Institute of Electrical and Microengineering
Biomedical and neuromorphic microelectronic systems

Microcontrôleurs 2023, MT-BA4, Microcontrôleurs et systèmes numériques 2023, EL-BA4. TP12-2023-v5.5.fm
v5.5 A. Schmid 2022, May 23

MICROCONTRÔLEURS
MICROCONTRÔLEURS ET SYSTÈMES NUMÉRIQUES
TRAVAIL PRATIQUE NO 12

MT GROUPE A	MT Groupe B		EL

No du Groupe	Premier Etudiant	Second Etudiant	Evaluation	Visa Correcteur

12. INTERFACES ET PÉRIPHÉRIQUES USUELS 3: LCD AVANCÉ

Ce travail pratique voit l'étude d'accès avancés au LCD.

12.1 DÉPLACEMENT DE L'AFFICHAGE ET DU CURSEUR

PREPARATION

Téléchargez le programme lcd2.asm donné en Figure 12.1.

Comment faut-il modifier lcd2.asm afin que l'affichage apparaisse et défile sur la deuxième ligne (sans tenir compte des curseurs) ?

PREPARATION

Complétez en Figure 12.2 les sous-routines qui permettent de déplacer verticalement le curseur. La méthode à appliquer consiste à lire l'adresse courante du curseur, puis ajouter l'offset en agissant sur le bit qui indique la ligne 1 ou 2.

Associez les nouvelles sous-routines aux boutons PD2, PD3, et PD7. Complétez le code dans un nouveau fichier lcd2-2.asm et testez.

```
; file    lcd2.asm   target ATmega128L-4MHz-STK300
; purpose LCD HD44780U, various display/cursor operations
.include "macros.asm"        ; include macro definitions
.include "definitions.asm"   ; include register/constant definitions

reset: LDSP   RAMEND         ; set up stack pointer (SP)
     OUTI     DDRB,0xff      ; configure portB to output
     rcall    LCD_init       ; initialize LCD
     rcall    LCD_blink_on   ; turn blinking on
     jmp      intro

; === include ===
.include "lcd.asm"

intro: ldi    a0,'A'         ; write the character 'A'
     rcall    lcd_putc
     ldi      a0,'V'         ; write the character 'V'
     rcall    lcd_putc
     ldi      a0,'R'         ; write the character 'R'
     rcall    lcd_putc

main:
     WAIT_MS  100
     CP0  PIND,0,LCD_home              ; Call if Port=0
     CP0  PIND,1,LCD_clear
     CP0  PIND,2,LCD_display_right
     CP0  PIND,3,LCD_display_left
     CP0  PIND,4,LCD_cursor_right
     CP0  PIND,5,LCD_cursor_left
     JP0  PIND,6,intro; Jump if Port=0
     rjmp main
```

Figure 12.1: lcd2.asm.

```
LCD_cursor_up:
     lds       w, LCD_IR      ; read AC (address counter)
     [ ]i      w,[ ]          ; clear bit for line 1
     ori       w,[ ]          ; set bit(table 8.3/4 lecture notes,last line)
     sts       LCD_IR,w
     ret

LCD_cursor_down:
     lds       w, LCD_IR      ; read AC (address counter)
     ori       w,[ ]          ; set bit for line 2
     ori       w,[ ]          ; set bit
     sts       LCD_IR,w
     ret

LCD_cursor_toggle:
     lds       w, LCD_IR      ; read AC (address counter)
     subi      w,[ ]          ; add offset
     ori       w,[ ]          ; set bit
     sts       LCD_IR,w
     ret
```

Figure 12.2: Routines de déplacement vertical du curseur à insérer dans un nouveau fichier lcd2-2.asm basé sur lcd2.asm.

12.2 LCD EN MODE PSEUDO-GRAPHIQUE

Il est possible d'utiliser le LCD en mode pseudo-graphique, c'est-à-dire d'effectuer l'affichage d'un pixel particulier dans la matrice de 16x2 caractères. De la place mémoire dans le générateur de caractères a été réservée à cet effet à hauteur de 64bytes (CGRAM), soit huit caractères.

Le programme animation1.asm donné en Figure 12.3 est basé sur animation0.asm donné dans les notes de cours. Il permet l'affichage d'un effet de scrolling sur un caractère en forme de flèche vers le haut, ou vers le bas si un bouton poussoir est activé.

