

MICROCONTROLEURS

MICROCONTROLEURS ET SYSTEMES NUMERIQUES

TRAVAIL PRATIQUE NO 10

MT GROUPE A	MT Groupe B	EL		
No du Groupe	Premier Etudiant	Second Etudiant	Evaluation	Visa Correcteur

10. INTERFACES ET PÉRIPHÉRIQUES USUELS 1: CONVERTISSEUR A/N ET MOTEUR SERVO, TÉLÉCOMMANDÉ IR

Ce travail pratique voit l'interfaçage de divers périphériques mettant en oeuvre le convertisseur analogique-numérique interne au microcontrôleur, ainsi que divers protocoles de transmission série. Ce TP se concentre sur les thèmes suivants:

- convertisseur A/N et son utilisation avec un moteur servo, et
- télécommande IR.

10.1 CONVERTISSEUR ANALOGIQUE/NUMÉRIQUE ET MOTEUR SERVO

Dans cette manipulation, le moteur servo suit la valeur lue sur le potentiomètre. Dans une première étape, nous cherchons donc à acquérir cette valeur, puis dans une deuxième étape à la transformer en un signal de contrôle acceptable par le servo.

10.1.1 CONVERTISSEUR ANALOGIQUE/NUMÉRIQUE

Téléchargez le programme ad.asm donné en Figure 10.1. Placez le module M3 comprenant le connecteur BNC et le potentiomètre sur le port F.

Analysez le code, et complétez les explications suivantes concernant le fonctionnement du convertisseur:

- Pour activer le convertisseur A/N en anglais ADC signifiant [] , il faut mettre à ‘1’ le bit [] signifiant [] situé dans le registre [] et signifiant [].
- Il faut aussi sélectionner un facteur de division de l’horloge interne. A partir des 4MHz on doit obtenir une valeur entre []Hz et []kHz. Ce facteur de division est contrôlé avec les 3 bits [] dans le registre []. Dans notre cas cette valeur est [] (en binaire 0b[]) ce qui correspond à un facteur de division de []. Donc l’horloge pour le sous-système AD tourne à []kHz.

```

; file ad.asm      target ATmega128L-4MHz-STK300
; purpose ADC demo without interrupts
; module: M3, output port: PORTF
; Aref voltage on STK300, MCU pin 62
.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 the LCD

    OUTI    ADCSR, (1<<ADEN)+6 ; AD Enable, PS=CK/64
    OUTI    ADMUX, POT         ; select channel POT (potentiometer)
    rjmp   main               ; jump ahead to the main program

.include "lcd.asm"             ; include the LCD routines
.include "printf.asm"          ; include formatted printing routines

main:
    sbi     ADCSR, ADSC      ; AD start conversion
    WP1    ADCSR, ADSC      ; wait if ADIF=0
    in     a0, ADCL          ; read low byte first
    in     a1, ADCH          ; read high byte second
    PRINTF  LCD              ; print formatted
.db    CR, CR, "ADC=", FHEX2, a, "=", FDEC2, a, "      ", 0
WAIT_US 100000                ; wait 100 msec
    rjmp   main               ; jump back to main

```

Figure 10.1: ad.asm.

- Le convertisseur A/N utilise la technique de l'approximation successive. La conversion d'une valeur nécessite 13 coups d'horloge, donc [] microsecondes. Le convertisseur est précédé d'un multiplexeur qui permet de sélectionner un canal parmi []. Le numéro du canal sélectionné doit être écrit dans le registre []. Dans notre cas c'est le canal [] qui est sélectionné. Pour initier la conversion il faut mettre à '1' le bit [] signifiant [] dans le registre []. La fin de la conversion est indiqué par le bit [] qui change de '1' à '0.'
- La conversion est faite sur [] bits et le résultat se trouve dans les 2 bytes [] et []. L'ordre de lecture est important; il faut d'abord lire le registre [].

