenue dans la variable système **FPTR** déverrouille la mémoire FLASH ou EEPROM.
- **UPDAT-CP** (--) Met à jour la variable système persistante **EEP-CP** à partir de la variable **CP**.
- **UPDAT-LAST** (--) Met à jour la variable système persistante **EEP-LAST** à partir de la variable **LAST**.
- **UPDAT-PTR** (--) Met à jour les différentes variables système persistantes à partir des valeurs de leur correspondantes non persistantes.
- **UPDAT-RUN** (a --) Met à jour la variable système persistante **EEP-RUN**. **a** est la nouvelle adresse du programme à exécuter au démarrage.
- **UPDAT-VP** (--) Met à jour la variable système persistante **EEP-VP** à partir de la valeur de **VP**.
- **WR-BYTE** (c --) Écris un octet dans la mémoire persistante à l'adresse indiquée par la variable système **FPTR**. Incrémente **FPTR**.
- **WR-ROW** (a ud --) Écriture d'un bloc de 128 octets dans la mémoire persistante. **a** est l'adresse RAM qui contient les données à écrires et **ud** l'adresse destination. Si **ud** n'est pas alignée sur un bloc de 128 octets il le sera en mettant les 7 bits les moins significatifs à zéro. Dans la version **DISCOVERY** **ud** est remplacé par une adresse de type *entier simple non signé*.

- **WR-WORD** (n --) Écris un entier 16 bits dans la mémoire persitante à l'adresse pointée par **FPTR**. Incrémente **FPTR**.

Index

Module const_ratio.asm

Ce module définie des constantes arithmétiques constituée de 2 entiers dont le rapport approche un nombre irrationnel. Ces valeurs sont utilisées dans les opérations de *scaling*.

- **12RT2** (-- 26797 25293) Empile 2 constantes dont le rapport s'approche de la racine 12ième de 2.
- **E** (-- 28667 10546) Le rapport des 2 entiers empilés donne une valeur approximative de la base du logarithme népérien.
- **LN2S** (-- 485 11464) Empile 2 entiers dont le rapport approxime la valeur de **10^3 * ln(2)/2^{14}.**
- **LOG2S** (-- 2040 11103) Empile 2 entiers dont le rapport approxime la valeur de **10^4 * log(2)/2^{14}.**
- **PI** (-- 355 113) Empile 2 entiers dont le rapport se rapproche de la constante **PI**.
- **SQRT10** (-- 22936 7253) Empile deux entiers dont le rapport approxime la racine carrée de 10.
- **SQRT2** (-- 19601 13860) Empile deux entiers dont le rapport approxime la racine carrée de 2.
- **SQRT3** (-- 18817 10864) Empile deux entiers dont le rapport approxime la racine carrée de 3.

Index

Module ctable.asm

Ce module définit le vocabulaire servant à la création et l'initialisation de tables de constantes persistantes. Deux types de tables peuvent-être créées, les tables d'octets et les tables d'entiers 16 bits. Ce module dépend du module **flash.asm**.

- **CALLOT** (u -- ad) Alloue **u** octet dans l'espace **code** de la mémoire FLASH. **ad** est l'adresse de base de ce bloc de mémoire.
- **CTABL@** (u ad -- c) Empile l'octet **c** d'indice **u** de la table dont l'adresse de base est **ud**. Les indices de tables commence à zéro.
- **CTABLE** (u -- ad ; <string>) Création d'une table de **u** octets dans l'espace code. **<string>** est le nom de la table. **ad** est l'adresse de base de la table nouvellement créée. **ATTENTION:** même si la variable **TFLASH** est à *faux* l'espace est toujours alloué dans la mémoire FLASH. Par contre la variable qui référence la table sera créée dans le dictionnaire en mémoire RAM. Donc après un redémarrage l'espace alloué pour cette table est toujours réservé en mémoire FLASH mais la variable de référence est perdue. Donc les tables de constantes ne devrait-être créée qu'en mode **TO-FLASH** pour que la référence soit préservée après redémarrage.
- **CTINIT** (ad --) Outil d'initialisation des tables de constantes de type octet. **ad** est l'adresse de base de la table. L'utilisateur doit saisir au terminal une valeur entière dans l'intervalle {0..255} pour

chaque entrée de la table. L'initialisation se termine lorsqu'une ligne vide ou une valeur non numérique est saisie au terminal.