Etudiez la première partie de ce programme (sous-routine LCD_drCGRAMupw), et complétez ci-dessous:

- L'adresse du caractère reprogrammé dans la CGRAM est [ ].
- Adaptez sur la Figure 12.4 le schéma de principe donné dans les notes de cours, présentant les positions des pointeurs, et offset au programme dans le cas ou la flèche défile du bas vers le haut.
- Que contiennent r18, r22, et r24 ? [ ]
  [ ]
  [ ]
  [ ].
- Sur quel bouton-poussoir faut-il presser afin de changer le sens de défilement de la flèche ?
  [ ].

```
; file animation1.asm       target ATmega128L-4MHz-STK300
; purpose display a button-controlled character scrolling animation
.include "macros.asm"
.include "definitions.asm"

reset:
      LDSP  RAMEND
      rcall LCD_init
      jmp   main

str0:
.db   0x03,0
arrow0:
.db
0b00000100,0b00001110,0b00011111,0b00000100,0b00000100,0b00000100,0b00000100,0b00000000

.include "lcd.asm"
.include "printf.asm"

main:
      ldi   r22,8

   prog0:
      in    r25,PIND
      com   r25
      tst   r25
      breq  _dnm

      rcall      LCD_home
      PRINTF     LCD
.db   "  Arrowhead down ",0
      rcall      LCD_drCGRAMdnw             ;load animation arrowhead downwards
      rjmp  _gon

  _dnm:rcallLCD_home
      PRINTFLCD
.db   "  Arrowhead up  ",0,0
      rcall      LCD_drCGRAMupw             ;load animation arrowhead upwards


 _gon:ldi   r16,str0                        ;load text, including animated character
      ldi   zl, low(2*str0)                 ;load pointer to string
      ldi   zh,high(2*str0)
      rcall LCD_putstring                   ;display string
      WAIT_MS200                            ;wait
      dec   r22                             ;decrement offset
      _BRNE prog0                           ;animated sequence steps not completed
      rjmp  main                            ;infinite loop

LCD_putstring:
; in  z
      lpm                                   ;load program memory into r0
      tst   r0                              ;test for end
      breq  done
      mov   a0,r0                           ;load argument
      rcall LCD_putc
      adiw  zl,1
      rjmp  LCD_putstring
done: ret
```

Figure 12.3: animation1.asm.

```
LCD_drCGRAMupw:
      lds   u, LCD_IR                     ;read IR to check busy flag  (bit7)
      JB1   u,7,LCD_drCGRAMupw            ;Jump if Bit=1 (still busy)
      ldi   r16, 0b01011000               ;2MSBs:write into CGRAM(instruction),
                                          ;6LSBs:address in CGRAM and in charact.
      sts   LCD_IR, r16                   ;store w in IR
      ldi   zl,low(2*arrow0)+8
      ldi   zh,high(2*arrow0)
      mov   r23,zl                        ;upper address limit
      dec   r23
      mov   r24,r23                       ;store upper limit of character in memory
      ldi   r18,8                         ;load size of caracter in table arrow0
      sub   zl,r22                        ;subtract current value of moving offset

   loop01:
      lds   u, LCD_IR                     ;read IR to check busy flag  (bit7)
      JB1   u,7,loop01                    ;Jump if Bit=1 (still busy)
      lpm                                 ;load from z into r0
      mov   r16,r0
      adiw  zl,1
      mov   r23,r24                       ;garantee z remains in character memory
      sub   r23,zl                        ;zone, if not then restart at the begining
      brge  _reg1                         ;subtract current value of moving offset
      subi  zl,8

  _reg1:stsLCD_DR, r16                    ;load definition of one charecter line
      dec   r18
      brne  loop01
      rcall LCD_home                      ;leaving CGRAM
      ret

LCD_drCGRAMdnw:
      lds   u, LCD_IR
      JB1   u,7,LCD_drCGRAMdnw
      ldi   r16,
      sts
      ldi   zl,low(2*arrow0)+8
      ldi   zh,high(2*arrow0)
      mov   r23,zl
            r23,9
      mov   r24,r23
      ldi   r18,8
      sub   zl,r22

   loop02:
      lds   u, LCD_IR
      JB1   u,7,loop02
      lpm
      mov   r16,r0
            zl
      mov   r23,r24
      sub   r23,
            _reg2
            zl,8

  _reg2:stsLCD_DR, r16
      dec   r18
      brne
      rcall LCD_home
      ret
```

Figure 12.3: animation1.asm.

Complétez la sous-routine LCD_drCGRAMdnw afin qu'elle réalise l'affichage de la flèche défilant vers le bas, tête vers le bas, sans changer la table contenant la définition de la flèche. Pour cela vous devez reconsidérer les positions de la mémoire pointées, ainsi que les valeurs de l'offset. Evaluez soigneusement dans quel sens doit évoluer le pointeur z afin que la flèche défile vers le bas, ainsi que dans quel sens il faut parcourrir la table arrow0 afin que la tête de la flèche soit orientée vers le bas. Remplissez la Figure 12.5 afin de vous aider dans cette tâche.