Vérifiez la linéarité de l'ADC en reportant les valeurs de tensions du potentiomètres lues (par exemple à l'oscilloscope (MEASURE, CH1, Type=Mean) par rapport aux codes convertis et lus au LCD. Placez la sonde sur le point de mesure PM2 du module M3.

Faites attention à la tension de référence présentée au convertisseur qui est configurée comme suit (Figure 10.2). Le jumper 1 est en position sur la gauche, sélectionnant le potentiomètre bleu comme source de la référence.

Placez la molette bleue du potentiomètre en position (b) puis en position (c) et reportez les résultats obtenus en Figure 10.3.



Figure 10.2: Configuration de la tension de référence de l'ADC.

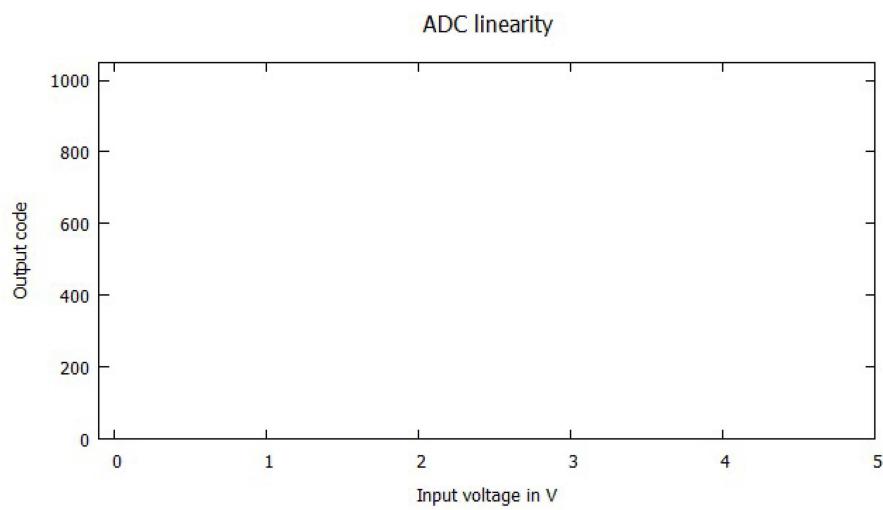


Figure 10.3: Résultats de conversion par l'ADC.

10.1.2 MOTEURS SERVO

Placez le module M3 sur le port F, et le module M4 sur le port B. Connectez le moteur servo Futaba S3003 (noir) sur le connecteur P7 du M4, en respectant strictement le schéma de connexion présenté Figure 10.4.

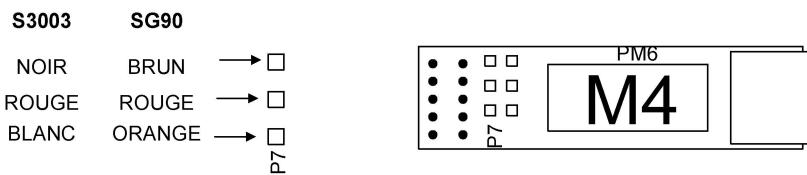


Figure 10.4: Schéma de connexion des deux moteurs servo.

