```
;|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||;
;||||||||||||||||||||||||||||||||| Constantes ||||||||||||||||||||||||||||||||||;
;|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||;

;---------------------------- Configuración del PWM ---------------------------;
;
; Compare Output Mode: COMn1:COMn0 = 1:0 => Clear OCn on compare match
; Waveform Generation Mode: WGMn1:WGMn0 1:1 => Fast PWM
; Clock select: CSn2:CSn1:CSn0 = 0:0:1 => CPU clock, no prescaling
;
; * Timer/Counter-n Control Register: (n = 0, 2. Timer/Counter0, Timer/Counter2)
;      |  FOCn  |  WGMn0 |  COMn1 |  COMn0 |  WGMn1 |  CSn2  |  CSn1  |  CSn0  |
;      |   0    |   1    |   1    |   0    |   1    |   0    |   0    |   1    |
;
; Timer0 Overflow Interrupt Enable: TOIE0 = 1 => Enabled
; Timer2 Overflow Interrupt Enable: TOIE2 = 0 => Disabled
;
; * Timer/Counter Interrupt Mask Register:
;      |  OCIE2 |  TOIE2 | TICIE1 | OCIE1A | OCIE1B |  TOIE1 |  OCIE0 |  TOIE0 |
;      |   x    |   0    |   x    |   x    |   x    |   x    |   x    |   1    |
;
.equ PWM_FAST_PWM_CONFIG_T1 = (1<<COM01) | (1<<WGM01) | (1<<WGM00) | (1<<CS00)
.equ PWM_FAST_PWM_CONFIG_T2 = (1<<COM21) | (1<<WGM21) | (1<<WGM20) | (1<<CS20)
.equ PWM_OFF_PWM_CONFIG     = 0
.equ PWM_OV_INTERRUPT_MASK  = (1<<TOIE0)
.equ PWM_SINE_TABLE_LEN     = 62
.equ PWM_SINE_MEDIAN        = 127

;--------------------- Configuración del Timer1 (16 bits) ---------------------;
;
; Waveform Generation Mode: WGM13:WGM12:WGM11:WGM10 0:0:0:0 => Normal mode
; Clock select: CS12:CS11:CS10 = 0:1:1 => CPU clock, divided by 64 -> 250 kHz
;
; * Timer/Counter1 Control Register A:
;      | COM1A1 | COM1A0 | COM1B1 | COM1B0 |  FOC1A |  FOC1B |  WGM11 |  WGM10 |
;      |   0    |   0    |   0    |   0    |   0    |   0    |   0    |   0    |
;
; * Timer/Counter1 Control Register B:
;      |  ICNC1 |  ICES1 |  *Res* |  WGM13 |  WGM12 |  CS12  |  CS11  |  CS10  |
;      |   0    |   0    |   0    |   0    |   0    |   0    |   1    |   1    |
;
.equ TIMER1_CLOCK_64_PRESCALER = (1<<CS11) | (1<<CS10)
.equ TIMER1_OFF                = 0
.equ TIMER1_50ms_DELAY_START   = -12500 ; 12500 / 250 kHz = 50 ms

;---------------------------- Configuración del ADC ---------------------------;
;
; Reference Selection: REFS1:REFS0 = 0:1 => AVCC with ext capacitor at AREF pin
; ADC Left Adjust Result: ADLAR = 1 => On (8 bits precision reading ADCH only)
; Analog Channel and Gain Selection Bits: MUX4:MUX3:MUX2:MUX1:MUX0
;   1) ImpedanceMeasure  = 6
;   2) OffsetCalibration = 7
;
; * ADC Multiplexer Selection Register:
;      |  REFS1 |  REFS0 |  ADLAR |  MUX4  |  MUX3  |  MUX2  |  MUX1  |  MUX0  |
;      |   0    |   1    |   1    |   0    |   0    |   x    |   x    |   x    |
;
; ADC enable: ADEN = 1 => Enabled
; ADC Auto Trigger Mode: ADATE = 0 => Disabled
; ADC Interrupt Enable: ADIE = 1 => Enabled
; ADC Prescaler Select Bits: ADPS2:ADPS1:ADPS0 = 1:0:1 => fCk_ADC = fCk/32
;
; * ADC Control and Status Register A:
;      |  ADEN  |  ADSC  |  ADATE |  ADIF  |  ADIE  |  ADPS2 |  ADPS1 |  ADPS0 |
;      |   1    |   0    |   0    |   0    |   1    |   1    |   0    |   1    |
;

.equ ADC_AREF_LEFT_ADJUST_CONFIG = (1<<REFS0) | (1<<ADLAR)
.equ ADC_ENABLE_AUTO_INT_PRESC   = (1<<ADEN)  | (1<<ADIE) | \
                                   (1<<ADPS2) | (1<<ADPS0)
.equ ADC_DISABLE            = 0
.equ ADC_PERIODS_TO_SAMPLE  = 9
.equ ADC_SAMPLES_PER_PERIOD = 37 ; Ya que fADC = 37,037 kHz (sampling freq)
```

```
74  .equ ADC_SAMPLES_TABLE_LEN  = ADC_PERIODS_TO_SAMPLE * ADC_SAMPLES_PER_PERIOD
75
76  ; Enumerativo para la entrada a medir
77  .equ ADC_IMPEDANCE_MEASURE  = 6
78  .equ ADC_OFFSET_CALIBRATION = 7
79
80  ;--------------------- Configuración de rangos de medición --------------------;
81  ; Valores del multiplexor MUX2
82  .equ MUX2_x30nA  = 0
83  .equ MUX2_x80nA  = 1
84  .equ MUX2_x120nA = 2
85  .equ MUX2_x200nA = 3
86
87  ; Enumerativo para los rangos de medición
88  .equ MEAS_RANGE_1 = 0
89  .equ MEAS_RANGE_2 = 1
90  .equ MEAS_RANGE_3 = 2
91  .equ MEAS_RANGE_4 = 3
92  ;---------------------------- Registros especiales ----------------------------;
93  .def tmp   = R16        ; Temporario, siempre se podrá pisar sin salvarlo
94  .def param = R17        ; Parámetro para rutinas
95  .def iter  = R18        ; Iterador, múltiple uso, salvar al pisarlo
96  .def iter2 = R19        ; Iterador, múltiple uso, salvar al pisarlo
97
98  ; XXX XXX XXX XXX XXX XXX -- NOTE -- XXX XXX XXX XXX XXX XXX
99  ; Registros siempre en uso por interrupciones, no usar para otra cosa durante
100 ; las mediciones! --> R20, X (R27:R26), R25:R24, Y (R29:R28)
101 .def tbl_i  = R20 ; Iterador de tabla en RAM, en uso durante mediciones!
102 .def tbl_jl = R24 ; Iterador de 16-bits (parte baja), en uso durante mediciones!
103 .def tbl_jh = R25 ; Iterador de 16-bits (parte alta), en uso durante mediciones!