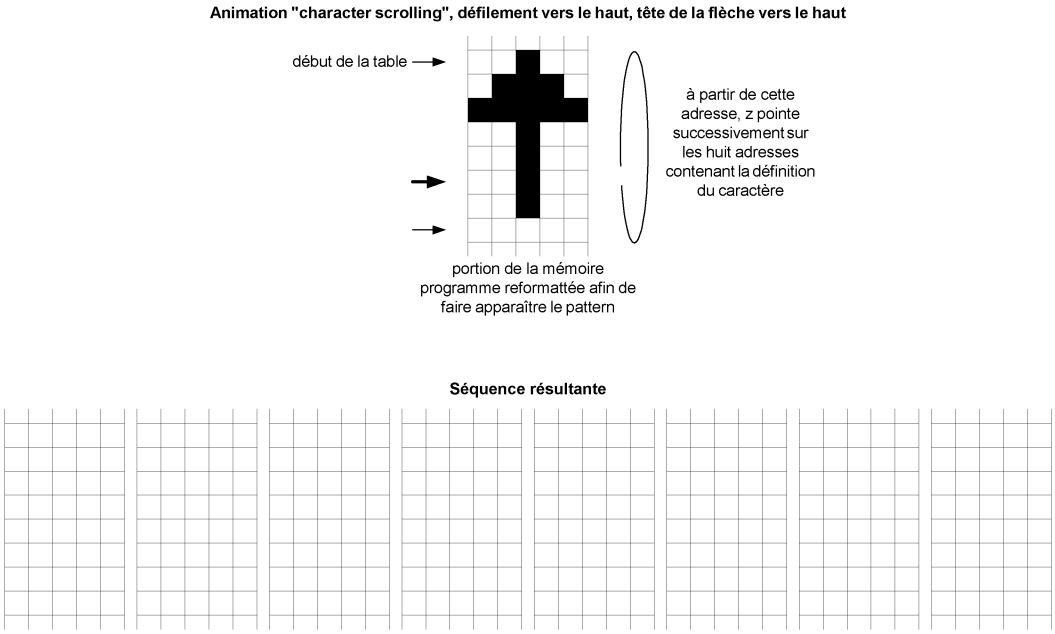

Figure 12.4: Animation générée par LCD_drCGRAMupw.

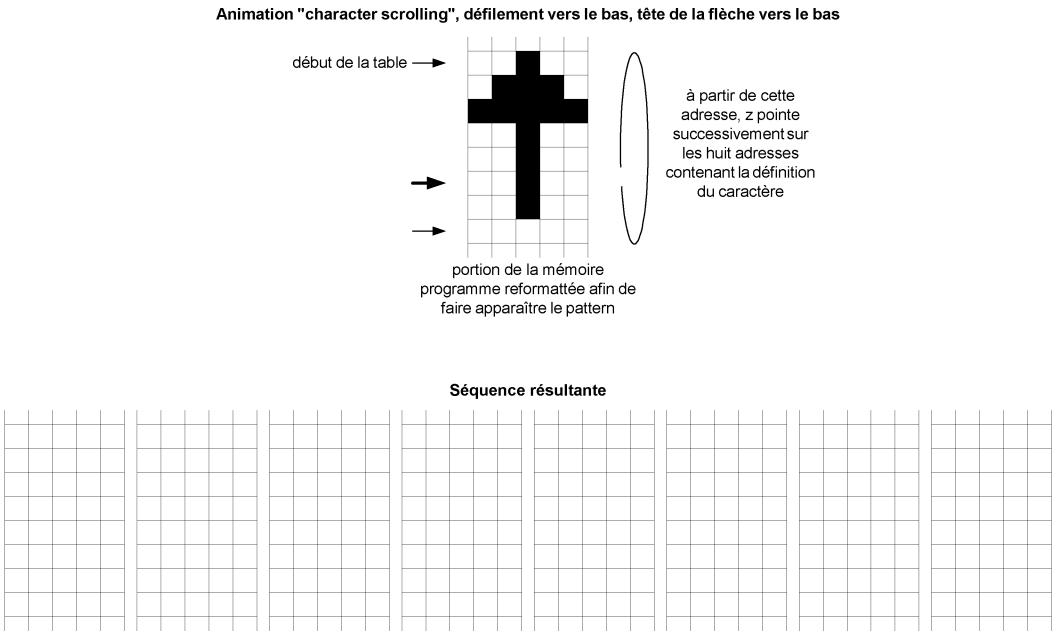

Figure 12.5: Animation générée par LCD_drCGRAMdnw.