Téléchargez le programme **servo1.asm** donné en Figure 10.5. Les routines étudiées au programme **ad.asm** sont utilisées afin de lire la valeur assignée par le potentiomètre; puis, la valeur est transformée en train d'impulsions afin de faire suivre par le moteur servo la position de consigne assignée au moyen du potentiomètre.

```

; file      servo1.asm    target ATmega128L-4MHz-STK300
; purpose   servo motor control from potentiometer
; module: M3, output port: PORTF
; module: M4, P7 servo Futaba S3003, output port: PORTB
.include "macros.asm"           ; macro definitions
.include "definitions.asm"      ; register/constant definitions

reset:
    LDSP      RAMEND          ; set up stack pointer (SP)
    OUTI      DDRB, 0xff        ; configure portB to output
    rcall    LCD_init          ; initialize the LCD

    OUTI      ADCSR, (1<<ADEN)+6 ; AD Enable, PS=CK/64
    OUTI      ADMUX, POT         ; select channel with potentiometer POT
    rjmp    main                ; jump ahead to the main program

.include "lcd.asm"              ; include the LCD routines
.include "printf.asm"           ; include formatted print routines

main: P0      PORTB, SERVO1    ; pin=0
    WAIT_US  20000
    sbi     ADCSR, ADSC        ; AD start conversion
    WP1     ADCSR, ADSC        ; wait if ADIF=0
    in     a0, ADCL            ; read low byte first
    in     a1, ADCH            ; read high byte second
    ADDI2   a1, a0, 1000        ; add an offset of 1000

    PRINTF   LCD               ; print formatted
    .db "pulse=", FDEC2, a, "usec ", CR, 0

    P1      PORTB, SERVO1    ; pin=1
loop: DEC2   a1, a0
    brne   loop
    rjmp   main

```

Figure 10.5: servo1.asm.

Analysez le code **servo1.asm**, et complétez les explications suivantes concernant le fonctionnement du moteur servo:

- La partie opérative du code peut être séparée en deux parties correspondant au niveaux ‘1’ et ‘0’ du signal de commande. Expliquez les étapes consécutives lorsque le signal est au niveau logique ‘0’:

- Expliquez ce qui se passe lorsque le signal est au niveau logique ‘1’:

.
- Ainsi, le moteur servo est contrôlé par une modulation en largeur d’impulsion; l’acronym anglais est signifiant . La durée d’une impulsion (pulse) varie entre (0%) et (100%). Ces impulsions sont répétées toutes les .

Complétez la macro ADDI2 donnée en Figure 10.6 qui additionne une valeur immédiate à une variable 2-bytes (présentée au moyen de deux registres).

```
.macro ADDI2 ; r1,r0,const
     
     
.endmacro
```

Figure 10.6: Macro ADDI2.

Placez une sonde d’oscilloscope sur la ligne du signal au point de mesure PM6 (voire Figure 10.4), c’est-à-dire en connexion électrique avec la borne qui porte le fil blanc du moteur servo. Reportez en Figure 10.7 et Figure 10.8 le signal observé avec une base de temps de 5ms, et respectivement de 250us, en indiquant avec une ligne continue l’impulsion à 0% et en ligne pointillée l’impulsion à 100%.

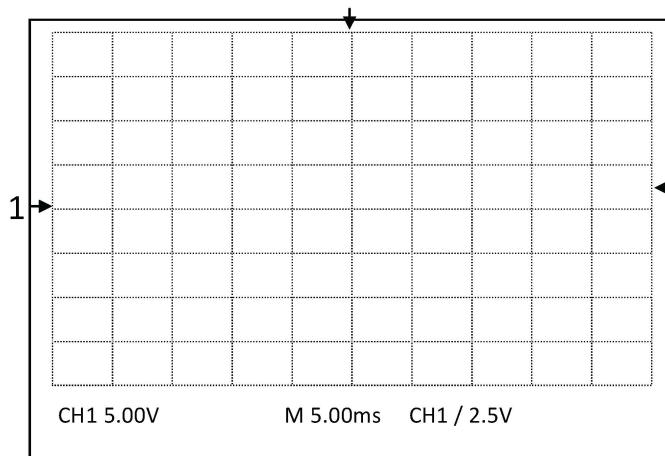


Figure 10.7: Signaux observés: ligne pleine: 0%, ligne pointillée: 100%.

Utilisez les curseurs afin de vérifier les timings.

