```
;miATa1
;Programa contador
;SETA pruebas xsetaseta@gmail.com
.include"m88def.inc"

.dseg
nu1:    .byte 1     ;representación numero de 9 cifras
nu2:    .byte 1
nu3:    .byte 1
nu4:    .byte 1
nu5:    .byte 1
nu6:    .byte 1
nu7:    .byte 1
nu8:    .byte 1
nu9:    .byte 1
nu10:   .byte 1

num0:   .byte 1     ;numero de 4bytes para convertir
num1:   .byte 1
num2:   .byte 1
num3:   .byte 1

.cseg
.def temp=r16
.def temp2=r17
.def tiempo=r18
.def tiempo1=r19
.def tiempo2=r20
.def menu=r21
.def cambio=r22
.def incremento=r23
.def argu=r24   ;argumento de 1byte
.def argu2=r25

.org 0x0000

        rjmp reset
        rjmp interup0


interup0:
        ldi incremento,1
        reti

reset:
        ldi R25,LOW(ramend) ;Coloca la pila al final de la memoria RAM
        out SPL,R25
        ldi R25,HIGH(ramend)
        out SPH,R25

        ser temp        ;temp=255
        out DDRB,temp ;Puerto D como salida
        ser temp        ;temp=255
        out DDRC,temp ;Puerto C como salida
        ldi temp,0xb00000001;
        out DDRD,temp ;//puerto D 0000-1111 0=entrada 1=salida
        rcall LCD_INI
        rcall LCD_CLS
        rcall LCD_HOME

        ldi temp,1
        out EIMSK,temp  ;EIMSK – External Interrupt Mask Register
        ldi temp,2
        sts EICRA,temp  ;flanco de bajada

        sei             ;activa interrupciones
        ldi argu,10     ;espera a que se estabilice la interrupción
        rcall Delay1mS  ;Al principio se me activa una interrupción

        clr incremento
        rcall nborra
        rcall nprint
```



```
inicio:
        in  menu,PIND
menu1:
        mov temp2,menu
        andi temp2,64   ;Detecta incremento
        breq menu2
        ldi incremento,1
        ldi cambio,1
norebote:               ;Quitar esto para autoincremento
        in temp2,PIND   ;|#
        andi temp2,64   ;|#
        brne norebote   ;|#
        rjmp xmenu
menu2:
        mov temp2,menu  ;borra contador
        andi temp2,128
        breq xmenu
        rcall nborra
        ldi cambio,1
xmenu:
        andi incremento,1 ;Detecta incremento
        breq otro1
        rcall nincremento
        clr incremento
otro1:
        cpi cambio,1    ;detecta cambio
        brne nocambio
        sbi PORTD,0
        rcall nprint

        ldi ZH,HIGH(2*texto1)
        ldi ZL,LOW(2*texto1)
        ldi argu,64
        rcall PrintAtStr

        ldi argu,100
        rcall Delay1mS

        ldi ZH,HIGH(2*texto2)
        ldi ZL,LOW(2*texto2)
        ldi argu,64
        rcall PrintAtStr
        ldi argu,100        ;coloca el cursor fuera de pantalla
        rcall LCD_AT
        cbi PORTD,0
        clr cambio
nocambio:

        rjmp inicio

nincremento:
        inc r0
        brne xnincremento
        inc r1
        brne xnincremento
        inc r2
        brne xnincremento
        inc r3
xnincremento:
        ldi cambio,1
        ret
nborra:
        clr r0
        clr r1
        clr r2
        clr r3
        ret
nprint:
        sts num0,r0
        sts num1,r1
        sts num2,r2
        sts num3,r3
        rcall BINTODEC
```

```
        ldi argu,0
        ldi argu2,9
        rcall PrintAtNum

        ret



texto1:
.DB "   CAMBIO  ",0x00
texto2:
.DB "xsetaseta@GMAIL",0x00



;_____________________________________
BINTODEC:           ;Convierte num0-3 a nu1-9 decimal (num0-3 se destruyen)
        push r16
        push r17
        push r18
        push r19
        push r0
        push r1
        push r2
        push r3

        ldi r16,0x00    ;1.000.000.000
        ldi r17,0xca
        ldi r18,0x9a
        ldi r19,0x3b
        rcall divix
        sts nu10,argu

        ldi r16,0x00    ;100.000.000
        ldi r17,0xe1
        ldi r18,0xf5
        ldi r19,0x05
        rcall divix
        sts nu9,argu

        ldi r16,0x80    ;10.000.000
        ldi r17,0x96
        ldi r18,0x98
        clr r19
        rcall divix
        sts nu8,argu

        ldi r16,0x40    ;1.000.000
        ldi r17,0x42
        ldi r18,0x0f
        clr r19
        rcall divix
        sts nu7,argu

        ldi r16,0xa0    ;100.000
        ldi r17,0x86
        ldi r18,0x01
        clr r19
        rcall divix
        sts nu6,argu

        ldi r16,0x10    ;1.0000
        ldi r17,0x27
        clr r18
        clr r19
        rcall divix
        sts nu5,argu
saltoborra:
        ldi r16,0xe8    ;1000
        ldi r17,0x03
        clr r18
        clr r19
```



```
        rcall divix
        sts nu4,argu

        ldi r16,0x64    ;100
        clr r17
        clr r18
        clr r19
        rcall divix
        sts nu3,argu