104
105
106 ;|////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
107 ;|//////////////////////| Direcciones reservadas en RAM |\\\\\\\\\\\\\\\\\\\\\\|;
108 ;|////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
109 .dseg
110 .org SRAM_START
111
112 ; Tabla: seno escalado, listo para actualizar el PWM
113 PWM_SINE_RAM_TABLE:
114     .byte PWM_SINE_TABLE_LEN
115
116 ; Tabla: entrada del ADC muestreada durante ADC_PERIODS_TO_SAMPLE períodos
117 ADC_SAMPLES_RAM_TABLE:
118     .byte ADC_SAMPLES_TABLE_LEN
119
120 ; Tabla: máximos de cada período, luego se ordenará para obtener la mediana
121 ADC_MAXS_RAM_TABLE:
122     .byte ADC_PERIODS_TO_SAMPLE
123
124 ; Tabla: mínimos de cada período, luego se ordenará para obtener la mediana
125 ADC_MINS_RAM_TABLE:
126     .byte ADC_PERIODS_TO_SAMPLE
127
128 ; Conversión de BCD a ASCII, en esta posición queda el resultado en ASCII
129 BCD_TO_ASCII_CONVERT_RAM:
130     .byte 5*16
131
132
133 ;|////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
134 ;|////////////////////////| Vector de interrupciones |\\\\\\\\\\\\\\\\\\\\\\\\|;
135 ;|////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
136 .cseg
137
138 ; Interrupción reset -> MAIN
139 .org 0x0
140     jmp     MAIN
141
142 ; Interrupción de overflow del Timer0
143 .org OVF0addr
144     jmp     PWM_DUTY_CYCLE_UPDATE_ISR
145
146 ; Interrupción de conversión completa del ADC
```

```
.org ADCCaddr
    jmp     ADC_SAMPLE_STORE_TO_RAM_ISR

; Interrupción de botón de ESC (INT2)
.org INT2addr
    jmp     INT2_ESC_BUTTON_ISR

; Final del vector de interrupciones
.org INT_VECTORS_SIZE


;|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
;|/////////////////////////////| Datos en flash |\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
;|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
PWM_SINE_FLASH_TABLE:
    .db 0x00, 0x0D, 0x19, 0x26, 0x32, 0x3D, 0x48, 0x52, 0x5B, 0x64, 0x6B, 0x72
    .db 0x77, 0x7B, 0x7D, 0x7F, 0x7F, 0x7E, 0x7B, 0x78, 0x73, 0x6D, 0x66, 0x5E
    .db 0x55, 0x4B, 0x40, 0x35, 0x29, 0x1C, 0x10, 0x03, 0xF6, 0xEA, 0xDD, 0xD1
    .db 0xC6, 0xBB, 0xB0, 0xA7, 0x9E, 0x97, 0x90, 0x8B, 0x86, 0x83, 0x81, 0x81
    .db 0x82, 0x84, 0x87, 0x8C, 0x91, 0x98, 0xA0, 0xA9, 0xB3, 0xBD, 0xC8, 0xD4
    .db 0xE0, 0xED

; === Multiplexor MUX2 para cada rango de medición ===
MEAS_RANGE_FLASH_SIN_MUX2_VALUES:
    .db MUX2_x200nA, MUX2_x120nA, MUX2_x80nA, MUX2_x30nA

; === Valor de amplitud para cada rango de medición ===
; 128 --> 100,0 % --> 200,0 nA de corriente pico
;  84 -->  65,6 % -->  52,5 nA de corriente pico
;  92 -->  71,9 % -->  21,6 nA de corriente pico
;  30 -->  23,4 % -->   7,0 nA de corriente pico
MEAS_RANGE_FLASH_SINAMPS:
    .db 128, 128, 128, 128

; === Valores de piso para cada rango de medición, en kilo ohm (16 bit!) ===
MEAS_RANGE_FLASH_FLOOR_VALUES:
    .dw 0, 143, 243, 335 ; Unidad: kohm

; === Valores del parámetro p inicial para cada rango de medición (16 bit!) ===
MEAS_RANGE_FLASH_P_FACTOR_DEFAULTS:
    .dw 421, 649, 901, 2607

; === Valores de continua de corrección para cada rango de medición ===
MEAS_RANGE_FLASH_CONTINUE_MUX2_VALUES:
    .db MUX2_x30nA, MUX2_x80nA, MUX2_x120nA, MUX2_x200nA

; === Valor inicial de calibración del PWM de Offset ===
PWM_OFFSET_FLASH_CALIB_VALUE:
    .db 210, 218, 218, 223
```

```
#ifdef AVRA
    .nolist
    .include "m32def.inc"
    .list
#endif


.include "configs.asm"
.include "pwm.asm"
.include "adc.asm"
.include "measure_routines.asm"
.include "MEDIR.asm"
.include "BCD.asm"
.include "LCD.asm"
.include "MENU.asm"
.include "CORREGIR_FIN.asm"


;||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||;
;|||||||||||||||||||||||||||||||||||| Main ||||||||||||||||||||||||||||||||||||;
;||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||;
MAIN:
    ldi     tmp,HIGH(RAMEND)
    out     SPH,tmp
    ldi     tmp,LOW(RAMEND)
    out     SPL,tmp   ; Stack pointer
    sei               ; Habilita las interrupciones (global)

;------------------------ Configuración de los puertos ------------------------;
;
;       PA0 --> RS   |                          PC0 --> D0  |
;       PA1 --> R/W  | LCD                      PC1 --> D1  |
;       PA2 --> E    |                          PC2 --> D2  |
;       PA3 <-- ButtonLeft                      PC3 --> D3  | LCD
;       PA4 <-- ButtonRight                     PC4 --> D4  |
;       PA5 <-- ButtonOk                        PC5 --> D5  |
;       PA6 <== ADC: ImpedanceMeasure           PC6 --> D6  |
;       PA7 <== ADC: OffsetCalibration          PC7 --> D7  |
;
;
;       PB0 --> A    | MUX2                     PD0 --> S0  |
;       PB1 --> B    |                          PD1 --> S1  | MUX1
;       PB2 <== INT2: ButtonEsc                 PD2 --> S2  |
;       PB3 ==> OC0: PWMSineWave                PD3 --> S3  |
;       PB4 --> S0   |                          PD4 *Unused*
;       PB5 --> S1   | MUX0                     PD5 *Unused*
;       PB6 --> S2   |                          PD6 *Unused*
;       PB7 --> S3   |                          PD7 ==> OC2: PWMOffsetAdjust
;
    ; Salidas y entradas
    ldi     tmp,0b00000111
    out     DDRA,tmp
    ldi     tmp,0b11111011
    out     DDRB,tmp
    ldi     tmp,0b11111111
    out     DDRC,tmp
    ldi     tmp,0b10001111
    out     DDRD,tmp

    ; Resistencias pull-up
    ldi     tmp,0b00111000
    out     PORTA,tmp
    ldi     tmp,0b00000100
    out     PORTB,tmp
    ldi     tmp,0b01110000
    out     PORTD,tmp

    ; Interrupción de escape
    ldi     tmp,(1<<INT2) ; Interrupción INT2 (flanco descendente por defecto)
    out     GICR,tmp

;------------------------------------------------------------------------------;
    jmp     MENU
```

```
74
75      rjmp    MAIN ; Si se llega hasta aquí por error, se comienza de nuevo
76
77
78  ;|//////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
79  ;|//////////////////////| Delay de 50 milisegundos |\\\\\\\\\\\\\\\\\\\\\\\\|;
80  ;|//////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
81  ;
82  ; Utiliza el Timer1. Solo usa el registro temporal, este nunca se salva.
83  ;
84  DELAY_50ms:
85      ldi     tmp,HIGH(TIMER1_50ms_DELAY_START)
86      out     TCNT1H,tmp
87      ldi     tmp,LOW(TIMER1_50ms_DELAY_START)
88      out     TCNT1L,tmp                      ; Carga del contador del Timer1
89
90      ldi     tmp,TIMER1_CLOCK_64_PRESCALER
91      out     TCCR1B,tmp                      ; Activación del Timer1
92
93  keep_waiting:
94      in      tmp,TIFR
95      sbrs    tmp,TOV1        ; Saltea si el flag de overflow está encendido
96      rjmp    keep_waiting
97
98      ldi     tmp,TIMER1_OFF
99      out     TCCR1B,tmp                      ; Desactivación del Timer1
100     ldi     tmp,(1<<TOV1)
101     out     TIFR,tmp                        ; Borrado del flag de overflow
102
103     ret
104
105
106 ;|//////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
107 ;|//////////////////////| Delay en segundos (1 a 255) |\\\\\\\\\\\\\\\\\\\\\\\\|;
108 ;|//////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
109 ;
110 ; param (R17) <- cantidad de segundos, 1 < param < 255
111 ; Utiliza la rutina de delay de 50 ms. Salva todos los registros que arruina.
112 ;
113 DELAY_PARAM_SECONDS:
114     push    iter
115     push    iter2 ; Registros salvados en el stack
116
117     mov     iter,param ; Contador de segundos, se carga con el parámetro
118 loop_1s:
119     ldi     iter2,20   ; Contador de 50 milisegundos: 20 * 50 ms = 1000 ms = 1 s
120 loop_50ms:
121     rcall   DELAY_50ms
122     dec     iter2      ; Decremento del contador de 50 milisegundos
123     brne    loop_50ms  ; Si se contaron 20 vueltas de 50 ms se deja de repetir
124     dec     iter       ; Decremento del contador de segundos
125     brne    loop_1s    ; Si se contaron 60 vueltas de 1 s se deja de repetir
126
127     pop     iter2
128     pop     iter  ; Registros recuperados del stack
129     ret
```

```
1   INT2_ESC_BUTTON_ISR:
2           POP R16
3           POP R16
4           SEI     ; Deshace los cambios de la llamada a la interrupción y continúa
    hacia MENU
5
6   MENU:
7           LDI param,MEAS_RANGE_4
8           CALL PWM_SINE_STOP    ; Se inicializa el PWM de senoidal
9           CALL PWM_OFFSET_START ; Se inicializa la referencia del OpAmp
10
11          ; Se limpia la RAM de pantalla LCD
12          LDI R16,80
13          LDI ZH,HIGH(BCD_TO_ASCII_CONVERT_RAM)
14          LDI ZL,LOW(BCD_TO_ASCII_CONVERT_RAM)
15          LDI R17,'*'
16  LCD_RAM_INIT:
17          ST Z+,R17
18          DEC R16
19          BRNE LCD_RAM_INIT
20          CALL LCD_INIT
21
22  MED:
23          LDI R16, 0X01
24          CALL CMNDWRT      ;Pone el cursor al principio de la 1ra línea
25          CALL DELAY_1_6ms
26
27
28          LDI R16,'M'
29          CALL DATAWRT
30          LDI R16,'E'
31          CALL DATAWRT
32          LDI R16,'D'
