**Institute of Electrical and Microengineering**
Biomedical and neuromorphic microelectronic systems

Microcontrôleurs 2023, MT-BA4, Microcontrôleurs et systèmes numériques 2023, EL-BA4. TP05-2023-v5.5.fm
v5.5 A. Schmid 2022, May 23

# MICROCONTRÔLEURS
# MICROCONTRÔLEURS ET SYSTÈMES NUMÉRIQUES
# TRAVAIL PRATIQUE NO 5

| MT GROUPE A | MT Groupe B | | EL |
|---|---|---|---|

| No du Groupe | Premier Etudiant | Second Etudiant | Evaluation | Visa Correcteur |
|---|---|---|---|---|
| 093 | Nathann Morand | Felipe Ramirez | | |

PREPARATION PREPARATION PREPARATION

## 5. OPÉRATIONS AVANCÉES AVEC L'AFFICHAGE LCD

Ce travail pratique voit l'étude de l'affichage de chaînes de caractèressur le module LCD, et de chaînes de caractères formattées, avec conversions.

### 5.1 AFFICHACHE LCD ET CHAÎNES DE CARACTÈRES

#### 5.1.1 AFFICHAGE D'UNE CHAÎNE DE CARACTÈRES

L'affichage de chaînes de caractères constantes et une fonction standard. La chaîne constante est placée en mémoire programme au moyen de la directive .db signifiant define byte. Cette directive place le code ASCII de la chaîne de caractères à l'adresse sélectionnée.

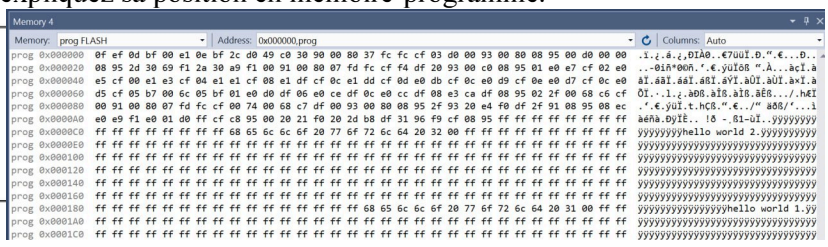
Une étiquette est placée dans le code devant la chaîne qui assigne une constante symbolique à son adresse. La chaîne est terminée par la valeur 0 afin d'en permettre la détection de fin. Par exemple :

```
string1:
.db  "this is a string",0
```

Complétez la sous-routine LCD_putstring donné en Figure 5.1 qui affiche une chaîne constante au LCD.

A quelle adresse de la mémoire-programme se trouve le 'h' de "hello world 1" ? Trouvez-le en affichant le contenu de la mémoire-programme, et expliquez sa position en mémoire-programme.

0x190
il se trouve mis plus loing en mémoire a cause de l'offset de l'instruction .org 200




Téléchargez le code sur le système-cible, et observez ce qui est affiché.

Remplacez la multiplication qui suivent les instruction de définition de z par un facteur de 1x ou de 4x, puis téléchargez, observez ce qui est affiché, et comparez avec le contenu de la mémoire-programme aux adresses correspondantes.

```
; file    puts02.asm   target ATmega128L-4MHz-STK300
; purpose display an ASCII string
.include "macros.asm"
.include "definitions.asm"

reset:
     LDSP      RAMEND
     rcall     LCD_init
     rjmp      main
.include       "lcd.asm"

main:
     ldi       r16,str0
     ldi       zl, low(2*str0)     ; load pointer to string
     ldi       zh, high(2*str0)
     rcall     LCD_putstring       ; display string
     rjmp PC                       ; infinite loop

LCD_putstring:
; in z
     lpm                           ; load program memory into r0
     tst       r0                  ; test for end of string
     breq      done
     mov       a0,r0               ; load argument
     rcall     LCD_putc
     adiw      zl,1                ; increase pointer address
     rjmp      LCD_putstring       ; restart until end of string
done:ret

.org 200
str0:
.db  "hello world 1",0
.org str0/2
.db  "hello world 2",0
.org str0*2
.db  "hello world 3",0
```

*Figure 5.1: puts02.asm.*

### 5.1.2 AFFICHAGE D'UN CHIFFRE AU FORMAT HEXADÉCIMAL

Complétez putx.asm donné en Figure 5.2 une fonction qui affiche une valeur hexadécimale. La valeur lue au format binaire sur les interrupteurs est affichée sur les LEDs, et est affichée en hexadécimal sur l'affichage LCD.

