Anexos

Descripción en VHDL del multiplicador secuencial basado en sumas y desplazamientos

```
library IEEE;
 use IEEE.STD_LOGIC_1164.ALL;
 use IEEE.STD_LOGIC_ARITH.ALL;
 use IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY long IS
PORT(ent:IN STD_LOGIC_VECTOR (7 DOWNTO 0);
      sel,sel2:IN STD_LOGIC;
      SAL:OUT STD_LOGIC_VECTOR (7 DOWNTO 0));
END ENTITY;

ARCHITECTURE ALGOR OF long IS
SIGNAL inn,a,b,SALH,SALL: STD_LOGIC_VECTOR (7 DOWNTO 0);
BEGIN
inn<= not ent;
PROCESS(sel,inn)
BEGIN
      IF SEL='1' THEN a<=inn;
                        ELSE
                              b<=inn;
      END IF;
END PROCESS;

process(a,b)
variable dataa,datab:std_logic_vector (7 downto 0);
variable c:std_logic_vector (15 downto 0):=(others=>'0');
variable cf:std_logic_vector (31 downto 0);
begin

if a(7)='1' then
dataa:=(others=>'1');
else
dataa:=(others=>'0');
end if;

if b(7)='1' then
datab:=(others=>'1');
else
datab:=(others=>'0');
end if;

cf:=c&datab&b;

for i in 1 to 16 loop
      if cf(0)='1' then
```

```
            cf(31 downto 16):=cf(31 downto 16)+(dataa&a);
            end if;
      cf(31 downto 0):='0'&cf(31 downto 1);
      end loop;

--sal<=cf(15 downto 0);
SALH<=CF(15 DOWNTO 8);
SALL<=CF(7 DOWNTO 0);

end process;

PROCESS(SEL2)
BEGIN
      IF SEL2='0'THEN
            SAL<=SALL;
            ELSE
            SAL<=SALH;
            END IF;
END PROCESS;

END ALGOR;
```

Descripción en VHDL del multiplicador basado en el algoritmo de Booth

```
library IEEE;
 use IEEE.STD_LOGIC_1164.ALL;
 use IEEE.STD_LOGIC_ARITH.ALL;
 use IEEE.STD_LOGIC_UNSIGNED.ALL;

ENTITY booth IS
PORT(ent:IN STD_LOGIC_VECTOR (7 DOWNTO 0);
      sel,sel2:IN STD_LOGIC;
      SAL:OUT STD_LOGIC_VECTOR (7 DOWNTO 0));
END ENTITY;

ARCHITECTURE ALGOR OF booth IS
SIGNAL inn,A,B,SALH,SALL: STD_LOGIC_VECTOR (7 DOWNTO 0);
BEGIN
inn<= not ent;
PROCESS(sel,inn)
BEGIN
      IF SEL='1' THEN A<=inn;
                        ELSE
                              B<=inn;
      END IF;
END PROCESS;
```

```
PROCESS(A,B)

VARIABLE C:STD_LOGIC_VECTOR (8 DOWNTO 0):=(OTHERS=>'0');
VARIABLE CF:STD_LOGIC_VECTOR (16 DOWNTO 0);
VARIABLE X,T:STD_LOGIC;
VARIABLE DATAA:STD_LOGIC_VECTOR (8 DOWNTO 0);

BEGIN

IF A(7)='1' THEN
DATAA:='1'&A;
ELSE
DATAA:='0'&A;
END IF;

X:='0';
T:='0';
CF:=C&B;

FOR I IN 1 TO 8 LOOP
      IF CF(0)='0' AND X='1' THEN
      CF(16 DOWNTO 8):=CF(16 DOWNTO 8)+DATAA;
      ELSIF CF(0)='1' AND X='0' THEN
      CF(16 DOWNTO 8):=CF(16 DOWNTO 8)-DATAA;
      END IF;
      T:=CF(16);
      X:=CF(0);
      CF(16 DOWNTO 0):=T&CF(16 DOWNTO 1);
END LOOP;

--SAL<=CF(15 DOWNTO 0);

SALH<=CF(15 DOWNTO 8);
SALL<=CF(7 DOWNTO 0);


END PROCESS;

PROCESS(SEL2)
BEGIN
      IF SEL2='0'THEN
            SAL<=SALL;
            ELSE
            SAL<=SALH;
            END IF;
END PROCESS;

END ALGOR;
```

Programa en lenguaje ensamblador empleado para programar el PIC

```
;*********************************************************************
******

      LIST P=18F452          ;directiva    para    definir    procesador
especifico