33          CALL DATAWRT
34          LDI R16,'I'
35          CALL DATAWRT
36          LDI R16,'R'
37          CALL DATAWRT
38          LDI R16,' '
39          CALL DATAWRT
40          LDI R16,' '
41          CALL DATAWRT
42          LDI R16,' '
43          CALL DATAWRT
44          LDI R16,'O'
45          CALL DATAWRT
46          LDI R16,'K'
47          CALL DATAWRT
48
49
50          LDI R16,$C0
51          CALL CMNDWRT
52
53          LDI R16,60
54          CALL DATAWRT
55          LDI R16,62
56          CALL DATAWRT
57
58  BOTONES:
59
60          IN R17, PINA
61          BST R17,3
62          BRTC CALIBRAR_1
63          BST R17,4
64          BRTC CORREGIR_JMP
65          BST R17,5
66          BRTC MEDIR_JMP
67
68  JMP BOTONES
69
70  MEDIR_JMP:
71          JMP MEDIR_MENU
72
```

```
CORREGIR_JMP:
        JMP CORREGIR_1

CALIBRAR_1:
        LDI R16, 0X01
        CALL CMNDWRT      ;Pone el cursor al principio de la 1ra línea
        CALL DELAY_1_6ms


        LDI R16,'C'
        CALL DATAWRT
        LDI R16,'A'
        CALL DATAWRT
        LDI R16,'L'
        CALL DATAWRT
        LDI R16,'I'
        CALL DATAWRT
        LDI R16,'B'
        CALL DATAWRT
        LDI R16,'R'
        CALL DATAWRT
        LDI R16,'A'
        CALL DATAWRT
        LDI R16,'R'
        CALL DATAWRT
        LDI R16,' '
        CALL DATAWRT
        LDI R16,' '
        CALL DATAWRT
        LDI R16,' '
        CALL DATAWRT
        LDI R16,'O'
        CALL DATAWRT
        LDI R16,'K'
        CALL DATAWRT


        LDI R16,$C0
        CALL CMNDWRT

        LDI R16,60
        CALL DATAWRT
        LDI R16,62
        CALL DATAWRT

        CALL DELAY_50ms
BOT:
        IN R17, PINA
        BST R17,3
        BRTC CORREGIR_1
        BST R17,4
        BRTC MED_JMP
        BST R17,5
        BRTC CALIBRAR_JMP
JMP BOT

CALIBRAR_JMP:
        JMP CALIBRAR

MED_JMP:
        JMP MED


MEDIR_MENU:
        CALL MEDIR
        JMP RESULTADOS_INIC
        RET

CALIBRAR:
    RJMP CALIBRAR

CORREGIR_1:
        LDI R16, 0X01
```

```
146           CALL CMNDWRT      ;Pone el cursor al principio de la 1ra línea
147           CALL DELAY_1_6ms
148
149
150           LDI R16,'C'
151           CALL DATAWRT
152           LDI R16,'O'
153           CALL DATAWRT
154           LDI R16,'R'
155           CALL DATAWRT
156           LDI R16,'R'
157           CALL DATAWRT
158           LDI R16,'E'
159           CALL DATAWRT
160           LDI R16,'G'
161           CALL DATAWRT
162           LDI R16,'I'
163           CALL DATAWRT
164           LDI R16,'R'
165           CALL DATAWRT
166           LDI R16,' '
167           CALL DATAWRT
168           LDI R16,' '
169           CALL DATAWRT
170           LDI R16,' '
171           CALL DATAWRT
172           LDI R16,'O'
173           CALL DATAWRT
174           LDI R16,'K'
175           CALL DATAWRT
176
177
178           LDI R16,$C0
179           CALL CMNDWRT
180
181           LDI R16,60
182           CALL DATAWRT
183           LDI R16,62
184           CALL DATAWRT
185
186           CALL DELAY_50ms
187   BOT2:
188           IN R17, PINA
189           BST R17,3
190           BRTC MED_JMP1
191           BST R17,4
192           BRTC CALIBRAR_JMP1
193           BST R17,5
194           BRTC CORREGIR
195   JMP BOT2
196
197   MED_JMP1:
198       JMP MED
199
200   CALIBRAR_JMP1:
201       JMP CALIBRAR_1
202
203   CORREGIR:
204           LDI R22, 0
205           LDI R16, 0X01
206           CALL CMNDWRT
207           CALL DELAY_1_6ms
208
209           LDI R16,'I'
210           CALL DATAWRT
211           LDI R16,'M'
212           CALL DATAWRT
213           LDI R16,'P'
214           CALL DATAWRT
215           LDI R16,'E'
216           CALL DATAWRT
217           LDI R16,'D'
218           CALL DATAWRT
```

```
219           LDI R16,'A'
220           CALL DATAWRT
221           LDI R16,'N'
222           CALL DATAWRT
223           LDI R16,'C'
224           CALL DATAWRT
225           LDI R16,'I'
226           CALL DATAWRT
227           LDI R16,'A'
228           CALL DATAWRT
229
230           LDI R16,$C0
231           CALL CMNDWRT
232
233           LDI R16,0X30
234           CALL DATAWRT
235           LDI R16,0X30
236           CALL DATAWRT
237           LDI R16,'k'
238           CALL DATAWRT
239           LDI R16,'O'
240           CALL DATAWRT
241           LDI R16,'h'
242           CALL DATAWRT
243           LDI R16,'m'
244           CALL DATAWRT
245
246       CALL DELAY_50ms
247
248           LDI R20,0
249           LDI R21,0
250
251   BOTONES_2:
252           IN R17 ,PINA
253           BST R17,3  ;Pin 3 del puerto A (AUMENTAR)
254           BRTC AUMENTAR
255           BST R17,4  ;Pin 4 del puerto A (REDUCIR)
256           BRTC REDUCIR_OK
257           BST R17,5  ;PIN 5 DEL PUERTO A (OK)
258           BRTC TMAX_OK
259   JMP BOTONES_2
260
261   REDUCIR_OK:JMP REDUCIR ;CON EL BRANCH NO ALCANZA PARA SALTAR
262   TMAX_OK:JMP TMAX
263
264   AUMENTAR:
265           INC R20
266           CPI R20,10
267           BRNE AUMENTAR_A
268           INC R21
269           LDI R20,0
270
271
272   AUMENTAR_A:
273
274           CPI  R21,8      ;DEJA HASTA 7,9 MOHM
275           BRSH BOTONES_2
276
277           LDI R16, 0X01
278           CALL CMNDWRT
279           CALL DELAY_1_6ms
280
281           LDI R16,'I'
282           CALL DATAWRT
283           LDI R16,'M'
284           CALL DATAWRT
285           LDI R16,'P'
286           CALL DATAWRT
287           LDI R16,'E'
288           CALL DATAWRT
289           LDI R16,'D'
290           CALL DATAWRT
291           LDI R16,'A'
```

```
292           CALL DATAWRT
293           LDI R16,'N'
294           CALL DATAWRT
295           LDI R16,'C'
296           CALL DATAWRT
297           LDI R16,'I'
298           CALL DATAWRT
299           LDI R16,'A'
300           CALL DATAWRT
301
302           LDI R16,$C0
303           CALL CMNDWRT
304
305           LDI R22,48    ;PARA PASAR A ASCII
306           ADD R21,R22
307           ADD R20,R22
308
309           MOV R16,R21
310           CALL DATAWRT
311           MOV R16,R20
312           CALL DATAWRT
313           LDI R16,'k'
314           CALL DATAWRT
315           LDI R16,'O'
316           CALL DATAWRT
317           LDI R16,'h'
318           CALL DATAWRT
319           LDI R16,'m'
320           CALL DATAWRT
321
322           SUB R21,R22  ;VUELVO A LA VARIABLE
323           SUB R20,R22
324
325           CALL DELAY_50ms
326           CALL DELAY_50ms
327
328           JMP BOTONES_2
329
330   REDUCIR:
331           CPI R20,0
332           BREQ REDUCIR_B
333           DEC R20
334           JMP REDUCIR_A
335   REDUCIR_B:
336           CPI R21,0
337           BREQ SALIDA
338           DEC R21
339           LDI R20,9
340           JMP REDUCIR_A
341   SALIDA:
342           JMP BOTONES_2
343
344
345
346   REDUCIR_A:
347
348           LDI R16, 0X01
349           CALL CMNDWRT
350           CALL DELAY_1_6ms
351
352           LDI R16,'I'
353           CALL DATAWRT
354           LDI R16,'M'
355           CALL DATAWRT
356           LDI R16,'P'
357           CALL DATAWRT
358           LDI R16,'E'
359           CALL DATAWRT
360           LDI R16,'D'
361           CALL DATAWRT
362           LDI R16,'A'
363           CALL DATAWRT
364           LDI R16,'N'
```

```
365           CALL DATAWRT
366           LDI R16,'C'
367           CALL DATAWRT
368           LDI R16,'I'
369           CALL DATAWRT
370           LDI R16,'A'
371           CALL DATAWRT
372
373           LDI R16,$C0
374           CALL CMNDWRT
375
376           LDI R22,48     ;PARA PASAR A ASCII
377           ADD R21,R22
378           ADD R20,R22
379
380           MOV R16,R21    ;CAMBIAR
381           CALL DATAWRT
382           MOV R16,R20    ;CAMBIAR
383           CALL DATAWRT
384           LDI R16,'k'
385           CALL DATAWRT
386           LDI R16,'O'
387           CALL DATAWRT
388           LDI R16,'h'
389           CALL DATAWRT
390           LDI R16,'m'
391           CALL DATAWRT
392
393           SUB R21,R22   ;VUELVO A LA VARIABLE
394           SUB R20,R22
395
396           CALL DELAY_50ms
397           CALL DELAY_50ms
398
399           JMP BOTONES_2
400
401   TMAX:
402           LDI R16, 0X01
403           CALL CMNDWRT
404           CALL DELAY_1_6ms
405
406           LDI R16,'T'
407           CALL DATAWRT
408           LDI R16,'M'
409           CALL DATAWRT
410           LDI R16,'A'
411           CALL DATAWRT
412           LDI R16,'X'
413           CALL DATAWRT
414
415           LDI R16,$C0
416           CALL CMNDWRT
417
418           LDI R16,0X30
419           CALL DATAWRT
420           LDI R16,0X30
421           CALL DATAWRT
422
423           LDI R16,'M'
424           CALL DATAWRT
425           LDI R16,'I'
426           CALL DATAWRT
427           LDI R16,'N'
428           CALL DATAWRT
429
430           LDI R23,0
431           LDI R22,48
432
433           CALL DELAY_50ms
434
435
436       BOTONES_3:
437           IN R17 ,PINA
```

