```
;||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||;
;||||||||||||||||||||||||||||||||| 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_2  = 0
89  .equ MEAS_RANGE_8  = 1
90  .equ MEAS_RANGE_20 = 2
91  .equ MEAS_RANGE_60 = 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
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_x80nA, MUX2_x30nA, 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, 84, 92, 30

; === Valores de piso para cada rango de medición, en décimas de mega ohm ===
MEAS_RANGE_FLASH_FLOOR_VALUES:
    .db 0, 20, 80, 200 ; 0 Mohm, 2 Mohm, 8 Mohm, 20 Mohm

; === Valores del parámetro p inicial para cada rango de medición (16 bit!) ===
MEAS_RANGE_FLASH_P_FACTOR_DEFAULTS:
    .dw 52, 202, 506, 1550

; === 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 0,0 ; cero para completar la palabra de 16 bits
    ;.db PWM_SINE_MEDIAN,0 ; cero para completar la palabra de 16 bits
```

```
#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

;------------------------------------------------------------------------------;
    call    PWM_SINE_STOP    ; Se inicializa el PWM de senoidal
```

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

```
;||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
;||||||||| 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
```

```
    pop     R0 ; Registros recuperados del stack
    ret


;-------------------------------------------------------------------------;
;--------------- Macros que utilizarán las siguientes rutinas ------------;
;-------------------------------------------------------------------------;

; Carga en el puntero Z la dirección de flash recibida como argumento + param
.macro flash_point_Z_plus_param
    ldi     ZH,HIGH(@0<<1)
    ldi     ZL,LOW(@0<<1) ; Puntero en flash
    clr     tmp       ; Se borra el registro temporario
    add     ZL,param ; Se desplaza en la tabla según el parámetro de entrada
    adc     ZH,tmp    ; Se suma el acarreo a la parte alta (tmp = 0)
.endmacro

; Escribe el valor del MUX2 según R0, sin chequear su contenido
.macro set_MUX2_with_R0_value
    cli     ; Se desactivan las interrupciones para hacer el cambio
    in      tmp,PORTB                       ; Lectura desde el puerto B
    cbr     tmp,(1<<PORTB0)|(1<<PORTB1) ; Limpieza de los bit del MUX2
    or      tmp,R0                          ; Se graban los bit del MUX2 con R0
    out     PORTB,tmp                       ; Escritura hacia el puerto B
    sei     ; Se vuelven a activar las interrupciones
.endmacro


;|/////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
;|/////////| Encender la senoidal, para el rango de medición deseado |\\\\\\\\\|;
;|/////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
;
; param (R17) <- rango, 4 opciones: MEAS_RANGE_X con X en {2, 8, 20, 60}
; Encenderá el PWM de la senoidal para un rango de medición determinado,
; 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_SINE_START:
    push    R0
    push    param
    push    ZH
    push    ZL ; Registros salvados en el stack

    ; Selección del multiplexor MUX2
    flash_point_Z_plus_param MEAS_RANGE_FLASH_SIN_MUX2_VALUES
    lpm     R0,Z
    set_MUX2_with_R0_value

    ; Creación de tabla en RAM
    flash_point_Z_plus_param MEAS_RANGE_FLASH_SINAMPS
    lpm     param,Z  ; Valor correspondiente de amplitud para la rutina
    rcall   LOAD_SINE_RAM_TABLE_SCALED

    ; Inicialización del puntero X y el contador tbl_i, no se pueden usar más!
    rcall   SINE_RAM_TABLE_GO_BEGINNING

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

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


;|/////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
;|////////////| Apagar la onda de salida (dejar PWM al 50 % fijo) |\\\\\\\\\\\\|;
```

```
147 ;|////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
148 ;
149 ; Apaga la onda del PWM, es decir que deja lo deja en su valor medio de forma
150 ; constante (50 % de duty cycle). Salva todos los registros que arruina.