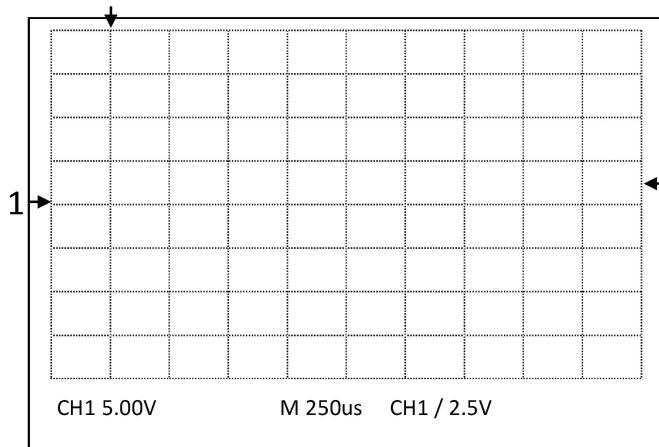


Figure 10.8: Signaux observés: ligne pleine: 0%, ligne pointillée: 100%.

Placez le moteur servo TowerPlus SG90 sur P7 en remplacement du moteur servo Futaba S3003, et en respectant la consigne de connexion (code couleurs des câbles) en Figure 10.4.

Téléchargez le code de contrôle donné dans le fichier `servo36218.asm`. Gardez la sonde de l'oscilloscope sur le point de mesure PM6 et observez la taille des impulsions positives par le menu de l'oscilloscope, MEASURE, CH1, Type: Pos Width.

Quel est la particularité de ce moteur servo ?

Que contrôle le signal électrique transmis au moteur ?

Etudiez le programme présenté en Figure 10.9 et Figure 10.10. Celui-ci opère en deux phases. Dans une première phase, le point de référence zéro de l'axe du moteur doit être défini, car il n'existe pas en interne, contrairement au moteur servo classique. Utilisez PD0 et PD1 afin de positionner l'axe puis PD7 pour entrer cette valeur de référence. Dans la deuxième phase, les boutons PD7-PD0 sont utilisés afin d'effectuer des rotation de valeurs et vitesses pré-définies.

Comparez les valeur de taille d'impulsion affichées à l'oscilloscope et au LCD. Quelle est la raison de la différence que est parfois observée ?

```

; file      servo36218.asm    target ATmega128L-4MHz-STK300
; purpose 360-servo motor control as a classical 180-servo
; with increased angle capability
; module: M4, P7 servo Futaba S3003, output port: PORTB
.include "macros.asm"           ; macro definitions
.include "definitions.asm"      ; register/constant definitions
.equ npt = 1484                 ; effective/observed neutral point of individual servo

reset:
    LDSP      RAMEND          ; set up stack pointer (SP)
    OUTI      DDRB,0xff        ; configure portB to output
    rcall    LCD_init         ; initialize the LCD
    rjmp     main             ; jump to the main program

.include "lcd.asm"              ; include the LCD routines
.include "printf.asm"           ; include formatted print routines

.macro CA3 ;call a subroutine with three arguments in a1:a0 b0
    ldi      a0, low(@1)       ;speed and rotation direction
    ldi      a1, high(@1)      ;speed and rotation direction
    ldi      b0, @2            ;angle
    rcall   @0
.endmacro
; main -----
main:
init:                                ; initializations
    P0      PORTB,SERVO1      ; pin=0
    LDI2    a1,a0,npt

    PRINTF    LCD              ; print formatted
.db  "Set NP > PD0/PD1",LF,0
    PRINTFLCD          ; print formatted
.db  "PD7 to set",LF,0
npset:                                ; neutral point setting
    in      r23,PIND
    cpi    r23, 0b11111110
    breq   _cw
    cpi    r23, 0b11111101
    breq   _ccw
    cpi    r23, 0b01111111
    breq   _npmem
_exec:
    rcall   servoreg_pulse
    rjmp    npset

_cw:
    ADDI2   a1,a0,2           ; increase pulse timing
    rjmp    _exec

_ccw:
    SUBI2   a1,a0,2           ; decrease pulse timing
    rjmp    _exec

_npmem:
    LDI2    a1,a0,npt
    rcall   servoreg_pulse
    rcall   LCD_home
    PRINTF   LCD              ; print formatted
.db  "AR>PD7:PD0 R:PD4",LF,0
    WAIT_MS  50