```
438         BST R17,3  ;Pin 3 del puerto A (AUMENTAR)
439         BRTC AUMENTAR_T
440         BST R17,4  ;Pin 4 del puerto A (REDUCIR)
441         BRTC REDUCIR_T
442         BST R17,5  ;PIN 5 DEL PUERTO A (OK)
443         BRTC FIN
444     JMP BOTONES_3
445     FIN:
446         LDI R16, 0X01
447         CALL CMNDWRT
448         CALL DELAY_1_6ms
449
450         LDI R16,'E'
451         CALL DATAWRT
452         LDI R16,'S'
453         CALL DATAWRT
454         LDI R16,'P'
455         CALL DATAWRT
456         LDI R16,'E'
457         CALL DATAWRT
458         LDI R16,'R'
459         CALL DATAWRT
460         LDI R16,'E'
461         CALL DATAWRT
462         LDI R16,'.'
463         CALL DATAWRT
464         LDI R16,'.'
465         CALL DATAWRT
466         LDI R16,'.'
467         CALL DATAWRT
468         JMP CORREGIR_ELECTRODO
469
470     REDUCIR_T:
471
472         CPI R23,0
473         BRNE REDUCIR_TP
474         JMP BOTONES_3
475
476
477     AUMENTAR_T:
478         CPI R23,9
479         BRSH BOTONES_3
480         INC R23
481
482
483     AUMENTAR_TP:
484
485         LDI R16, 0X01
486         CALL CMNDWRT
487         CALL DELAY_1_6ms
488
489         LDI R16,'T'
490         CALL DATAWRT
491         LDI R16,'M'
492         CALL DATAWRT
493         LDI R16,'A'
494         CALL DATAWRT
495         LDI R16,'X'
496         CALL DATAWRT
497
498         LDI R16,$C0
499         CALL CMNDWRT
500
501         ADD R23,R22
502
503         MOV R16,R23
504         CALL DATAWRT
505         LDI R16,0X30
506         CALL DATAWRT
507
508         LDI R16,'M'
509         CALL DATAWRT
510         LDI R16,'I'
```

```
511             CALL DATAWRT
512             LDI R16,'N'
513             CALL DATAWRT
514
515             SUB R23,R22
516
517             CALL DELAY_50ms
518
519         JMP BOTONES_3
520
521
522         REDUCIR_TP:
523
524             DEC R23
525             LDI R16, 0X01
526             CALL CMNDWRT
527             CALL DELAY_1_6ms
528
529             LDI R16,'T'
530             CALL DATAWRT
531             LDI R16,'M'
532             CALL DATAWRT
533             LDI R16,'A'
534             CALL DATAWRT
535             LDI R16,'X'
536             CALL DATAWRT
537
538             LDI R16,$C0
539             CALL CMNDWRT
540
541             ADD R23,R22
542
543             MOV R16,R23
544             CALL DATAWRT
545             LDI R16,'0'
546             CALL DATAWRT
547
548             LDI R16,'M'
549             CALL DATAWRT
550             LDI R16,'I'
551             CALL DATAWRT
552             LDI R16,'N'
553             CALL DATAWRT
554
555             SUB R23,R22
556
557             CALL DELAY_50ms
558
559         JMP BOTONES_3
560
561
562
563
564             FIN2:RJMP FIN2
565
566         ;*****************************************************************************
    *****************************************
567
568             LCD_INIT:
569                 CBI PORTA,2
570                 CALL DELAY_50ms
571                 LDI R16,0X38
572                 CALL CMNDWRT
573                 LDI R16,0X0E
574                 CALL CMNDWRT
575                 LDI R16,0X01
576                 CALL CMNDWRT
577                 CALL DELAY_50ms
578                 LDI R16,0X06
579                 CALL CMNDWRT
580                 RET
581
582             ;*****************************************************************************
```

```
    ;*************************************************************
583
584         DELAY_40ms:
585             LDI R17,25
586         DR2:
587             CALL DELAY_1_6ms
588             DEC R17
589             BRNE DR2
590             RET
591
592         ;*************************************************************************************
    ;*****************************************************
593         DELAY_1_6ms:
594             PUSH R17
595             LDI R17,16
596         DR1:
597         CALL DELAY_100us
598         DEC R17
599         BRNE DR1
600         POP R17
601         RET
602
603         ;*************************************************************************************
    ;*****************************************************
604         DELAY_100us:
605             PUSH R17
606             LDI R17,13
607         DR0:  CALL SDELAY
608         DEC R17
609         BRNE DR0
610         POP R17
611         RET
612
613         ;*************************************************************************************
    ;************************************************
614
615         SDELAY:
616               NOP
617               NOP
618               RET
619
620         ;*************************************************************************************
    ;************************************************
621         CMNDWRT:
622             OUT PORTC,R16
623             CBI PORTA,0
624             CBI PORTA,1
625             SBI PORTA,2
626             CALL DELAY_100us
627             CBI PORTA,2
628             CALL DELAY_50ms
629             RET
630         ;*************************************************************************************
    ;************************************************
631         DATAWRT:
632             OUT PORTC,R16
633             SBI PORTA,0
634             CBI PORTA,1
635             SBI PORTA,2
636             CALL DELAY_100us
637             CBI PORTA,2
638             CALL DELAY_50ms
639             RET
```

```
 1  MEDIR:
 2      ldi     param,MEAS_RANGE_4
 3      call    PWM_OFFSET_START
 4
 5  medir_gral:
 6      call    PWM_SINE_START
 7      call    DELAY_50ms
 8      call    ADC_SAMPLING_TO_RAM_FROM_IMPEDANCE_MEASURE_IN
 9      call    PWM_SINE_STOP
10      call    GET_THE_PEAK_VALUE_IN_R2_FROM_ADC_RAM_TABLE
11      call    GET_DECIMEG_IMPEDANCE_FROM_PARAM_RANGE_R2_PEAK_VALUE_IN_R1_R0
12
13      ldi     ZH,HIGH(MEAS_RANGE_FLASH_FLOOR_VALUES<<1)
14      ldi     ZL,LOW(MEAS_RANGE_FLASH_FLOOR_VALUES<<1)
15
16      mov     R2,param
17      lsl     R2          ; R2 = param * 2
18      clr     tmp
19      add     ZL,R2       ; Z = Z + param * 2
20      adc     ZH,tmp      ; Actualización de la parte alta con el carry (tmp = 0)
21
22      lpm     R2,Z+ ; Lectura del parámetro P desde flash, little-endian (L)
23      lpm     R3,Z  ; Lectura del parámetro P desde flash, little-endian (H)
24
25      cp      R0,R2        ; Comparación, parte baja
26      cpc     R1,R3        ; Comparación, parte alta
27      brsh    medir_listo ; Si R1:R0 (medición actual) >= R3:R2 (piso de rango)
28      dec     param        ; Si no, paso a un rango más bajo
29      brne    medir_gral  ; Repetir medición
30
31  medir_listo:
32      call    BCD
33      ret
```

```
1   CORREGIR_ELECTRODO:
2       ; R21 -> R11 Tiene las decenas de la impedancia en kohms
3       ; R20 -> R10 Tiene la unidad de la impedancia en kohms
4       ; R23 -> R12 Tiene TMAX en cantidades de a 10 min
5
6       clr     iter    ; iterador de la cantidad de minutos de corrección
7       mov     R10,R20 ; copia hacia registros no utilizados por las rutinas
8       mov     R11,R21 ; copia hacia registros no utilizados por las rutinas
9       mov     R12,R23 ; copia hacia registros no utilizados por las rutinas
10
11  continuar_corrigiendo:
12      ; Se mide para saber en qué rango corregir (queda en param)
13      ; o si no es necesario hacerlo (medición en R1:R0)
14      call    MEDIR
15
16      ; Se copia la impedancia medida hacia R3:R2
17      mov     R3,R1
18      mov     R2,R0
19
20      ; Verificación de la impedancia
21      ldi     tmp,10
22      mul     tmp,R11 ; R1:R0 <- Decenas de la impedancia en kohms * 10
23      add     R0,R10  ; R1:R0 <- R1:R0 + Unidad de la impedancia en kohms
24      clr     tmp
25      adc     R1,tmp  ; Suma del carry a la parte alta (ya que tmp = 0)
26
27      ; R1:R0 <- valor de impedancia deseada que se comparará con R3:R2 (medida)
28      cp      R0,R2
29      cpc     R1,R3
30      brsh    final_correccion ; Si R1:R0 (deseada) >= R3:R2 (medida)
31
32      call    PWM_CONTINUE_CORRECTION_START
33      ldi     param,60
34      call    DELAY_PARAM_SECONDS
35      inc     iter
36
37      ; Verificación de TMAX
38      ldi     tmp,10
39      mul     tmp,R12 ; R1:R0 <- TMAX en minutos, como es < 90, sólo interesa R0
40      ; Aquí: R0 = TMAX; iter = TACTUAL (en minutos)
41      cp      iter,R0
42      brsh    final_correccion ; Si iter >= R0, se ha cumplido el tiempo máximo
43
44      rjmp    continuar_corrigiendo
45
46  final_correccion:
47      call    PWM_SINE_STOP
48      call    DELAY_50ms