151 ;
152 PWM_SINE_STOP:
153     push    R0 ; Registros salvados en el stack
154
155     ; Se pone el MUX2 en x30nA para minimizar errores de offset
156     ldi     tmp,MUX2_x30nA
157     mov     R0,tmp
158     set_MUX2_with_R0_value
159
160     ; Configuración del Timer0 como PWM
161     ldi     tmp,PWM_FAST_PWM_CONFIG_T1
162     out     TCCR0,tmp   ; Habilita el PWM en modo rápido
163     in      tmp,TIMSK
164     cbr     tmp,PWM_OV_INTERRUPT_MASK
165     out     TIMSK,tmp   ; Deshabilita la interrupción de overflow Timer0
166
167     ldi     tmp,PWM_SINE_MEDIAN
168     out     OCR0,tmp    ; Pone el valor medio en la salida
169
170     pop     R0 ; Registros recuperados del stack
171     ret
172
173
174 ;|////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
175 ;|//| Encender la continua de corrección, para el rango de medición deseado |\\|;
176 ;|////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
177 ;
178 ; param (R17) <- rango, 4 opciones: MEAS_RANGE_X con X en {2, 8, 20, 60}
179 ; Encenderá el PWM en el modo de corrección, es decir, señales continuas. Según
180 ; el rango especificado en el parámetro de entrada, por medio de un valor
181 ; "enumerativo" MEAS_RANGE_X, no hace chequeos de borde, se asume que se recibe
182 ; un valor correcto. Salva todos los registros que arruina.
183 ;
184 PWM_CONTINUE_CORRECTION_START:
185     push    R0
186     push    ZH
187     push    ZL ; Registros salvados en el stack
188
189     ; Selección del multiplexor MUX2
190     flash_point_Z_plus_param MEAS_RANGE_FLASH_CONTINUE_MUX2_VALUES
191     lpm     R0,Z
192     set_MUX2_with_R0_value
193
194     ; Deshabilitación del Timer0 como PWM
195     ldi     tmp,PWM_OFF_PWM_CONFIG
196     out     TCCR0,tmp   ; Deshabilita el PWM en modo rápido
197     in      tmp,TIMSK
198     cbr     tmp,PWM_OV_INTERRUPT_MASK
199     out     TIMSK,tmp   ; Deshabilita la interrupción de overflow Timer0
200
201     ; Corriente continua máxima, 0 en un sentido, 1 en otro
202     ; TODO: ver cuál es el correcto
203     sbi     PORTB,PORTB3
204     ;cbi     PORTB,PORTB3
205
206     pop     ZL
207     pop     ZH
208     pop     R0 ; Registros recuperados del stack
209     ret
210
211
212 ;|////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
213 ;|//////////////////| Encender el PWM de Offset de referencia |\\\\\\\\\\\\\\\\\\|;
214 ;|////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
215 ;
216 ; Lee desde flash (TODO: EEPROM) los datos de calibración de Offset.
217 ; Salva todos los registros que arruina. Incluso param (R17) y R2.
218 ;
219 PWM_OFFSET_START:
```

```
220     push    ZH
221     push    ZL ; Registros salvados en el stack
222
223     ; Configuración del Timer2 como PWM
224     ldi     tmp,PWM_FAST_PWM_CONFIG_T2
225     out     TCCR2,tmp  ; Habilita el PWM en modo rápido
226
227     ldi     ZH,HIGH(PWM_OFFSET_FLASH_CALIB_VALUE<<1)
228     ldi     ZL,LOW(PWM_OFFSET_FLASH_CALIB_VALUE<<1) ; Puntero en flash
229
230     lpm     tmp,Z      ; Carga el valor medio (duty cycle) desde flash
231     out     OCR2,tmp   ; Pone el valor medio en la salida
232
233     pop     ZL
234     pop     ZH ; Registros recuperados del stack
235     ret
```

```
INT2_ESC_BUTTON_ISR:
        POP R16
        POP R16
        SEI     ; Deshace los cambios de la llamada a la interrupción y continúa hacia MENU

MENU:
        CALL LCD_INIT

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


        LDI R16,'M'
        CALL DATAWRT
        LDI R16,'E'
        CALL DATAWRT
        LDI R16,'D'
        CALL DATAWRT
        LDI R16,'I'
        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

BOTONES:

        IN R17, PINA
        BST R17,3