Les huit bits copiés des boutons-poussoirs sont affichés sous forme de deux caractères ASCII représentant chacun un nibble (demi-byte = 4 bits). Pour transformer un nibble en code ASCII, une table (lookup table) contenant les caractères possibles du code hexadécimal est utilisée. Les étapes à suivre sont les suivantes:

- Traitement des bits de poids fort (higher nibble):
  - extraction de la valeur du higher nibble par masquage;
  - attribution de l'offset obtenu au pointeur z;
  - chargement du caractère pointé par z dans la table située en mémoire programme;

```
; file    putx02.asm   target ATmega128L-4MHz-STK300
; purpose display a hex value
.include "macros.asm"   ; include macro definitions
.include "definitions.asm"

reset:
     LDSP RAMEND         ; load stack pointer
     OUTI DDRB,0xff      ; make portB output
     rcallLCD_init       ; initialize LCD
     rjmp main
.include "lcd.asm"

main:rcallLCD_home       ; place cursor to home position
     in   a0,PIND        ; read switches
     out  PORTB,a0       ; write to LED
     com  a0             ; invert a0
     rcall LCD_putx      ; display in hex on LCD
     WAIT_MS 100
     rjmp main

hextb:
.db  "0123456789abcdef"

LCD_putx:
; in a0
     push      a0                  ; save
     swap      a0                  ; display high nibble first
     andi      a0, 0x0f            ; mask higher nibble
     mov       zl, a0              ; load low byte of z
     clr       zh                  ; clear high byte of z
     subi      zl, low(-z*hextb)   ; add offset to table base
     sbci      zh, high(-z*hextb)
     lpm                           ; look up ASCII code
     mov       a0,r0
     recall    LCD_putc            ; put character to LCD
     pop       a0
     andi      a0,0x0f
     mov       zl, a0              ; load offset in low byte
     clr       zh                  ; clear high byte
     subi      zl, low(-z*hextb)   ; add offset to table base
     sbci      zh, high(-z*hextb)
     lpm                           ; look up ASCII code
     mov       a0,r0
     recall    LCD_putc            ; put character to LCD
     ret
```

*Figure 5.2: putx02.asm.*

- affichage au LCD.

• Traitement du lower nibble de façon identique.

### 5.1.3 AFFICHAGE D'UN CHIFFRE AU FORMAT BINAIRE

Complétez putb.asm donné en Figure 5.3 une routine qui affiche au format binaire la valeur lue en binaire sur les interrupteurs. La méthode proposée consiste à effectuer une boucle parcourue huit fois dans laquelle les opérations suivantes sont effectuées:

- décalage à gauche de la valeur lue sur les boutons-poussoirs;
- vérification de la valeur du carry; si ce dernier est actif, alors il faut afficher un '1' à la position courante du LCD sinon un zéro par défaut;
- affichage LCD.

```
; file    putb02.asm   target ATmega128L-4MHz-STK300
; purpose display a binary value
.include "macros.asm"   ; include macro definitions
.include "definitions.asm"

reset:
     LDSP      RAMEND              ; load stack pointer
     OUTI      DDRB,0xff           ; make portB output
     rcall     LCD_init            ; initialize LCD
     rjmp      main
.include "lcd.asm"

main:rcallLCD_clear               ; place cursor to home position
     in        a0,PIND             ; read switches
     out       PORTB,a0            ; write to LED
     com       a0                  ; invert a0
     rcall     LCD_putb            ; display in binary on LCD
     WAIT_MS   100
     rjmp      main

LCD_putb:
; in a0
     mov       a1, a0              ; move argument to different register
     ldi       a2 ,8               ; load counter
loop:
     ldi       a0, '0'             ; load '0'
     lsl       a1                  ; shift bit into carry
     brcc      PC+2
     ldi       a0,'1'              ; load '1'
     recall    LCD_putc            ; put character to LCD
     dec       a2                  ; decrement counter
     brne      loop
     ret
```

*Figure 5.3: putb02.asm.*

### 5.1.4 AFFICHAGE DE CHAÎNES FORMATTÉES

Utilisez la librairie printf.asm pour afficher au LCD la valeur lue sur les boutons-poussoirs dans les différents formats demandés. Aidez-vous du code useprintf.asm dans lequel vous remplacez la ligne commentée par la ligne de code d'impression formattée adéquate.

- Affichage de la valeur lue sur les boutons-poussoirs en décimal signé.
  - Quelle est la ligne de code ?
    .db "b=",DECISIGN, a,0.
- Affichage de la valeur lue en binaire non-signé sur les boutons-poussoirs dans le format fractionaire suivant:
  - Higher nibble: partie entière;
  - Lower nibble: partie fractionaire.
  - Quelle est la plage possible ?
    0-15,9375 par pas de 0.0625;
  - quelle est la ligne de code ?
    .db "a=",FRAC,a,4,$24,0.

```
; file    useprintf.asm   target ATmega128L-4MHz-STK300
; purpose display formatted string
.include "macros.asm"
.include "definitions.asm"

reset:
     LDSP      RAMEND          ; load stack pointer
     OUTI      DDRB,0xff       ; make portB output
     rcall     LCD_init        ; initialize LCD
     rjmp      main
.include "lcd.asm"
.include "printf.asm"

main:
     in        a0,PIND         ; read switches
     out       PORTB,a0        ; write to LED
     com       a0              ; invert a0

     rcall     LCD_home
     rcall     LCD_clear
     PRINTF    LCD
     ; insert your printing command line here
     WAIT_MS   100
     rjmp      main
```

*Figure 5.4: useprintf.asm.*