49      call    MEDIR
50      jmp     RESULTADOS_INIC
```

```
1
2       BCD:
3
4               LDI ZH,HIGH(BCD_TO_ASCII_CONVERT_RAM)
5               LDI ZL,LOW(BCD_TO_ASCII_CONVERT_RAM)
6               .def rBin1H =r1  ;REGISTROS MULTIPLICACION
7               .def rBin1L =r0
8               .def rBin2H =r19
9               .def rBin2L =r20
10              .def rmp =r18
11
12
13      Bin2ToAsc5:
14      rcall Bin2ToBcd5
15      ldi rmp,4
16      mov rBin2L,rmp
17  Bin2ToAsc5a:
18      ld rmp,z
19      tst rmp
20      brne Bin2ToAsc5b
21      ldi rmp,' '
22      st z+,rmp
23      dec rBin2L
24      brne Bin2ToAsc5a
25      ld rmp,z
26  Bin2ToAsc5b:
27      inc rBin2L
28  Bin2ToAsc5c:
29      subi rmp,-'0'
30      st z+,rmp
31      ld rmp,z
32      dec rBin2L ;
33      brne Bin2ToAsc5c
34      sbiw ZL,5 ;DIRECCIÓN FINAL
35      ret
36
37
38
39      Bin2ToBcd5:
40      push rBin1H
41      push rBin1L
42      ldi rmp,HIGH(10000)
43      mov rBin2H,rmp
44      ldi rmp,LOW(10000)
45      mov rBin2L,rmp
46      rcall Bin2ToDigit
47      ldi rmp,HIGH(1000)
48      mov rBin2H,rmp
49      ldi rmp,LOW(1000)
50      mov rBin2L,rmp
51      rcall Bin2ToDigit
52      ldi rmp,HIGH(100)
53      mov rBin2H,rmp
54      ldi rmp,LOW(100)
55      mov rBin2L,rmp
56      rcall Bin2ToDigit
57      ldi rmp,HIGH(10)
58      mov rBin2H,rmp
59      ldi rmp,LOW(10)
60      mov rBin2L,rmp
61      rcall Bin2ToDigit
62      st z,rBin1L
63      sbiw ZL,4
64      pop rBin1L
65      pop rBin1H
66      ret
67
68
69  Bin2ToDigit:
70      clr rmp
71  Bin2ToDigita:
72      cp rBin1H,rBin2H
73      brcs Bin2ToDigitc
```

```
74      brne Bin2ToDigitb
75      cp rBin1L,rBin2L
76      brcs Bin2ToDigitc
77  Bin2ToDigitb:
78      sub rBin1L,rBin2L
79      sbc rBin1H,rBin2H
80      inc rmp
81      rjmp Bin2ToDigita
82  Bin2ToDigitc:
83      st z+,rmp
84      ret
```

```
 1
 2
 3  RESULTADOS_INIC:
 4              LDI R19,1 ;Electrodo 1
 5              LDI ZH,HIGH(BCD_TO_ASCII_CONVERT_RAM)
 6              LDI ZL,LOW(BCD_TO_ASCII_CONVERT_RAM)
 7              LDI R22,48
 8
 9
10  RESULTADOS:
11
12              CALL LCD_INIT ;Funcion para inicializar la LCD
13              MOV R21,R19
14
15              CPI R21,10
16              BRLO RESULTADOS_1
17
18              DIEZ:JMP DIEZ_I
19
20  RESULTADOS_1:
21
22              ADD R19,R22
23              LDI R16,'E'
24              CALL DATAWRT
25              LDI R16,'l'
26              CALL DATAWRT
27              LDI R16,'e'
28              CALL DATAWRT
29              LDI R16,'c'
30              CALL DATAWRT
31              LDI R16,'t'
32              CALL DATAWRT
33              LDI R16,'r'
34              CALL DATAWRT
35              LDI R16,'o'
36              CALL DATAWRT
37              LDI R16,'d'
38              CALL DATAWRT
39              LDI R16,'o'
40              CALL DATAWRT
41              LDI R16,' '
42              CALL DATAWRT
43              MOV R16,R19   ;El numero del electrodo(1,2,..16)
44              CALL DATAWRT
45              LDI R16,$C0   ;Esto hay que revisarlo, es para bajar a la segunda lin
    ea del LCD
46              CALL CMNDWRT
47              LD R16,Z+   ;El resultado de la corrección
48              CALL DATAWRT
49              LD R16,Z+   ;El resultado de la corrección
50              CALL DATAWRT
51              LD R16,Z+   ;El resultado de la corrección
52              CALL DATAWRT
53              LD R16,Z+   ;El resultado de la corrección
54              CALL DATAWRT
55              LD R16,Z   ;El resultado de la corrección
56              CALL DATAWRT
57              LDI R16,'k'
58              CALL DATAWRT
59              LDI R16,'O'
60              CALL DATAWRT
61              LDI R16,'h'
62              CALL DATAWRT
63              LDI R16,'m'
64              CALL DATAWRT
65              SUB R19,R22
66
67              LDI R16,0X0F
68              CALL CMNDWRT
69
70              CALL DELAY_50ms
71
72
```

```
 73  BOTONES_4:
 74          IN R17,PINA
 75          BST R17,3  ;Pin 3 del puerto A (AUMENTAR)
 76          BRTC SUBIR
 77          BST R17,4  ;Pin 4 del puerto A (REDUCIR)
 78          BRTC BAJAR
 79  JMP BOTONES_4
 80
 81  SUBIR:
 82          ADIW ZL,1 ; Avanza el puntero
 83          INC R19
 84          CPI R19,17 ;Si se pasa del 16, vuelve al 1
 85          BREQ PRIMERO
 86          JMP RESULTADOS
 87  PRIMERO:
 88          LDI R19,16
 89          SBIW ZL,1
 90          JMP BOTONES_4
 91  BAJAR:
 92          DEC R19
 93          CPI R19,0  ; Si se pasa del 1, vuelve al 16
 94          BREQ ULTIMO
 95          SBIW ZL,9  ; Retrocede el puntero
 96          JMP RESULTADOS
 97  ULTIMO:
 98          LDI R19,1
 99          LDI ZH,HIGH(BCD_TO_ASCII_CONVERT_RAM)
100          LDI ZL,LOW(BCD_TO_ASCII_CONVERT_RAM)
101          LDI R16, 0X01
102          CALL CMNDWRT      ;Pone el cursor al principio de la 1ra línea
103          CALL DELAY_1_6ms
104          JMP RESULTADOS_1
105
106
107  DIEZ_I:
108              SUBI R21,10
109              ADD R21,R22
110
111              LDI R16,'E'
112              CALL DATAWRT
113              LDI R16,'l'
114              CALL DATAWRT
115              LDI R16,'e'
116              CALL DATAWRT
117              LDI R16,'c'
118              CALL DATAWRT
119              LDI R16,'t'
120              CALL DATAWRT
121              LDI R16,'r'
122              CALL DATAWRT
123              LDI R16,'o'
124              CALL DATAWRT
125              LDI R16,'d'
126              CALL DATAWRT
127              LDI R16,'o'
128              CALL DATAWRT
129              LDI R16,' '
130              CALL DATAWRT
131              LDI R16,0x31
132              CALL DATAWRT
133              MOV R16,R21   ;El numero del electrodo(1,2,..16)
134              CALL DATAWRT
135              LDI R16,$C0   ;Esto hay que revisarlo, es para bajar a la segunda lin
     ea del LCD
136              CALL CMNDWRT
137              LD R16,Z+   ;El resultado de la corrección
138              CALL DATAWRT
139              LD R16,Z+   ;El resultado de la corrección
140              CALL DATAWRT
141              LD R16,Z+   ;El resultado de la corrección
142              CALL DATAWRT
143              LD R16,Z+   ;El resultado de la corrección
144              CALL DATAWRT
```

```
145             LD R16,Z   ;El resultado de la corrección
146             CALL DATAWRT
147             LDI R16,'k'
148             CALL DATAWRT
149             LDI R16,'O'
150             CALL DATAWRT
151             LDI R16,'h'
152             CALL DATAWRT
153             LDI R16,'m'
154             CALL DATAWRT
155
156             SUB R21,R22  ;VUELVO A LA VARIABLE
157
158
159             LDI R16,0X0F
160             CALL CMNDWRT
161
162             CALL DELAY_50ms
163
164
165             JMP BOTONES_4
166
```

```
;||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
;||||||||| Interrupción de actualización del ciclo de trabajo del PWM |||||||||;
;||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
PWM_DUTY_CYCLE_UPDATE_ISR:
    push    tmp ; En una interrupción hay que salvar el registro temporal
    in      tmp,SREG
    push    tmp ; También hay que salvar el status register

    ld      tmp,X+    ; Próximo valor del ciclo de trabajo desde RAM
    out     OCR0,tmp  ; Valor actualizado de ciclo de trabajo
    dec     tbl_i     ; Decremento del iterador de la tabla en RAM
    brne    skip_go_beginning ; Saltea si tbl_i no es cero
    ; Go beginning
    rcall   SINE_RAM_TABLE_GO_BEGINNING
skip_go_beginning:
    pop     tmp
    out     SREG,tmp ; Se recupera el status register
    pop     tmp ; Se recupera el registro temporal
    reti


;|////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
;|///| Apuntado de la tabla del seno en RAM e inicialización del contador |\\\|;
;|////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
;
; NOTA: tbl_i (R18) y X (R27:R26) siempre en uso.
;
SINE_RAM_TABLE_GO_BEGINNING:
    ldi     XH,HIGH(PWM_SINE_RAM_TABLE)
    ldi     XL,LOW(PWM_SINE_RAM_TABLE) ; Tabla de onda escalada en RAM
    ldi     tbl_i,PWM_SINE_TABLE_LEN   ; Iterador de la tabla en RAM
    ret


;|////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
;|//////////////////////| Carga de la tabla del seno |\\\\\\\\\\\\\\\\\\\\\\\\|;
;|////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
;
; param (R17) <- escalado de amplitud x 128, ejemplo: 25% = 32/128 => param = 32
; Al finalizar la tabla estará cargada en PWM_SINE_RAM_TABLE, escalada por
; param/128 y con un valor medio de PWM_SINE_MEDIAN. Salva todos los registros
; que arruina.