        BRTC CALIBRAR_1
        BST R17,4
        BRTC CORREGIR_JMP
        BST R17,5
        BRTC MEDIR_JMP

JMP BOTONES

MEDIR_JMP:
        JMP MEDIR_MENU

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
```

```
73          LDI R16,'L'
74          CALL DATAWRT
75          LDI R16,'I'
76          CALL DATAWRT
77          LDI R16,'B'
78          CALL DATAWRT
79          LDI R16,'R'
80          CALL DATAWRT
81          LDI R16,'A'
82          CALL DATAWRT
83          LDI R16,'R'
84          CALL DATAWRT
85          LDI R16,' '
86          CALL DATAWRT
87          LDI R16,' '
88          CALL DATAWRT
89          LDI R16,' '
90          CALL DATAWRT
91          LDI R16,'O'
92          CALL DATAWRT
93          LDI R16,'K'
94          CALL DATAWRT
95
96
97          LDI R16,$C0
98          CALL CMNDWRT
99
100         LDI R16,60
101         CALL DATAWRT
102         LDI R16,62
103         CALL DATAWRT
104
105         CALL DELAY_50ms
106 BOT:
107         IN R17, PINA
108         BST R17,3
109         BRTC CORREGIR_1
110         BST R17,4
111         BRTC MED_JMP
112         BST R17,5
113         BRTC CALIBRAR_JMP
114 JMP BOT
115
116 CALIBRAR_JMP:
117         JMP CALIBRAR
118
119 MED_JMP:
120         JMP MED
121
122
123 MEDIR_MENU:
124         CALL MEDIR
125         JMP RESULTADOS_INIC
126         RET
127
128 CALIBRAR:
129     RJMP CALIBRAR
130
131 CORREGIR_1:
132         LDI R16, 0X01
133         CALL CMNDWRT      ;Pone el cursor al principio de la 1ra línea
134         CALL DELAY_1_6ms
135
136
137         LDI R16,'C'
138         CALL DATAWRT
139         LDI R16,'O'
140         CALL DATAWRT
141         LDI R16,'R'
142         CALL DATAWRT
143         LDI R16,'R'
144         CALL DATAWRT
145         LDI R16,'E'
```

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

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

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

```
365           CALL CMNDWRT
366
367           LDI R22,48     ;PARA PASAR A ASCII
368           ADD R21,R22
369           ADD R20,R22
370
371           MOV R16,R21    ;CAMBIAR
372           CALL DATAWRT
373           LDI R16,'.'
374           CALL DATAWRT
375           MOV R16,R20    ;CAMBIAR
376           CALL DATAWRT
377           LDI R16,'M'
378           CALL DATAWRT
379           LDI R16,'O'
380           CALL DATAWRT
381           LDI R16,'h'
382           CALL DATAWRT
383           LDI R16,'m'
384           CALL DATAWRT
385
386           SUB R21,R22   ;VUELVO A LA VARIABLE
387           SUB R20,R22
388
389           CALL DELAY_50ms
390           CALL DELAY_50ms
391
392           JMP BOTONES_2
393
394   TMAX:
395           LDI R16, 0X01
396           CALL CMNDWRT
397           CALL DELAY_1_6ms
398
399           LDI R16,'T'
400           CALL DATAWRT
401           LDI R16,'M'
402           CALL DATAWRT
403           LDI R16,'A'
404           CALL DATAWRT
405           LDI R16,'X'
406           CALL DATAWRT
407
408           LDI R16,$C0
409           CALL CMNDWRT
410
411           LDI R16,0X30
412           CALL DATAWRT
413           LDI R16,0X30
414           CALL DATAWRT
415
416           LDI R16,'M'
417           CALL DATAWRT
418           LDI R16,'I'
419           CALL DATAWRT
420           LDI R16,'N'
421           CALL DATAWRT
422
423           LDI R23,0
424           LDI R22,48
425
426           CALL DELAY_50ms
427
428
429       BOTONES_3:
430           IN R17 ,PINA
431           BST R17,3  ;Pin 3 del puerto A (AUMENTAR)
432           BRTC AUMENTAR_T
433           BST R17,4  ;Pin 4 del puerto A (REDUCIR)
434           BRTC REDUCIR_T
435           BST R17,5  ;PIN 5 DEL PUERTO A (OK)
436           BRTC FIN
437       JMP BOTONES_3
```

