Manuel de l'utilisateur de Tiny BASIC

Tiny BASIC pour STM8 est un langage simple qui cependant permet de configurer et d'utiliser tous les périphériques du microcontrôleur. La seule limitation est que les interruptions ne sont pas utilisée. Tiny BASIC lui-même n'utilise que les interruptions suivantes:

- TIMER4 Update pour le compteur de millisecondes
- Uart(1 ou 3) RX full, pour la réception des caractères du terminal.
- AWU pour la commande auto wake up.
- Sur la carte **NUCLEO_8S208RB**, le bouton **USER** déclenche l'interruption externe **EXTI4**.

L'objectif de ce manuel est de présenter les fonctionnalités du langage à travers des applications du microcontrôleur. Je n'ai pas définie toutes les constantes des registres du MCU dans le langage il est donc nécessaire de se réréfer au feuillet de spécifications ainsi qu'au manuel de référence du STM8S. Les manuel d'utilisateur des des cartes NUCLEO-8S208RB et NUCLEO-8S207K8 sont aussi utile.

Le réperoite **BASIC** contient plusieurs programmes qui peuvent servir d'exemples.

Il est aussi recommandé de lire en pré-requis de ce manuel la référence du langage Tiny BASIC

La commande **WORDS** affiche la liste complète des mots qui sont dans le dictionnaires.

```
>words
ABS     ADCON       ADCREAD     ALLOC       AND
ASC     AUTORUN     AWU     BIT     BRES
BSET        BTEST       BTOGL       BUFFER      BYE
CHAIN       CHAR        CONST       CR1     CR2
DATA        DDR     DEC     DIM     DIR
DO      DREAD       DROP        DWRITE      EDIT
EEFREE      EEPROM      END     ERASE       FCPU
FOR     FREE        GET     GOSUB       GOTO
HEX     I2C.CLOSE   I2C.OPEN    I2C.READ    I2C.WRITE
IDR     IF      INPUT       KEY     KEY?
LET     LIST        LOG2        LSHIFT      NEW
NEXT        NOT     ODR     ON      OR
PAD     PAUSE       PEEK        PICK        PINP
PMODE       POKE        POP     POUT        PRINT
PORTA       PORTB       PORTC       PORTD       PORTE
PORTF       PORTG       PORTI       PUSH        PUT
READ        REBOOT      REM     RESTORE     RETURN
RND     RSHIFT      RUN     SAVE        SIZE
SLEEP       STEP        STOP        TICKS       TIMEOUT
TIMER       TO      TONE        TRACE       UBOUND
UFLASH      UNTIL       USR     WAIT        WORDS
WRITE       XOR
107 words in dictionary

>
```

Pour la carte **NUCLEO-8S208RB** il y a 4 commandes de plus car le périphérique **SPI** est disponible.

exécution des programmes

Si une ligne de commande est saisie sans numéro de ligne elle est compilée et exécutée immédiatement. Par contre si le texte commence par un entier entre 1 et 32767 cette ligne est considérée comme faisant partie d'un programme et après sa compilation elle est insérée dans la zone texte réservée au progammes BASIC. Les programmes sont exécutés à partir de la mémoire RAM. Pour les cartes **NUCLEO-STM8S208RB** et **NUCLEO-STM8S207K8** il y a 6Ko de mémoire RAM une partie ce cette mémoire est utilisée par l'interpréteur et il reste environ 5561 octets disponibles pour les progammes. Les programmes sauvegardés en mémoire FLASH sont excécutés sur place.

exemple 1 blinky

Sur la carte il y a une LED indentifiée **LD2** ou **LD3**. Cette LED est connecté à la broche qui correspond au bit 5 du GPIO C. Cette GPIO est pré-configurée en mode sortie par le système Tiny BASIC. Pour contrôler son état il suffit donc de modifier l'éatt du bit 5 du registre **ODR** du GPIO C. Dans ce premier exemple nous allons faire clignoer cette LED au rythme de 1 fois par seconde. Le programme est interrompu en enfonçant n'importe quelle touche du terminal.

```
1 BLINK
5 ' Blink LED2 on card
10 DO BTOGL PORTC , BIT ( 5 ) PAUSE 500 UNTIL KEY?
20 LET A = KEY
30 BRES PORTC , BIT ( 5 )
40 END
```

Notez que vous pouvez saisir le texte aussi bien en minuscules qu'en majuscules. l'interpréteur convertie en majuscules.

Une autre méthode pour faire clignoter la LED est d'utiliser la commande **DWRITE** comme illustré dans l'exemple suivant:

```
5 ' CTRL+C pour arrêter le programme
7 ' clignote 3 fois par seconde
10 LET B = 1
20 FOR A = 0 TO 0 STEP 0 ' boucle infinie
30 DWRITE 13 , B ' la LED sur la broche D13
40 LET B = 1 - B
50 PAUSE 333
60 NEXT A
```