```

Figure 10.9: servo36218.asm, première partie.

```

ang_rot:                                ; fsm, utilization codes at locations t7:t0
    in      r23,PIND
t0: cpi      r23,0b11111110
    brne   t1
    CA3     _s360, (npt+26), 0x36; cw 90, low-speed
t1: cpi      r23,0b11111101
    brne   t2
    CA3     _s360, (npt+316), 0x0e; cw 180, high-speed
t2: cpi      r23,0b11111011
    brne   t3
    CA3     _s360, (npt+310), 0x36; cw 720, high-speed
t3: cpi      r23,0b01111111
    brne   t4
    CA3     _s360, (npt-26), 0x36; ccw 90, low-speed
t4: cpi      r23,0b10111111
    brne   t5
    CA3     _s360, (npt-310), 0x0e; ccw 180, high-speed
t5: cpi      r23,0b11011111
    brne   t6
    CA3     _s360, (npt-316), 0x36; ccw 720, high-speed
t6: cpi      r23,0b11100111
    brne   t7
    rjmp   init      ; recalibrate neutral point
t7: rjmp   ang_rot

; _s360, in a1:a0, a2 out void, mod a2,w
; purpose execute arbitrary rotation
_s360:
ls3601:
    rcall   servoreg_pulse
    dec     b0
    brne   ls3601
    ret

; servoreg_pulse, in a1,a0, out servo port, mod a3,a2
; purpose generates pulse of length a1,a0
servoreg_pulse:
    PRINTF    LCD          ; print formatted
    .db "pulse=", FDEC2,a,"usec    ",CR,0

    WAIT_US   20000
    MOV2     a3,a2, a1,a0
    P1       PORTB,SERVO1; pin=1
lpssp01: DEC2   a3,a2
    brne   lpssp01
    P0       PORTB,SERVO1; pin=0
    ret

```

Figure 10.10: servo36218.asm, suite.

10.2 TÉLÉCOMMANDE INFRA-ROUGE

Au cours de cette manipulation, nous étudions le standard NEC couramment utilisé pour la télécommande infra-rouge d'appareils électroniques. Le standard NEC utilise un code modulé en distance d'impulsion. Chaque bit est transmis de façon consécutive, LSB en tête, avec une durée de bit différente suivant qu'il s'agisse d'un logic-1, ou logic-0, comme décrit sur la Figure 10.11.

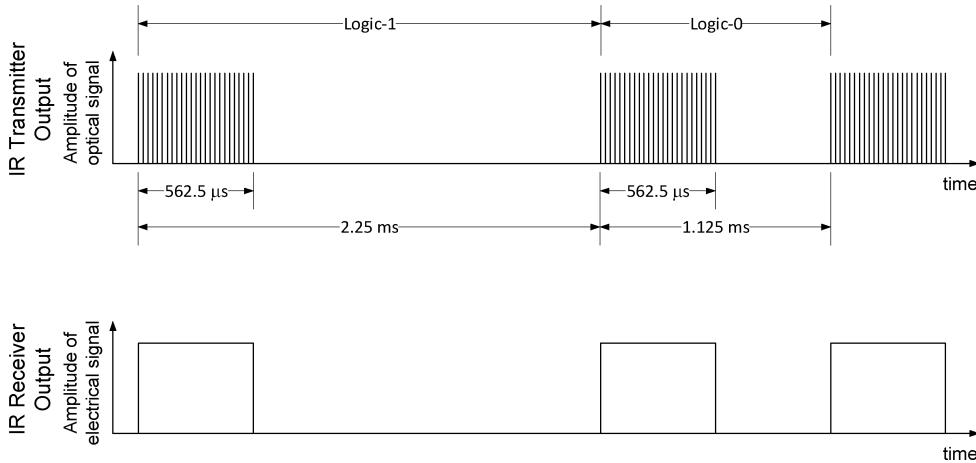


Figure 10.11: Exemple du code NEC.