- **WTABL@** (u ad -- n) Empile l'entier **n** d'indice **u** de la table dont l'adresse de base est **ud**. Les indices de tables commence à zéro.
- **WTABLE** (u -- ad ; <string>) Création d'une table de **u** entiers dans l'espace code. **<string>** est le nom de la table. **ad** est l'adresse de base de la table nouvellement créée.
- **WTINIT** (ad --) Outil d'initialisation d'une table de contantes entiers dans la mémoire persistante. **ad** est l'adresse de la table. L'utilisateur doit saisir au terminal une valeur entière dans l'intervalle {-32768..32767} pour chaque entrée de la table qu'il désire initialiser. L'initialisation se termine lorsqu'une ligne vide ou une valeur non numérique est saisie au terminal. **ATTENTION:** même si la variable **TFLASH** est à *faux* l'espace est toujours alloué dans la mémoire FLASH. Par contre la variable qui référence la table sera créée dans le dictionnaire en mémoire RAM. Donc après un redémarrage l'espace alloué pour cette table est toujours réservé en mémoire FLASH mais la variable de référence est perdue. Donc les tables de constantes ne devrait-être créée qu'en mode **TO-FLASH** pour que la référence soit préservée après redémarrage.
- **[N]?** (n+ -- n T | a F) Ce mot est invoqué par **CTINIT** et **WTINIT** pour effectuer la saisie au terminal des entiers qui servent à initialiser une table de constantes. Pour chaque entier requit l'invite **[n+]?** est affichée au terminal en attente de la saisie par l'utilisateur d'un entier. **n+** est l'indice de table qui recevra l'entier saisie. Ce mot retourne l'entier saisie **n** et **T** ou **a** et **F** Si la valeur saisie n'est pas un entier.

Index

Module double.asm

Le fichier principal contient quelques opérations sur les entiers double de 32 bits. Ce module en ajoute d'autres.

- **DBL-VER** (--) imprime la version de la librairie
- **NUMBER?** nbsp; (a -- s -1 |d -2 | a F) **a** pointe une chaîne de caractère qui sera éventuellement convertie en entier simple ou double.
    - Si un entier dont la valeur est dans l'intervalle {-32768..32767}, la valeur de cet entier **s** est retourné avec au sommet de la pile la valeur **-1**.
    - Si un entier trop hors de l'intervalle d'un entier simple est trouvé la valeur de l'entier double **d** est retournée ainsi que **-2** sommet de la pile. Si aucun entier n'est trouvé l'adresse de la chaîne **a** et la valeur faux **F** est retourné.
- **DABS** (d -- d) Retourne la valeur absolue de l'entier double **d**.
- **DSIGN** (d -- d 0|-1) Retourne l'entier double **d** ainsi que son signe **0|-1**.
- **DS/MOD** (ud us -- ur udq) Divise l'entier double non signé **ud** par l'entier simple non signé **us** et retourne le quotient comme entier double non signé **udq** et le reste comme entier simple non signé **ur**.

- **D#** (d1 -- d2) dépose dans le tampon pointé par **HOLD** le caractère représentant le digit le moins significatif de l'entier double **d1** et retourne le quotient **d2**
- **D#S** (d -- s) Convertie l'entier double **d** en déposant les caratères dans le tampon pointé par **HOLD** jusqu'à ce que le quotient devienne nulle. **s** est un entier simple de valeur nulle.
- **D.** (d --) Imprime l'entier double **d** au terminal.
- **UDS*** (ud us -- uprod) Effectue le produit d'un entier double non signé **ud** par un entier simple non signé **us**. Retourne le produit entier double non signé **uprod**.
- **DS*** (d us -- prod) Effectue la multiplication d'un entier double **d** par un entier simple non signé **us**. Retourne le produit double signé **prod**.
- **2SWAP** (d1 d2 -- d2 d1) inverse l'ordre de 2 entiers double sur la pile.
- **DCLZ** (d -- u) Compte le nombre de bits à zéro de l'entier double **d** en partant du bit le plus significatif . S'arrête au premier bit à **1** rencontré. **u** est le nombre de bits à zéro {0..32}.
- **<2ROT** (d1 d2 d3 -- d3 d2 d1) Effectue la rotation des 3 entiers double au sommet de la pile en envoyant le sommet en 3ième position.
- **2ROT** (d1 d2 d3 -- d2 d3 d1) Effectue la rotation des 3 entiers double au sommet de la pile en ramenant le 3ième au sommet de la pile.
- **D0=** (d -- 0|-1) vérifie la valeur de l'entier double **d** et le remplace par **0** si sa valeur est non nulle et par **-1** si sa valeur est nulle.
- **D=** (d1 d2 -- f) compare les 2 entiers doubles **d1** et **d2** et les remplace par un indicateur booléen indiquant s'ils sont égaux.
- **D>** (d1 d2 -- f) compare les 2 entiers doubles **d1** et **d2** et les remplace par un indicateur booléen **f**. **f** est vrai si **d1** est plus grand que **d2**.
- **D<** (d1 d2 -- f) compare les 2 entiers doubles **d1** et **d2** et les remplace par un indicateur booléen **f**. **f** est vrai si **d1** est plus petit que **d2**.
- **D0<** (d -- f) compare l'entier double **d** avec zéro et le remplace par l'indicateur booléen **f**. **f** est vrai si d est plus petit que zéro.
- **2>R** (d -- R: d) Retire l'entier double **d** de la pile des arguments pour le déposé au sommet de la pile des retours.
- **2R>** (R: d -- d) Retire l'entier double **d** qui du sommet de la pile des retours et le dépose au sommet de la pile des arguments.
- **2R@** (-- d) Copie l'entier double qui se trouve au sommet de la pile des retours pour le déposer au sommet de la pile des arguments. Le contenu de **R:** n'est pas modifié.
- **2VARIABLE** <name> Crée une variable de type entier double portant le nom **<name>**.
- **2LITERAL** (d --) Compile l'entier double **d**. Ce mot ne peut-être utilisé qu'à l'intérieur d'une définition.