;
LOAD_SINE_RAM_TABLE_SCALED:
    push    R0
    push    R1
    push    tbl_i
    push    ZH
    push    ZL
    push    XH
    push    XL ; Registros salvados en el stack

    ldi     ZH,HIGH(PWM_SINE_FLASH_TABLE<<1)
    ldi     ZL,LOW(PWM_SINE_FLASH_TABLE<<1) ; Inicialización de puntero en flash
    rcall   SINE_RAM_TABLE_GO_BEGINNING ; Carga el puntero X y el contador tbl_i

loop_sine_table:
    lpm     tmp,Z+    ; Lectura desde flash, del sample original
    mulsu   tmp,param ; Sample escalado y multiplicado x 128 en R1:R0
    rol     R0 ;> División por 128: se multiplica por 2 el entero de 16 bits
    rol     R1 ;> con shifts y luego se divide por 256 quedándose con R1 (MSB)
    ldi     tmp,PWM_SINE_MEDIAN
    add     R1,tmp    ; En R1 queda el sample más la media
    st      X+,R1     ; Carga en RAM del sample final
    dec     tbl_i     ; Decremento del contador
    brne    loop_sine_table

    pop     XL
    pop     XH
    pop     ZL
    pop     ZH
    pop     tbl_i
    pop     R1
```

```
74      pop     R0 ; Registros recuperados del stack
75      ret
76
77
78  ;-------------------------------------------------------------------------;
79  ;--------------- Macros que utilizarán las siguientes rutinas ------------;
80  ;-------------------------------------------------------------------------;
81
82  ; Carga en el puntero Z la dirección de flash recibida como argumento + param
83  .macro flash_point_Z_plus_param
84      ldi     ZH,HIGH(@0<<1)
85      ldi     ZL,LOW(@0<<1) ; Puntero en flash
86      clr     tmp       ; Se borra el registro temporario
87      add     ZL,param ; Se desplaza en la tabla según el parámetro de entrada
88      adc     ZH,tmp    ; Se suma el acarreo a la parte alta (tmp = 0)
89  .endmacro
90
91  ; Escribe el valor del MUX2 según R0, sin chequear su contenido
92  .macro set_MUX2_with_R0_value
93      cli     ; Se desactivan las interrupciones para hacer el cambio
94      in      tmp,PORTB                       ; Lectura desde el puerto B
95      cbr     tmp,(1<<PORTB0)|(1<<PORTB1) ; Limpieza de los bit del MUX2
96      or      tmp,R0                          ; Se graban los bit del MUX2 con R0
97      out     PORTB,tmp                       ; Escritura hacia el puerto B
98      sei     ; Se vuelven a activar las interrupciones
99  .endmacro
100
101
102 ;|/////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
103 ;|////////| Encender la senoidal, para el rango de medición deseado |\\\\\\\\|;
104 ;|/////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
105 ;
106 ; param (R17) <- rango, 4 opciones: MEAS_RANGE_X con X en {1, 2, 3, 4}
107 ; Encenderá el PWM de la senoidal para un rango de medición determinado,
108 ; especificado en el parámetro de entrada, por medio de un valor "enumerativo"
109 ; MEAS_RANGE_X, no hace chequeos de borde, se asume que se recibe un valor
110 ; correcto. Salva todos los registros que arruina.
111 ;
112 PWM_SINE_START:
113     push    R0
114     push    param
115     push    ZH
116     push    ZL ; Registros salvados en el stack
117
118     ; Selección del multiplexor MUX2
119     flash_point_Z_plus_param MEAS_RANGE_FLASH_SIN_MUX2_VALUES
120     lpm     R0,Z
121     set_MUX2_with_R0_value
122
123     ; Creación de tabla en RAM
124     flash_point_Z_plus_param MEAS_RANGE_FLASH_SINAMPS
125     lpm     param,Z  ; Valor correspondiente de amplitud para la rutina
126     rcall   LOAD_SINE_RAM_TABLE_SCALED
127
128     ; Inicialización del puntero X y el contador tbl_i, no se pueden usar más!
129     rcall   SINE_RAM_TABLE_GO_BEGINNING
130
131     ; Configuración del Timer0 como PWM
132     ldi     tmp,PWM_FAST_PWM_CONFIG_T1
133     out     TCCR0,tmp   ; Habilita el PWM en modo rápido
134     in      tmp,TIMSK
135     sbr     tmp,PWM_OV_INTERRUPT_MASK
136     out     TIMSK,tmp   ; Habilita la interrupción de overflow Timer0
137
138     pop     ZL
139     pop     ZH
140     pop     param
141     pop     R0 ; Registros recuperados del stack
142     ret
143
144
145 ;|/////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
146 ;|////////////| Apagar la onda de salida (dejar PWM al 50 % fijo) |\\\\\\\\\\\|;
```

```
;|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
;
; Apaga la onda del PWM, es decir que deja lo deja en su valor medio de forma
; constante (50 % de duty cycle). Salva todos los registros que arruina.
;
PWM_SINE_STOP:
    push    R0 ; Registros salvados en el stack

    ; Se pone el MUX2 en x30nA para minimizar errores de offset
    ldi     tmp,MUX2_x30nA
    mov     R0,tmp
    set_MUX2_with_R0_value

    ; Configuración del Timer0 como PWM
    ldi     tmp,PWM_FAST_PWM_CONFIG_T1
    out     TCCR0,tmp    ; Habilita el PWM en modo rápido
    in      tmp,TIMSK
    cbr     tmp,PWM_OV_INTERRUPT_MASK
    out     TIMSK,tmp    ; Deshabilita la interrupción de overflow Timer0

    ldi     tmp,PWM_SINE_MEDIAN
    out     OCR0,tmp     ; Pone el valor medio en la salida

    pop     R0 ; Registros recuperados del stack
    ret


;|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
;|//| Encender la continua de corrección, para el rango de medición deseado |\|;
;|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
;
; param (R17) <- rango, 4 opciones: MEAS_RANGE_X con X en {1, 2, 3, 4}
; Encenderá el PWM en el modo de corrección, es decir, señales continuas. Según
; el rango especificado en el parámetro de entrada, por medio de un valor
; "enumerativo" MEAS_RANGE_X, no hace chequeos de borde, se asume que se recibe
; un valor correcto. Salva todos los registros que arruina.
;
PWM_CONTINUE_CORRECTION_START:
    push    R0
    push    ZH
    push    ZL ; Registros salvados en el stack

    ; Selección del multiplexor MUX2
    flash_point_Z_plus_param MEAS_RANGE_FLASH_CONTINUE_MUX2_VALUES
    lpm     R0,Z
    set_MUX2_with_R0_value

    ; Deshabilitación del Timer0 como PWM
    ldi     tmp,PWM_OFF_PWM_CONFIG
    out     TCCR0,tmp    ; Deshabilita el PWM en modo rápido
    in      tmp,TIMSK
    cbr     tmp,PWM_OV_INTERRUPT_MASK
    out     TIMSK,tmp    ; Deshabilita la interrupción de overflow Timer0

    ; Corriente continua máxima, 0 en un sentido, 1 en otro
    ; TODO: ver cuál es el correcto
    sbi     PORTB,PORTB3
    ;cbi     PORTB,PORTB3

    pop     ZL
    pop     ZH
    pop     R0 ; Registros recuperados del stack
    ret


;|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
;|/////////////////| Encender el PWM de Offset de referencia |\\\\\\\\\\\\\\\\|;
;|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
;
; param (R17) <- rango, 4 opciones: MEAS_RANGE_X con X en {1, 2, 3, 4}
; Lee desde flash (TODO: EEPROM) los datos de calibración de Offset.
; Salva todos los registros que arruina. Incluso param (R17) y R2.
;
```

```
220 PWM_OFFSET_START:
221     push    ZH
222     push    ZL ; Registros salvados en el stack
223
224     ; Configuración del Timer2 como PWM
225     ldi     tmp,PWM_FAST_PWM_CONFIG_T2
226     out     TCCR2,tmp  ; Habilita el PWM en modo rápido
227
228     flash_point_Z_plus_param PWM_OFFSET_FLASH_CALIB_VALUE
229     lpm     tmp,Z      ; Carga el valor medio (duty cycle) desde flash
230     out     OCR2,tmp   ; Pone el valor medio en la salida
231
232     pop     ZL
233     pop     ZH ; Registros recuperados del stack
234     ret
```

```
;||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
;|||||||||||||| Interrupción del ADC para cada conversión realizada ||||||||||||
;||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
ADC_SAMPLE_STORE_TO_RAM_ISR:
    push    tmp ; En una interrupción hay que salvar el registro temporal

    in      tmp,ADCH   ; Muestra de 8 bits, ya que el ajuste es a izquierda
    st      Y+,tmp
    sbiw    tbl_jl,1   ; Decremento del contador de 16 bits
    brne    skip_stop_sampling ; Saltea si tbl_j no es cero
    ; Stop sampling
    clt ; El borrado del flag T indica que la conversión ha terminado
    ldi     tmp,ADC_DISABLE
    out     ADCSRA,tmp ; Se desactiva el ADC
    out     ADCSRA,tmp ; Se desactiva el ADC
    rjmp    skip_adc_reenabling

skip_stop_sampling:
    sbi     ADCSRA,ADSC ; Inicio de la conversión
skip_adc_reenabling:
    pop     tmp ; Se recupera el registro temporal

    reti


;|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
;|////////| Registro de 9 períodos en RAM a 37 muestras por período |\\\\\\\\|;
;|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
;
; Al finalizar la tabla estará cargada en ADC_SAMPLES_RAM_TABLE con
; ADC_SAMPLES_TABLE_LEN muestras. Salva todos los registros que arruina.
;
ADC_SAMPLING_TO_RAM_FROM_IMPEDANCE_MEASURE_IN:
    ldi     YH,HIGH(ADC_SAMPLES_RAM_TABLE)
    ldi     YL,LOW(ADC_SAMPLES_RAM_TABLE)     ; Inicialización del puntero
    ldi     tbl_jh,HIGH(ADC_SAMPLES_TABLE_LEN)
    ldi     tbl_jl,LOW(ADC_SAMPLES_TABLE_LEN) ; Inicialización del contador
    set ; El flag T de SREG indicará que el muestreo está en curso

    ldi     tmp,ADC_AREF_LEFT_ADJUST_CONFIG | ADC_IMPEDANCE_MEASURE
    out     ADMUX,tmp

    ldi     tmp,ADC_ENABLE_AUTO_INT_PRESC
    out     ADCSRA,tmp

    sbi     ADCSRA,ADSC ; Inicio de la conversión

    ; Esperar hasta que todo esté muestreado en RAM
wait_for_sampling_completion:
    brts    wait_for_sampling_completion

    ret
```

```
1  ;|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
2  ;|///////| Búsqueda de extremos locales (máximo y mínimo por período) |\\\\\\\|;
3  ;|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
4  ;
5  ; Genera ADC_MAXS_RAM_TABLE y ADC_MINS_RAM_TABLE con los máximos y mínimos
6  ; locales de cada período, luego las tablas son ordenadas por otra rutina para
7  ; encontrar ambas medianas. Salva todos los registros que arruina.