Le récepteur IR utilisé effectue une inversion du signal par rapport au standard.

Les trames sont constituées d'un lead burst suivi d'un délai, puis de l'adresse émise sous forme vraie puis complémentés, et finalement de la commande émise sous forme vrai puis complémentée.

Placez le module M2 sur le port E. Connectez la sonde attachée au canal 1 de l'oscilloscope avec le point de mesure PM1 du module M2. A l'état de repos, cette ligne est à '1' (logic-1, 5 V). Lorsque l'on appuie sur un bouton de la télécommande, une suite de '0' (logic-0, 0 V) et '1' apparaissent.

Configurez l'oscilloscope comme indiqué en Figure 10.12a et utiliser l'acquisition en single-shot. Appuyez sur la touche 3 de la télécommande. Placez le signal de manière que tous les bits formant la trame soient visibles sur l'écran de l'oscilloscope; cela se fait en déplaçant la position horizontale qui est définie à partie flanc descendant détecté en mode trigger.

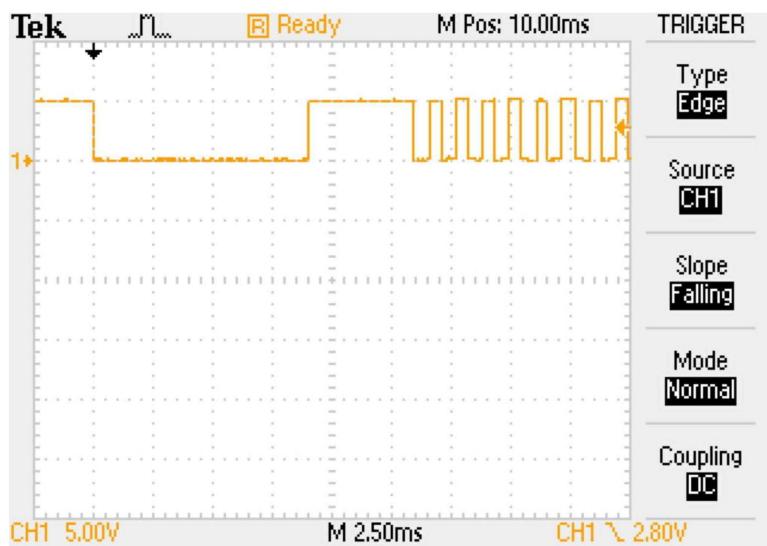
Utilisez les curseurs afin d'effectuer des mesures de timings sur le signal. La durée moyenne totale d'une trame est de ms.

Modifiez l'échelle horizontale afin d'étudier certaines parties de la trame (zoom in); dans le même temps, ajustez sur le milieu de l'écran la partie de la trame intéressante, tel que présenté en Figure 10.12b. Utilisez les curseurs afin d'effectuer des mesures de timings. La durée du "lead pulse" est de ms; la durée du délai suivant le lead pulse ("mark") est de ms.

Indiquez sur la Figure 10.12b le moment d'échantillonnage du premier bit de la trame.



a



b

Figure 10.12: Signaux RC5 observés.