- **2OVER** (d1 d2 -- d1 d2 d1) Copie l'entier double **d1** au sommet de la pile.
- **D/2** (d -- d/2) Divise l'entier double **d** par 2.
- **D*2** (d -- d2) Multiplie l'entier souble **d** par **d2=d*2**.
- **DLSHIFT** (d u -- d2) Décale vers la gauche l'entier double **d** de **u** bits **d2=d*2^u**
- **DRSHIFT** (d u -- d2) Décale vers la droite l'entier double **d** de **u** bits le bit le plus significatif est remplacé par zéro.
- **D*** (d1 d2 -- d3) Produit de 2 entiers double **d3=d1*d2**.
- **UD/MOD** (ud1 ud2 -- dr udq) Division de 2 entiers doubles non signés avec quotient et reste retournés. **udq=ud1/ud2** et **dr=ud1 MOD ud2**.
- **D/MOD** (d1 d2 -- dr dq) Division de 2 entiers doubles avec quotient et reste retournés. **dq=d1/d2** et **dr=d1 MOD d2**
- **D/** (d1 d2 -- dq) Division de 2 entiers doubles signés, seul le quotient est retourné. **dq=d1/d2**.
- **D+** (d1 d2 -- d3) Somme de 2 entiers doubles. **d3=d1+d2**.
- **D-** (d1 d2 -- d3) Soustraction de 2 entiers doubles. **d3=d1-d2**.

index

Modules float.asm et float24.asm

L'une ou l'autre des librairies peut-être sélectionnée dans le fichier **inc/config.inc**. Le vocabulaire est le même pour les 2 librairies.

- **FLOAT-VER** (--) Imprime au terminal la version de la librairie.
- **FZE** (-- Z) Retourne l'État du bit **Z** indiquant un résulat nul lors de la dernière opération.
- **FNE** (-- N) Retourne l'état du bit **N** indiquant un résultat négatif lors de la dernière opération.
- **FOV** (-- f) Retourne l'état du bit **V** indiquant un débordement lors de la dernière opération.
- **FABS** (f#1 -- f#2) Retourne la valeur absolue de **f#1** .
- **FNEGATE** (f#1 -- f#2) Retourne l'inverse numérique de **f#1**.
- **F0=** (f#1 -- f) Retourne VRAI si **f#1** est nul.
- **F=** (f#1 f#2 -- f) compare **f#1** et **f#2** retourne vrai s'ils sont égaux.
- **F>** (f#1 f#2 -- f) Retourne VRAI si __f#1 > **f#2**.
- **F<** (f#1 f#2 -- f) Retourne VRAI si __f#1 < **f#2**.
- **F0<** (f#1 -- f) Retourne VRAI si **f#1** est négatif.

- **F>S** (f# -- s) Convertie un nombre en virgule flottante en entier. Les décimales sont tronquées. Si l'entier est plus grand que **32767** retourne **-32768**
- **S>F** (s -- f#) Convertie un entier en nombre à virgule flottante.
- **F/** (f#1 f#2 -- f#3) **f#3** = **f#1** / **f#2**.
- **F*** (f#1 f#2 -- f#3) **f#3** = **f#1** * **f#2**.
- **F-** (f#1 f#2 -- f#3) **f#3** = **f#1** - **f#2**.
- **F+** (f#1 f#2 -- f#3) **f#3** = **f#1** + **f#2**.
- **F@** (a -- f#) Empile la valeur d'une variable de type **FVAR** située à l'adresse **a**.
- **F!** (f# a --) Initialise la variable de type **FVAR** située à l'adresse **a** avec la valeur **f#**.
- **FVAR** (<var_name>) Création d'une variable de type virgule flottante portant le nom **var_name**.
- **FCONST** (<var_name>) Création d'une constante de type virgule flottante portant le nom **var_name**.
- **FLITERAL** (f# --) Compile le nombre en virgule flottante **f#**.
- **F.** (f# --) Imprime le nombre en virgule flottante **f#**. Si l'exposant est négatif ou si le nombre ne peut-être représenté avec moins de 10 décimales alog il est imprimé en format scientifique.
- **E.** (f# --) Imprime un nombre en format scientifique à moins qu'il est moins de 6 décimales.

index