      #include <P18F452.INC>  ;Archivo  de  definicion  de  variables  del
procesador

;*********************************************************************
******


;Registros de Configuracion  SINTAXIS AL FINAL DE ESTE ARCHIVO

;  SELECCION DE OSCILADOR:
      CONFIG    OSC = HSPLL             ; OSCILADOR alta velocidad pll
;   CAMBIO DE OSCILADOR:
      CONFIG      OSCS = OFF           ;DESHABILITADO
;   Power-up Timer DE RESET AL ENCENDIDO:
      CONFIG    PWRT = ON        ;HABILITADO
;   RESET Brown-out POR BAJO VOLTAJE:
      CONFIG      BOR = OFF          ;HABILITADO
;   VOLTAJE Brown-out:
      CONFIG    BORV = 45       ; 4.5V
;   Watchdog Timer:
      CONFIG    WDT = OFF       ; HABILITADO
;  POSTESCALADOR DEL Watchdog:
      CONFIG    WDTPS = 128   ; 1:128
;   CCP2 MUX:
      CONFIG    CCP2MUX = OFF      ;DESHABILITADO=(RB3) HABILITADO= (RC1)
;   RESET POR DESBORDE DE Stack:
      CONFIG    STVR = OFF    ; DESHABILITADO
;   PROGRAMACION ICSP CON BAJO VOLTAJE:
      CONFIG    LVP = OFF       ;DESHABILITADO
;   Background Debugger:
          CONFIG  DEBUG = OFF     ;DESHABILITADO
;   PROTECCION DE CODIGO BLOQUE 0:
      CONFIG      CP0 = OFF       ;DESHABILITADO
;   PROTECCION DE CODIGO BLOQUE 1:
      CONFIG      CP1 = OFF       ;DESHABILITADO
;   PROTECCION DE CODIGO BLOQUE 2:
      CONFIG      CP2 = OFF       ;DESHABILITADO
;   PROTECCION DE CODIGO BLOQUE 3:
```

```
      CONFIG      CP3 = OFF      ;DESHABILITADO
;   PROTECCION DE CODIGO SECTOR DE BOOT:
      CONFIG      CPB = OFF      ;DESHABILITADO
;   PROTECCION DE DATOS EEPROM:
      CONFIG      CPD = OFF      ;DESHABILITADO
;   PROTECCION DE ESCRITURA BLOQUE 0:
      CONFIG      WRT0 = OFF     ;DESHABILITADO
;   PROTECCION DE ESCRITURA BLOQUE 1:
      CONFIG      WRT1 = OFF     ;DESHABILITADO
;   PROTECCION DE ESCRITURA BLOQUE 2:
      CONFIG      WRT2 = OFF     ;DESHABILITADO
;   PROTECCION DE ESCRITURA BLOQUE 3:
      CONFIG      WRT3 = OFF     ;DESHABILITADO
;   PROTECCION DE ESCRITURA BLOQUE DE BOOT:
      CONFIG      WRTB = OFF     ;DESHABILITADO
;   PROTECCION DE ESCRITURA DEL BLOQUE DE CONFIGURACION:
      CONFIG      WRTC = OFF     ;DESHABILITADO
;   PROTECCION Datos EEPROM :
      CONFIG      WRTD = OFF     ;DESHABILITADO
;   PROTECCION DE LECTURA DE TABLA BLOQUE 0:
      CONFIG      EBTR0 = OFF    ;DESHABILITADO
;   PROTECCION DE LECTURA DE TABLA BLOQUE 1:
      CONFIG      EBTR1 = OFF    ;DESHABILITADO
;   PROTECCION DE LECTURA DE TABLA BLOQUE 2:
      CONFIG      EBTR2 = OFF    ;DESHABILITADO
;   PROTECCION DE LECTURA DE TABLA BLOQUE 3:
      CONFIG      EBTR3 = OFF    ;DESHABILITADO
;   Boot Block Table Read Protection:
      CONFIG      EBTRB = OFF    ;DESHABILITADO


;***********************************************
;definicion de constantes
;***********************************************

VALOR1      EQU         0x08
VALOR2      EQU         0x09
DATO  EQU         0x0A


;*****************************************************************************
******
;Definicion de Variables en RAM.