exemple 2 PWM logiciel

Dans cet exemple l'intensité de la LED est contrôlée par PWM logiciel.

```
    1 PWM.SOFT
    5 ' Software PWM, controle LD2 sur la carte
    7 GOSUB HELP
   10 LET R = 511 , S = 1 , N = 0 , P = 0 : ? R ;
   20 LOOP ' PWM loop
   22 IF K = P : LET N = N + 1 , S = N / 10 + 1
   24 IF K <> P : LET S = 1 , N = 0
   26 LET P = K , K = 0
   30 IF R : BSET PORTC , BIT ( 5 )
   40 FOR A = 0 TO R : NEXT A
   50 BRES PORTC , BIT ( 5 )
   60 FOR A = A TO 1023 : NEXT A
   70 IF KEY? : LET K = KEY : GOSUB UPPER
   72 IF ( K = ASC ( \D ) OR K = ASC ( \U ) ) AND K = P : LET N = N + 1 , S
= N / 10 + 1
   74 IF K = 0 OR K <> P : LET S = 1 , N = 0
   78 IF K = 0 : GOTO 30
   80 IF K = ASC ( \U ) : GOTO 200
   84 IF K = ASC ( \F ) : LET R = 1023 : GOTO 600 : '  pleine intensite
   90 IF K = ASC ( \D ) : GOTO 400
   94 IF K = ASC ( \O ) : LET R = 0 : GOTO 600 : '  eteindre
   96 IF K = ASC ( \? ) : GOSUB HELP : GOTO 600
  100 IF K = ASC ( \Q ) : GOSUB CLS : END
  110 GOTO LOOP
  200 IF R < 1023 : LET R = R + S : GOTO 600
  210 GOTO LOOP
  400 IF R > 0 : LET R = R - S : GOTO 600
  410 GOTO LOOP
  600 IF R < 0 : LET R = 0
  602 IF R > 1023 : LET R = 1023
  604 GOSUB CLS : ? R ;
  610 GOTO LOOP
 1000 UPPER ' upper case letter
 1010 IF K < ASC ( \a ) : RETURN
 1020 IF K > ASC ( \z ) : RETURN
 1030 LET K = K - 32
 1040 RETURN
 2000 CLS ' clear terminal screen and move cursor home
 2010 ? CHAR ( 27 ) ; "[2J" ; CHAR ( 27 ) ; "[H"
 2020 RETURN
 3000 HELP
 3010 GOSUB CLS
 3012 ? "To control LD2 use:"
 3014 ? , "'D' decrease intensity"
 3016 ? , "'U' increase intensity"
 3018 ? , "'F' full intensity"
 3020 ? , "'O' turn off LD2"
 3024 ? , "'Q' quit."
 3026 ? , "'?' help"
 3028 ? "Press any key to leave this help screen."
 3030 DO UNTIL KEY? : ? KEY
 3032 GOSUB CLS
 3034 RETURN
```

L'intensité s'affiche en au à gauche sur le terminal.

L'intensité de la LED est contrôlée à partir du terminal avec les touches

- **u** pour augmenter l'intensitée
- **d** pour la réduire
- **f** pour la pleine intensité
- **o** pour l'éteindre
- **q** opur quitter le programme
- **?** pour afficher l'aide

exemple 3 lecture analogique

Dans cet exemple il s'agit encore de contrôler l'intensité de la LED mais cette fois l'intensité est déterminée par la lecture d'un potentimètre. Il faut brancher un potentiomètre de 10Ko entre **GND**,**V3,3** et l'entrée analogique **AN0** de la carte.

```
  1 AN.READ
  5 'demo lecture analogique
 10 LET K = 0 :PRINT K;: ADCON  1
 20 LET R =ADCREAD ( 0 )
 30 IF R :BSET PORTC,BIT(5)
 40 FOR A = 0 TO R :NEXT A
 50 BRES PORTC,BIT(5)
 60 FOR A =A TO  1023 :NEXT A
 70 IF KEY? :LET K =KEY
 80 IF K =ASC (\q):ADCON  0 :END
 90 PRINT "\b\b\b\b\b\b";R;
100 GOTO  20
```

Le programme peut-être interrompue par len enfonçant la touche **q** sur le terminal.

Sur le ligne 1 de ce programme on voit qu'il y a une étiquette **AN.READ**. Cette étiquette permet de sauvegarder ce programme en mémoire FLASH et de l'exécuter à partir de là. L'étiquette **AN.READ** va devenir le nom du fichier.

```
>save

>dir
$B704 206 bytes,AN.READ

>run an.read
$0
>autorun an.read

>reboot
 auto run program
```

```
432
>
```

On utilise la commande **SAVE** pour sauvegarder le programme en mémoire FLASH ensuite la commande **DIR** nous donne la liste des programmes sauvegardés. Le premier chiffre en hexadécimal est l'adresse d'exécution du programme **$B704**, ensuite viens la taille en décimal **206 octets** et finalement son nom **AN.READ**.

La commande **RUN** suivie d'un nom de fichier permet d'exécuter le fichier portant ce nom.

La commande **AUTORUN** suivit d'un nom de fichier, ici **AN.READ** permet de lancer automatiquement ce programme lorsque la carte est mise sous tension ou réinitialisée avec le bouton **RESET**, la commande **REBOOT** ou encore **CTRL+X**.

La commande **REBOOT** est utilisée pour réinitialiser la carte ce qui a pour effet de démarrer le programme **AN.READ**. Le message **auto run program** est affiché sur le terminal. **432** est la valeur de lecture du potentiomètre. En tournant l'axe du potentiomètre cette valeur change et l'intensité de la LED aussi.

périphérique I2C

I2C est l'acronyme anglophone pour **I**nter **I**ntegrated **C**ommunication. Il s'agit d'un protocole de type **bus** à 2 fils. **bus** veut dire que plus d'un dispositif peut-être branché sur le même bus. Chaque dispositif est identifié par une adresse de 7 bits (ou 10 bits). Dans le dossier **BASIC** il a 2 programme démontrant l'utilisation de ce périphérique. Les commandes qui utilisent ce périphériques sont:

- **I2C.OPEN** pour activé le périphérique.
- **I2C.CLOSE** pour le fermer.
- **I2C.WRITE** pour envoyé des données à un dispositif branché sur le bus.
- **I2C.READ** pour recevoir des données d'un dispositif branché sur le bus.

Le programme i2c_eeprom.bas fait la démonstration de l'utilisation d'une mémoire EEPROM à interface I2C.

Le programme i2c_oled.bas fait la démonstration d'un petit affichage OLED à interfaace I2C.