8  ;
9  SEARCH_FOR_LOCAL_EXTREMES_AND_LOAD_MIN_MAX_TABLES:
10     push    R1
11     push    R2
12     push    R3
13     push    iter
14     push    iter2
15     push    XH
16     push    XL
17     push    YH
18     push    YL
19     push    ZH
20     push    ZL ; Registros salvados en el stack
21
22     ldi     XH,HIGH(ADC_SAMPLES_RAM_TABLE)
23     ldi     XL,LOW(ADC_SAMPLES_RAM_TABLE) ; Puntero a la tabla de muestras
24     ldi     YH,HIGH(ADC_MAXS_RAM_TABLE)
25     ldi     YL,LOW(ADC_MAXS_RAM_TABLE)    ; Puntero a la tabla de máximos
26     ldi     ZH,HIGH(ADC_MINS_RAM_TABLE)
27     ldi     ZL,LOW(ADC_MINS_RAM_TABLE)    ; Puntero a la tabla de mínimos
28
29     ldi     iter,ADC_PERIODS_TO_SAMPLE    ; Contador para cada período
30 loop_periods:
31     ldi     iter2,ADC_SAMPLES_PER_PERIOD  ; Contador para cada sample
32     ld      R2,X  ; Registro para alojar el máximo, se carga con el primer valor
33     ld      R3,X  ; Registro para alojar el mínimo, se carga con el primer valor
34 loop_samples_one_period:
35     ; Se buscará mínimo y máximo (enteros no signados) dentro de este loop
36     ld      R1,X+ ; Carga de un sample y post incremento
37     cp      R2,R1 ; Si R2 >= R1 NO hay que actualizar el máximo (R2)
38     brsh    skip_update_max
39     mov     R2,R1 ; Actualización del máximo, si R1 > R2
40 skip_update_max:
41     cp      R1,R3 ; Si R1 >= R3 NO hay que actualizar el mínimo (R3)
42     brsh    skip_update_min
43     mov     R3,R1 ; Actualización del mínimo, si R1 < R3
44 skip_update_min:
45     dec     iter2 ; Decremento del contador de samples
46     brne    loop_samples_one_period
47     ; En este punto, el máximo del período está en R2 y el mínimo en R3
48     st      Y+,R2 ; Agregado del máximo en la tabla y post incremento
49     st      Z+,R3 ; Agregado del mínimo en la tabla y post incremento
50     dec     iter  ; Decremento del contador de períodos
51     brne    loop_periods
52
53     pop     ZL
54     pop     ZH
55     pop     YL
56     pop     YH
57     pop     XL
58     pop     XH
59     pop     iter2
60     pop     iter
61     pop     R3
62     pop     R2
63     pop     R1 ; Registros recuperados del stack
64     ret
65
66
67 ;|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
68 ;|///| Obtención de la mediana de la tabla apuntada por Z, de largo param |\\\|;
69 ;|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
70 ;
71 ; param (R17) <- largo de la tabla apuntada por Z
72 ; Ordena por el método de burbujeo una tabla de enteros no signados en memoria,
73 ; apuntada por Z, de largo param (R17). Luego toma el valor del medio para el
```

```
74  ; caso en que la cantidad de elementos sea impar, o el izquierdo de los dos
75  ; centrales, en el caso en que la cantidad de elementos es par. De esta forma
76  ; devuelve la mediana en R1. Salva todos los registros que arruina, incluso R17
77  ; y Z. NOTA: en el caso en que la cantidad de elementos fuera par, debería
78  ; devolver la media aritmética de ambos valores centrales, pero este caso no se
79  ; va a dar.
80  ;
81  CALCULATE_TO_R1_MEDIAN_IN_TABLE_POINTED_BY_Z_LENGTH_IN_PARAM:
82      push    R2
83      push    R3
84      push    iter
85      push    YH
86      push    YL ; Registros salvados en el stack
87
88      ; Ordenamiento por burbujeo, en una tabla pequeña no es un algoritmo tan
89      ; ineficiente, para la aplicación se justifica
90  table_has_changed_loop:
91      clt ; El flag T de SREG indicará si la tabla ha cambiado, en principio no
92      mov     iter,param ; Se inicializa el contador
93      dec     iter       ; Se mirará uno hacia adelante, se recorrerá uno menos
94      mov     YH,ZH
95      mov     YL,ZL      ; Se inicializa el puntero Y en Z
96  loop_table_elements:
97      ld      R2,Y       ; Carga un elemento de la tabla
98      ldd     R3,Y+1     ; Carga el siguiente elemento de la tabla
99      cp      R3,R2      ; Compara, si R3 >= R2, no hay que intercambiarlos
100     brsh    do_not_interchange
101     set ; La tabla va a cambiar, entonces se indica en el flag T
102     st      Y,R3       ; En la primera posición se pone la segunda
103     std     Y+1,R2     ; En la segunda posición se pone la primera
104 do_not_interchange:
105     adiw    YL,1       ; Incremento de Y
106     dec     iter       ; Decremento del contador
107     brne    loop_table_elements
108     brts    table_has_changed_loop
109
110     ; En este punto la tabla está ordenada, ahora la mediana se obtendrá de
111     ; la mitad de la misma
112     clr     R2
113     mov     tmp,param
114     asr     tmp        ; tmp/2: índice de la mitad de la tabla
115     mov     YH,ZH
116     mov     YL,ZL      ; Se inicializa el puntero Y en Z
117     add     YL,tmp     ; Se le suma a Y el lugar de la mitad de la tabla
118     adc     YH,R2      ; Se suma el acarreo a la parte alta (R2 = 0)
119     ld      R1,Y       ; Finalmente se obtiene la mediana en R1 para devolverla
120
121     pop     YL
122     pop     YH
123     pop     iter
124     pop     R3
125     pop     R2 ; Registros recuperados del stack
126     ret
127
128
129 ;|/////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
130 ;|////| Procesamiento de muestras del ADC en RAM para obtener valor pico |\\\\|;
131 ;|/////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
132 ;
133 ; Al finalizar, R2 contendrá el valor pico en bits, de la medición muestreada en
134 ; RAM por el ADC. Salva todos los registros que arruina.
135 ;
136 GET_THE_PEAK_VALUE_IN_R2_FROM_ADC_RAM_TABLE:
137     push    R1
138     push    param
139     push    ZH
140     push    ZL ; Registros salvados en el stack
141
142     rcall   SEARCH_FOR_LOCAL_EXTREMES_AND_LOAD_MIN_MAX_TABLES
143
144     ldi     param,ADC_PERIODS_TO_SAMPLE ; Cantidad de elementos en tablas
145     ldi     ZH,HIGH(ADC_MAXS_RAM_TABLE)
146     ldi     ZL,LOW(ADC_MAXS_RAM_TABLE)  ; Puntero a la tabla de máximos
```

```
147       rcall   CALCULATE_TO_R1_MEDIAN_IN_TABLE_POINTED_BY_Z_LENGTH_IN_PARAM
148       mov     R2,R1                       ; R2 = MAX
149
150       ldi     ZH,HIGH(ADC_MINS_RAM_TABLE)
151       ldi     ZL,LOW(ADC_MINS_RAM_TABLE)  ; Puntero a la tabla de mínimos
152       rcall   CALCULATE_TO_R1_MEDIAN_IN_TABLE_POINTED_BY_Z_LENGTH_IN_PARAM
153       sub     R2,R1                       ; R1 = MIN, valor pico = (MAX-MIN)/2
154       lsr     R2                          ; R2 contiene el valor pico medido
155
156       pop     ZL
157       pop     ZH
158       pop     param
159       pop     R1 ; Registros recuperados del stack
160       ret
161
162
163   ;|//////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
164   ;|///| Obtener impedancia en décimas de mega ohms, según rango de medición |\\|;
165   ;|//////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
166   ;
167   ; param (R17) <- rango, 4 opciones: MEAS_RANGE_X con X en {2, 8, 20, 60}
168   ; R2 <- valor pico de la medición en bits (valor de lectura L)
169   ; Al finalizar R1:R0 contendrá el valor de impedancia en décimas de mega ohm.
170   ; Lee desde flash (TODO: EEPROM) los datos de calibración de cada rango.
171   ; Salva todos los registros que arruina. Incluso param (R17) y R2.