```
438       FIN:
439           LDI R16, 0X01
440           CALL CMNDWRT
441           CALL DELAY_1_6ms
442
443           LDI R16,'E'
444           CALL DATAWRT
445           LDI R16,'S'
446           CALL DATAWRT
447           LDI R16,'P'
448           CALL DATAWRT
449           LDI R16,'E'
450           CALL DATAWRT
451           LDI R16,'R'
452           CALL DATAWRT
453           LDI R16,'E'
454           CALL DATAWRT
455           LDI R16,'.'
456           CALL DATAWRT
457           LDI R16,'.'
458           CALL DATAWRT
459           LDI R16,'.'
460           CALL DATAWRT
461           JMP CORREGIR_ELECTRODO
462
463       REDUCIR_T:
464
465           CPI R23,0
466           BRNE REDUCIR_TP
467           JMP BOTONES_3
468
469
470       AUMENTAR_T:
471           CPI R23,9
472           BRSH BOTONES_3
473           INC R23
474
475
476       AUMENTAR_TP:
477
478           LDI R16, 0X01
479           CALL CMNDWRT
480           CALL DELAY_1_6ms
481
482           LDI R16,'T'
483           CALL DATAWRT
484           LDI R16,'M'
485           CALL DATAWRT
486           LDI R16,'A'
487           CALL DATAWRT
488           LDI R16,'X'
489           CALL DATAWRT
490
491           LDI R16,$C0
492           CALL CMNDWRT
493
494           ADD R23,R22
495
496           MOV R16,R23
497           CALL DATAWRT
498           LDI R16,0X30
499           CALL DATAWRT
500
501           LDI R16,'M'
502           CALL DATAWRT
503           LDI R16,'I'
504           CALL DATAWRT
505           LDI R16,'N'
506           CALL DATAWRT
507
508           SUB R23,R22
509
510           CALL DELAY_50ms
```

```
511
512         JMP BOTONES_3
513
514
515         REDUCIR_TP:
516
517             DEC R23
518             LDI R16, 0X01
519             CALL CMNDWRT
520             CALL DELAY_1_6ms
521
522             LDI R16,'T'
523             CALL DATAWRT
524             LDI R16,'M'
525             CALL DATAWRT
526             LDI R16,'A'
527             CALL DATAWRT
528             LDI R16,'X'
529             CALL DATAWRT
530
531             LDI R16,$C0
532             CALL CMNDWRT
533
534             ADD R23,R22
535
536             MOV R16,R23
537             CALL DATAWRT
538             LDI R16,'0'
539             CALL DATAWRT
540
541             LDI R16,'M'
542             CALL DATAWRT
543             LDI R16,'I'
544             CALL DATAWRT
545             LDI R16,'N'
546             CALL DATAWRT
547
548             SUB R23,R22
549
550             CALL DELAY_50ms
551
552         JMP BOTONES_3
553
554
555
556
557             FIN2:RJMP FIN2
558
559         ;*************************************************************************
    ***********************************************
560
561             LCD_INIT:
562                 CBI PORTA,2
563                 CALL DELAY_50ms
564                 LDI R16,0X38
565                 CALL CMNDWRT
566                 LDI R16,0X0E
567                 CALL CMNDWRT
568                 LDI R16,0X01
569                 CALL CMNDWRT
570                 CALL DELAY_50ms
571                 LDI R16,0X06
572                 CALL CMNDWRT
573                 RET
574
575             ;*************************************************************************
    ****************************************************
576
577             DELAY_40ms:
578                 LDI R17,25
579             DR2:
580                 CALL DELAY_1_6ms
581                 DEC R17
```

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

```
 1  MEDIR:
 2          LDI param,MEAS_RANGE_60
 3
 4  MEDIR_GRAL:
 5          CALL PWM_SINE_START
 6          CALL DELAY_50ms
 7          CALL ADC_SAMPLING_TO_RAM_FROM_IMPEDANCE_MEASURE_IN
 8          CALL PWM_SINE_STOP
 9          CALL GET_THE_PEAK_VALUE_IN_R2_FROM_ADC_RAM_TABLE
10          CALL GET_DECIMEG_IMPEDANCE_FROM_PARAM_RANGE_R2_PEAK_VALUE_IN_R1_R0
11
12          flash_point_Z_plus_param MEAS_RANGE_FLASH_SIN_MUX2_VALUES
13          LPM     R2,Z        ; Se carga en R2 el piso del rango actual (param)
14
15          CLR     tmp         ; tmp = 0
16          CP      R0,R2       ; Comparación, parte baja
17          CPC     R1,tmp      ; Comparación, parte alta
18          BRSH    meas_ready ; Si R1:R0 (medición actual) >= R2 (piso de rango), li
    sto
19          DEC     param       ; Si no, paso a un rango más bajo
20          BRNE    MEDIR_GRAL ; Repetir medición
21
22  meas_ready:
23          CALL BCD
24          RET
```