        ldi r16,0x0a    ;10
        clr r17
        clr r18
        clr r19
        rcall divix
        sts nu2,argu

        lds r0,num0
        sts nu1,r0      ;1

        pop r3
        pop r2
        pop r1
        pop r0
        pop r19
        pop r18
        pop r17
        pop r16
        ret

divix:  ;Divide numero() entre r16-r19
        clr argu
        lds r0,num0
        lds r1,num1
        lds r2,num2
        lds r3,num3
divixvolver:
        sts num0,r0
        sts num1,r1
        sts num2,r2
        sts num3,r3
        sub r0,r16
        sbc r1,r17
        sbc r2,r18
        sbc r3,r19
        brcs exitdivix
        inc argu
        rjmp divixvolver
exitdivix:
        ret


PrintAtNum: ;AT=arg  Numero cifras=argu2
        rcall LCD_AT
        ldi temp,48
        subi argu2,3
        breq cifras3
        subi argu2,3
        breq cifras6
        lds argu,nu10
        add argu,temp
        rcall SENDCHAR
        lds argu,nu9
        add argu,temp
        rcall SENDCHAR
        lds argu,nu8
        add argu,temp
        rcall SENDCHAR
        lds argu,nu7
        add argu,temp
        rcall SENDCHAR
cifras6:
        lds argu,nu6
```

```
        add argu,temp
        rcall SENDCHAR
        lds argu,nu5
        add argu,temp
        rcall SENDCHAR
        lds argu,nu4
        add argu,temp
        rcall SENDCHAR
cifras3:
        lds argu,nu3
        add argu,temp
        rcall SENDCHAR
        lds argu,nu2
        add argu,temp
        rcall SENDCHAR
        lds argu,nu1
        add argu,temp
        rcall SENDCHAR
        ret

PrintAtStr: ;AT=argu , STR=z
        rcall LCD_AT
Print1:
        lpm argu,z+
        and argu,argu
        breq Print2     ;finaliza si es cero
        rcall SENDCHAR
        rjmp Print1
Print2:
        ret
LCD_CLS:
        ldi argu,1
        rcall SENDI
        ret
LCD_HOME:
        ldi argu,2
        rcall SENDI
        ret
LCD_AT:
        ori argu,128
        rcall SENDI
        ret
LCD_INI:
        ldi argu,10 ;100
        rcall Delay1mS
        ldi argu,0x03
        rcall Pon4bits

        cbi PORTB,5
        rcall ENABLE
        sbi PORTB,5
        ldi argu,3  ;30
        rcall Delay1mS
        ldi argu,0x03
        rcall Pon4bits
        cbi PORTB,5
        rcall ENABLE
        sbi PORTB,5
        ldi argu,3  ;30
        rcall Delay1mS
        ldi argu,0x03
        rcall Pon4bits
        cbi PORTB,5
        rcall ENABLE
        sbi PORTB,5
        ldi argu,3  ;30
        rcall Delay1mS

        ldi argu,0x02
        rcall Pon4bits
        cbi PORTB,5
        rcall ENABLE
        sbi PORTB,5
```



```
        ldi argu,0x2c
        rcall SENDI
        ldi argu,0x0f
        rcall SENDI
        ldi argu,0x04
        rcall SENDI
        ret

SENDCHAR:
        push argu
        push temp
        push temp2
        mov temp2,argu
        mov temp,argu
        ror temp
        ror temp
        ror temp
        ror temp
        mov argu,temp
        rcall Pon4bits
        rcall ENABLE
        mov argu,temp2
        rcall Pon4bits
        rcall ENABLE
        pop temp2
        pop temp
        pop argu
        ret

Pon4bits:
        push argu
        push temp2
        in temp2,PORTB;
        andi temp2,0xf0
        andi argu,0x0f
        or argu,temp2
        out PORTB,argu
        pop temp2
        pop argu
        ret

SENDI:
        push argu
        cbi PORTB,5
        rcall SENDCHAR
        ldi argu,1
        rcall Delay100uS    ;rcall Delay1mS
        sbi PORTB,5     ;
        pop argu
        ret

ENABLE:
        push argu
        sbi PORTB,4
        ldi argu,1
        rcall Delay100uS    ;rcall Delay1mS
        cbi PORTB,4
        pop argu
        ret

Delay10mS:
        mov tiempo,argu
zDela1: ldi tiempo1,222 ;4 mhz->9.99mS->222   20mhz->10.03mS->227
zDela2: ldi tiempo2,44  ;4 mhz->9.99mS->44   20mhz->10.03mS->220
zDela3: nop
        dec tiempo2
        brne zDela3
        nop
        dec tiempo1
        brne zDela2
        dec tiempo
        brne zDela1
```

```
        ret

Delay1mS:
        mov tiempo,argu
Dela1:  ldi tiempo1,111
Dela2:  ldi tiempo2,44  ;4 mhz->999uS->8   20mhz->999.5uS->44
Dela3:  nop
        dec tiempo2
        brne Dela3
        nop
        dec tiempo1
        brne Dela2
        dec tiempo
        brne Dela1
        ret

Delay100uS:
        mov tiempo,argu
yDela1: ldi tiempo1,11
yDela2: ldi tiempo2,44  ;4mhz->101.5uS->8  20mhz->99.5uS->44
yDela3: nop
        dec tiempo2
        brne yDela3
        nop
        dec tiempo1
        brne yDela2
        dec tiempo
        brne yDela1
        ret
```