172   ;
173   GET_DECIMEG_IMPEDANCE_FROM_PARAM_RANGE_R2_PEAK_VALUE_IN_R1_R0:
174       push    R2
175       push    R3
176       push    R4
177       push    R5
178       push    R6
179       push    R7
180       push    param
181       push    ZH
182       push    ZL ; Registros salvados en el stack
183
184       ldi     ZH,HIGH(MEAS_RANGE_FLASH_P_FACTOR_DEFAULTS<<1)
185       ldi     ZL,LOW(MEAS_RANGE_FLASH_P_FACTOR_DEFAULTS<<1)
186
187       lsl     param       ; param = param * 2
188       clr     tmp
189       add     ZL,param    ; Z = Z + param
190       adc     ZH,tmp      ; Actualización de la parte alta con el carry (tmp = 0)
191
192       lpm     R6,Z+ ; Lectura del parámetro P desde flash, little-endian (L)
193       lpm     R7,Z  ; Lectura del parámetro P desde flash, little-endian (H)
194
195       ; Multiplicación: R2:R1:R0 = R7:R6 * R2
196       mul     R7,R2       ; R1:R0 = R2 * R7, parte alta del producto
197       mov     R5,R1       ; Se va a salvar en R5:R4
198       mov     R4,R0       ; R5:R4 = R1:R0
199       mul     R6,R2       ; R1:R0 = R2 * R6, parte baja del producto
200       clr     R2          ; R2 = 0
201       add     R1,R4       ; R1 = R1 + R4, suma de ambas partes del producto
202       adc     R2,R5       ; R2 = R5 + carry
203
204       ; División por 256 (el parámetro P fue calculado para esto)
205       mov     R0,R1       ; Basta con desplazar los dos registros
206       mov     R1,R2       ; Resultado de la medición en décimas de mega ohm en R1:R0
207
208       pop     ZL
209       pop     ZH
210       pop     param
211       pop     R7
212       pop     R6
213       pop     R5
214       pop     R4
215       pop     R3
216       pop     R2 ; Registros recuperados del stack
217       ret
218
219
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;; Solo para testear, se cargan estos datos en RAM, generados con octave ;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;TEST_SAMPLES_FLASH_TABLE:
;    .db 0x81, 0x90, 0xA2, 0xAF, 0xC0, 0xC9, 0xD5, 0xDB, 0xE2, 0xE4, 0xE5, 0xDE
;    .db 0xD9, 0xD2, 0xC4, 0xB7, 0xAB, 0x98, 0x87, 0x77, 0x66, 0x56, 0x48, 0x3B
;    .db 0x30, 0x29, 0x1F, 0x1E, 0x1B, 0x1C, 0x22, 0x2A, 0x36, 0x40, 0x50, 0x5E
;    .db 0x6F, 0x7E, 0x92, 0xA2, 0xAF, 0xBF, 0xCC, 0xD3, 0xDD, 0xE0, 0xE4, 0xE4
;    .db 0xE0, 0xD9, 0xD1, 0xC6, 0xB9, 0xA9, 0x99, 0x8B, 0x77, 0x67, 0x57, 0x48
;    .db 0x3D, 0x2F, 0x25, 0x20, 0x1B, 0x1E, 0x20, 0x23, 0x2C, 0x34, 0x40, 0x4E
;    .db 0x5F, 0x6F, 0x80, 0x8F, 0xA2, 0xB1, 0xC0, 0xCB, 0xD5, 0xDA, 0xE3, 0xE3
;    .db 0xE2, 0xDF, 0xDA, 0xD0, 0xC5, 0xBA, 0xAC, 0x9D, 0x89, 0x79, 0x69, 0x56
;    .db 0x4A, 0x3C, 0x31, 0x29, 0x22, 0x1D, 0x1D, 0x1E, 0x21, 0x2B, 0x34, 0x41
;    .db 0x4C, 0x5E, 0x6F, 0x7C, 0x8F, 0xA0, 0xAD, 0xBF, 0xC9, 0xD5, 0xDA, 0xE1
;    .db 0xE3, 0xE5, 0xE0, 0xDB, 0xD2, 0xC6, 0xB9, 0xAC, 0x9B, 0x8B, 0x7C, 0x67
;    .db 0x5A, 0x4C, 0x3D, 0x30, 0x29, 0x23, 0x1E, 0x1B, 0x1C, 0x24, 0x28, 0x35
;    .db 0x3D, 0x4B, 0x5E, 0x6C, 0x7F, 0x8D, 0xA0, 0xAE, 0xBF, 0xCA, 0xD6, 0xDA
;    .db 0xE3, 0xE4, 0xE2, 0xE0, 0xDB, 0xD1, 0xC7, 0xBB, 0xAE, 0x9B, 0x8C, 0x79
;    .db 0x68, 0x5A, 0x4A, 0x3C, 0x2F, 0x2A, 0x22, 0x1C, 0x1B, 0x1F, 0x24, 0x28
;    .db 0x31, 0x3F, 0x4C, 0x5D, 0x6C, 0x7E, 0x8F, 0x9D, 0xAE, 0xBC, 0xC8, 0xD2
;    .db 0xDD, 0xE1, 0xE4, 0xE2, 0xE2, 0xDD, 0xD3, 0xC9, 0xBD, 0xAE, 0x9B, 0x8C
;    .db 0x7B, 0x6A, 0x59, 0x4A, 0x3F, 0x32, 0x27, 0x20, 0x1E, 0x1C, 0x1E, 0x22
;    .db 0x2B, 0x32, 0x3F, 0x4C, 0x5D, 0x6C, 0x7B, 0x8E, 0x9F, 0xAD, 0xBB, 0xCA
;    .db 0xD2, 0xDB, 0xDF, 0xE3, 0xE4, 0xE0, 0xDA, 0xD2, 0xC8, 0xBB, 0xAD, 0x9E
;    .db 0x8C, 0x7B, 0x6A, 0x5A, 0x4C, 0x3F, 0x31, 0x29, 0x22, 0x1D, 0x1D, 0x1D
;    .db 0x22, 0x2A, 0x33, 0x3D, 0x4A, 0x5C, 0x6A, 0x7B, 0x8C, 0x9C, 0xAC, 0xBC
;    .db 0xC7, 0xD4, 0xDA, 0xE2, 0xE2, 0xE4, 0xE2, 0xDC, 0xD5, 0xCA, 0xBD, 0xAE
;    .db 0x9E, 0x8C, 0x7C, 0x6B, 0x59, 0x4C, 0x3D, 0x31, 0x2B, 0x20, 0x1F, 0x1D
;    .db 0x1E, 0x23, 0x29, 0x31, 0x3F, 0x4C, 0x5C, 0x6A, 0x79, 0x8E, 0x9D, 0xAD
;    .db 0xBB, 0xC8, 0xD3, 0xDB, 0xDF, 0xE3, 0xE2, 0xE1, 0xDA, 0xD3, 0xCB, 0xBC
;    .db 0xB0, 0xA0, 0x8F, 0x7E, 0x6E, 0x5C, 0x4B, 0x3E, 0x34, 0x2B, 0x21, 0x1C
;    .db 0x1E, 0x1D, 0x20, 0x29, 0x30, 0x3F, 0x4B, 0x59, 0x69, 0x00
;
;TEST_SAMPLES_FLASH:
;    ldi     ZH,HIGH(TEST_SAMPLES_FLASH_TABLE<<1)
;    ldi     ZL,LOW(TEST_SAMPLES_FLASH_TABLE<<1)
;    ldi     XH,HIGH(ADC_SAMPLES_RAM_TABLE)
;    ldi     XL,LOW(ADC_SAMPLES_RAM_TABLE)
;    ldi     YH,HIGH(ADC_SAMPLES_TABLE_LEN) ; Ojo: no es un puntero!
;    ldi     YL,LOW(ADC_SAMPLES_TABLE_LEN)  ; Es un contador de 16 bits!
;loop_test_samples_table:
;    lpm     tmp,Z+
;    st      X+,tmp
;    sbiw    YL,1 ; Decremento del contador
;    brne    loop_test_samples_table
;
;    rcall   GET_THE_PEAK_VALUE_IN_R2_FROM_ADC_RAM_TABLE
;here:
;    rjmp    here
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
#|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|#
#|/////////////////////////////| Configuraciones |\\\\\\\\\\\\\\\\\\\\\\\\\\\\|#
#|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|#

# Nombre del fuente del programa
NAME = main

# Número de micro Atmega (88/168/328):
IC = 32

# Hardware programador (USBTiny)
PROGRAMMER = usbtiny

# Ensamblador:
AS = avra
# Biblioteca de inclusiones del ensamblador:
ASLIBPATH = /usr/share/avra
# Flags para ensamblado:
ASFLAGS = -l $(NAME).lss -m $(NAME).map -I $(ASLIBPATH) -D AVRA

# Software de programación (grabado en placa, load):
AVR = avrdude
# Flags para programación (load):
AVRFLAGS = -c $(PROGRAMMER) -p m$(IC)


#|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|#
#|////////////////////////| Objetivos y dependencias |\\\\\\\\\\\\\\\\\\\\\\\\|#
#|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|#

#--------------------------- Reglas de construcción ---------------------------#
SRC_FILES = $(wildcard *.asm)
$(NAME).hex: $(SRC_FILES)
$(NAME).eep.hex: $(SRC_FILES)

%.hex: %.asm
    $(AS) $(ASFLAGS) $< -o $@


#|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|#
#|////////////////////////////| Utilidades extras |\\\\\\\\\\\\\\\\\\\\\\\\\\\|#
#|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|#

#----------------------- Programar (grabar en el micro) -----------------------#
burn-flash: $(NAME).hex
    $(AVR) $(AVRFLAGS) -U flash:w:$(NAME).hex

burn-eeprom: $(NAME).eep.hex
    $(AVR) $(AVRFLAGS) -U eeprom:w:$(NAME).eep.hex

#----------------------------- Opciones del clock -----------------------------#
# Ext. Crystal/Resonator High Freq.
# Start-up time: 16K CK + 64 ms
# CKSEL=1111; SUT=11
fuse-crystal:
    $(AVR) $(AVRFLAGS) -U lfuse:w:0xFF:m -U hfuse:w:0xD9:m

fuse-internal:
    $(AVR) $(AVRFLAGS) -U lfuse:w:0xE1:m -U hfuse:w:0xD9:m

#----------------------------------- Limpiar ----------------------------------#
clean:
    rm -f *.hex *.cof *.obj *.map *.lss
```