Téléchargez le programme `ir_NECAsm` donné en Figure 10.13, et Figure 10.14, qui échantillonne le signal reçu, puis affiche la valeur décodée sur le LCD.

```

01 ; file ir_NECK.asm    target ATmega128L-4MHz-STK300
02 ; purpose IR sensor decoding NEC format
03 .include "macros.asm"
03 .include "definitions.asm"
05
06 reset:
07     LDSP          RAMEND    ; load stack pointer SP
08     rcall         LCD_init  ; initialize LCD
09     rjmp         main      ; jump to main
10
11 .include "lcd.asm"      ; include the LCD routines
12 .include "printf.asm"   ; include formatted printing routines
13
14 .equ T2 = 15532        ; start timeout, T2 = (14906 + (14906 * Terr2))
15                                ;>with Terr2 = 4.2% observed with the oscilloscope
16 .equ T1 = 1180          ; bit period, T1 = (1125 + (1125 * Terr1)) with
17                                ;>Terr1 = 6.% observed with the oscilloscope
18
19 main: CLR2    b1,b0    ; clear 2-byte register
20     CLR2    a1,a0
21     ldi     b2,16    ; load bit-counter
22     WP1    PINE,IR  ; Wait if Pin=1
23     WAIT_US T2      ; wait for timeout
24     clc      ; clearing carry
25
26 addr: P2C    PINE,IR  ; move Pin to Carry (P2C, 4 cycles)
27     ROL2    b1,b0    ; roll carry into 2-byte reg (ROL2, 2 cycles)
28     sbrc    b0,0     ; (branch not taken, 1 cycle; taken 2 cycles)
29     rjmp    rdz_a    ; (rjmp, 2 cycles)
30     WAIT_US (T1 - 4.5)
31     DJNZ    b2,addr  ; Decrement and Jump if Not Zero (
32                                ;>true, 2 cycles; false, 1 cycle)
33     jmp     next_a   ; (jmp, 3 cycles)
34 rdz_a:          ; read a zero
35     WAIT_US (2*T1 - 5.5)
36     DJNZ    b2,addr  ; Decrement and Jump if Not Zero
37
38 next_a: MOV2    d1,d0, b1, b0 ; store current address
39     MOV2    a1,a0,b1,b0
40     ldi     b2,16    ; load bit-counter
41     clc
42     CLR2    b1,b0
43
44 data: P2C    PINE,IR
45     ROL2    b1,b0
46     sbrc    b0,0
47     rjmp    rdz_d
48     WAIT_US (T1 - 4.5)
49     DJNZ    b2,data
50     jmp     next_b
51 rdz_d:          ; read a zero
52     WAIT_US (2*T1 - 5.5)
53     DJNZ    b2,data