```
1   CORREGIR_ELECTRODO:
2               ;R21 TIENE EL VALOR ALTO
3               ;R20 EL VALOR BAJO
4               ;R23 TIENE TMAX
5
6           CLR iter
7
8   CONTINUAR_CORRIGIENDO:
9           CALL MEDIR ; Esto sirve para saber en qué rango corregir (param)
10          CALL PWM_CONTINUE_CORRECTION_START
11          LDI PARAM,60
12          CALL DELAY_PARAM_SECONDS
13          INC iter
14          CALL MEDIR
15
16          ; Copiar la impedancia medida hacia R3:R2
17          MOV R3,R1
18          MOV R2,R0
19
20          ;VERIFICAR TMAX
21          LDI TMP,10
22          MUL TMP,R23 ; TMAX en minutos: R1:R0, como es < 90, sólo interesa R0
23          ; Aquí: R0 = TMAX; iter = TACTUAL
24          CP iter,R0
25          BREQ FINAL_CORRECCION ; Se ha cumplido el tiempo máximo
26
27
28          MUL TMP,R21 ; Decenas de la impedancia deciMegOhms
29          CLR TMP
30          ADD R0,R20  ; Unidad de la impedancia en deciMegOhms
31          ADC R1,TMP  ; Suma del carry (ya que TMP = 0)
32
33          ; R1:R0 es el valor de impedancia deseada que se comparará con R3:R2
34          CP R2,R0
35          CPC R3,R1
36          BRLO CONTINUAR_CORRIGIENDO ; Salta si R3:R2 (medida) > R1:R0 (deseada)
37
38  FINAL_CORRECCION:
39      CALL 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              LDI R16,'.'
56              CALL DATAWRT
57              LD R16,Z   ;El resultado de la corrección
58              CALL DATAWRT
59              LDI R16,'M'
60              CALL DATAWRT
61              LDI R16,'O'
62              CALL DATAWRT
63              LDI R16,'h'
64              CALL DATAWRT
65              LDI R16,'m'
66              CALL DATAWRT
67              SUB R19,R22
68
69              LDI R16,0X0F
70              CALL CMNDWRT
71
72              CALL DELAY_50ms
```

```
 73
 74
 75  BOTONES_4:
 76          IN R17,PINA
 77          BST R17,3  ;Pin 3 del puerto A (AUMENTAR)
 78          BRTC SUBIR
 79          BST R17,4  ;Pin 4 del puerto A (REDUCIR)
 80          BRTC BAJAR
 81  JMP BOTONES_4
 82
 83  SUBIR:
 84          ADIW ZL,1 ; Avanza el puntero
 85          INC R19
 86          CPI R19,17 ;Si se pasa del 16, vuelve al 1
 87          BREQ PRIMERO
 88          JMP RESULTADOS
 89  PRIMERO:
 90          LDI R19,16
 91          SBIW ZL,1
 92          JMP BOTONES_4
 93  BAJAR:
 94          SBIW ZL,5 ; Retrocede el puntero
 95          DEC R19
 96          CPI R19,0 ;Si se pasa del 1, vuelve al 16
 97          BREQ ULTIMO
 98          JMP RESULTADOS
 99  ULTIMO:
100          LDI R19,1
101          LDI ZH,HIGH(BCD_TO_ASCII_CONVERT_RAM)
102          LDI ZL,LOW(BCD_TO_ASCII_CONVERT_RAM)
103          JMP BOTONES_4
104
105
106  DIEZ_I:
107              SUBI R21,10
108              ADD R21,R22
109
110              LDI R16,'E'
111              CALL DATAWRT
112              LDI R16,'l'
113              CALL DATAWRT
114              LDI R16,'e'
115              CALL DATAWRT
116              LDI R16,'c'
117              CALL DATAWRT
118              LDI R16,'t'
119              CALL DATAWRT
120              LDI R16,'r'
121              CALL DATAWRT
122              LDI R16,'o'
123              CALL DATAWRT
124              LDI R16,'d'
125              CALL DATAWRT
126              LDI R16,'o'
127              CALL DATAWRT
128              LDI R16,' '
129              CALL DATAWRT
130              LDI R16,0x31
131              CALL DATAWRT
132              MOV R16,R21   ;El numero del electrodo(1,2,..16)
133              CALL DATAWRT
134              LDI R16,$C0   ;Esto hay que revisarlo, es para bajar a la segunda lin
     ea del LCD
135              CALL CMNDWRT
136              LD R16,Z+   ;El resultado de la corrección
137              CALL DATAWRT
138              LD R16,Z+   ;El resultado de la corrección
139              CALL DATAWRT
140              LD R16,Z+   ;El resultado de la corrección
141              CALL DATAWRT
142              LD R16,Z+   ;El resultado de la corrección
143              CALL DATAWRT
144              LDI R16,'.'
```

```
145             CALL DATAWRT
146             LD R16,Z   ;El resultado de la corrección
147             CALL DATAWRT
148             LDI R16,'M'
149             CALL DATAWRT
150             LDI R16,'O'
151             CALL DATAWRT
152             LDI R16,'h'
153             CALL DATAWRT
154             LDI R16,'m'
155             CALL DATAWRT
156
157             SUB R21,R22  ;VUELVO A LA VARIABLE
158
159
160             LDI R16,0X0F
161             CALL CMNDWRT
162
163             CALL DELAY_50ms
164
165
166             JMP BOTONES_4
167
```

```
;||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||
;|||||||||||||| 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
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```