```

Figure 10.13: ir_NECK.asm (première partie).

```

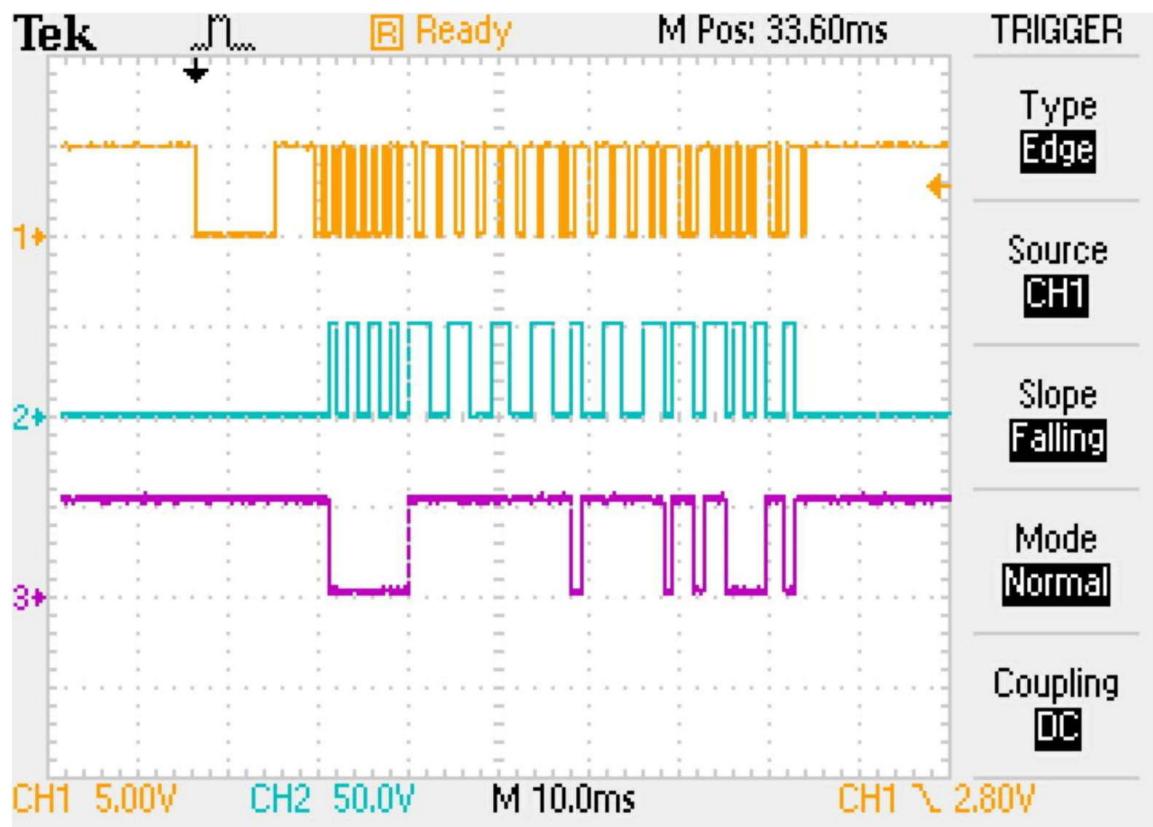
54 next_b:
55     MOV2    d3,d2,b1,b0          ; store current command
56 data_proc01:                      ; detect repeated code
57     _CPI    d3, 0xff
58     brne   data_proc02
59     _CPI    d2, 0xff
60     brne   data_proc02
61     _CPI    d1, 0xff
62     brne   data_proc02
63     _CPI    d0, 0xff
64     brne   data_proc02
65
66 display_repeat:
67     rcall   LCD_clear
68     rcall   LCD_home
69     MOV4    a1,a0,b1,b0,c3,c2,c1,c0; display the last correct code, i.e,
70     PRINTF  LCD                  ;>not the repeated code which reads
71 .db  "A=",FHEX2,a," C=",FHEX2,b,0      ;>address=data=0xff
72     PRINTF  LCD
73 .db  LF, "REPEAT",0
74     rjmp   main
75
76 data_proc02:                      ; detect transmission error
77     com    d1
78     cpse   d0, d1
79     brne   display_error
80     com    d3
81     cpse   d2, d3
82     brne   display_error
83
84 display_correct:
85     com    b0          ; complement b0 (chip delivers the complement)
86     com    b1
87     com    a0
88     com    a1
89     rcall   LCD_clear
90     rcall   LCD_home
91     PRINTF  LCD
92 .db  "A=",FHEX2,a," C=",FHEX2,b,0
93     MOV4    c3,c2,c1,c0,a1,a0,b1,b0 ; storing correct code to display in
94     rjmp   main                  ;>case of repeated code condition
95
96 display_error:
97     com    b0
98     com    b1
99     com    a0
100    com   a1
101    rcall   LCD_clear
102    rcall   LCD_home
103    PRINTF  LCD
104 .db  "EA=",FHEX2,a," EC=",FHEX2,b,0
105 rjmpmain

```

Figure 10.14: ir_NECAsm (deuxième partie).

Modifiez ce programme afin qu'il vous permette de visualiser à l'oscilloscope les traces supplémentaires suivantes, présentée en Figure 10.15

- une trace qui indique l'échantillonnage d'un bit par un changement du niveau de la pin 3 du PORTB,
- une trace qui indique la valeur décodée du bit sur la pin 5 du PORTB.



a

Figure 10.15: Signaux décodés NEC à observer.

Expliquez les changements effectués au programme.