            CBLOCK      0x000
            AUX1
            MULA
            MULB
            BCDH
```

```
            BCDL
            CUENTA
            H_BYTE
            L_BYTE
            GTSDFVUH
            R0
            BASURA2
            BFTRS
            R1
            BASURA
            R2
            count
            FSR
            temp
            BIN
            BCD_TEMP
            TEMP1
            TEMP2
            RETRA1
            RETRA2
            RETRA3
            RETRA4
            ENDC


; EJEMPLO DE UN STACK PARA LA INTERRUPCION DE BAJA PRIORIDAD

            CBLOCK      0x080
            WREG_TEMP   ;VARIABLE PARA RESPALDO DE CONTEXTO
            STATUS_TEMP ;VARIABLE PARA RESPALDO DE CONTEXTO
            BSR_TEMP    ;VARIABLE PARA RESPALDO DE CONTEXTO
            ENDC


;*****************************************************************************
******
;DATOS DE EEPROM
; LOS DATOS A PROGRAMAR EN LA EEPROM SE PUEDEN DEFINIR DESDE AQUI

      ;     ORG   0xf00000

      ;     DE    "Test Data",0,1,2,3,4,5,0X23,0XFB

;*****************************************************************************
******
;VECTOR DE RESET
; AL OCURRIR UN RESET EL CODIGO A PARTIR DE 0X0000 SERA EJECUTADO
```

```
            ORG   0x0000

            goto  INICIO            ;BRINCA AL INICIO DEL CODIGO
                                ;ESTE BRICO SE PONE PARA NO COLISIONAR
                                ;CON LOS VECTORES DE INTERRUPCION
                                ;SI NO HAY INTERRUPCIONES SE PUEDE OMITIR EL
SALTO.

;*********************************************************************
******
;VECTOR DE INTERRUPCION DE ALTA PRIORIDAD

            ORG   0x0008

            goto  HighInt           ;BRINCO A LA RUTINA DE INTERRUPCION
                                ;DE ALTA PRIORIDAD

;*********************************************************************
******
; vECTOR DE INTERRUPCION DE BAJA PRIORIDAD
; This code will start executing when a low priority interrupt occurs.

            ORG   0x0018
            goto  LowInt   ;BRINCO A LA RUTINA DE INTERRUPCION DE BAJA
PRIORIDAD


;SI LA MEMORIA DE PROGRAMA ES CRITICA, NO SE USARIAN LOS SALTOS PARA
REDIRIGIR LAS INTERRUPCIONES
;Y LAS RUTINAS COMENZARIAN EN LAS DIRECCIONES DE LOS VECTORES USANDO UN
SALTO PARA LLEGAR A INICIO




;*********************************************************************
******
;INICIO DEL PROGRAMA PRINCIPAL
; EL CODIGO DEL PROGRAMA PRINCIPAL COMIENZA AQUI.

INICIO

            CLRF        TRISD ;PUERTO D SALIDAS (DISPLAY)
            MOVLW 0XFF
            MOVWF TRISC ;PUERTO C ENTRADAS (MULTIPLICANDOS)
            MOVWF TRISB ;PUERTO B ENTRADAS (RESULTADO)
            BCF         TRISA,0     ; PORTA 0 SALIDA (SEL2)
            BCF         TRISA,1     ; PORTA 1 SALIDA (SEL)
```

```
            CALL  INILCD                        ;INICIALIZAR DISPLAY
START BSF         PORTA,1
            CLRF  MULA

            CLRF  MULB
            CLRF  H_BYTE
            CLRF  L_BYTE


AQUI
            BCF         AUX1,0                  ;LIMPIAR BANDERAS DE SIGNO
            BCF         AUX1,1
            BCF         AUX1,2

            BSF         PORTA,1
            CALL  RET100MS
            CALL  RET100MS
            CALL  RET100MS
            CALL  RET100MS


LEEA  MOVF   PORTC,W                ;LEER MULA
            COMF  WREG
            btfsc WREG,7                  ;CHECAR SIGNO DE MULA
            CALL  COM2A
            MOVWF MULA

            CLRF  WREG

            CALL  RET100MS
            CALL  RET100MS


            BCF         PORTA,1

            CALL  RET100MS
            CALL  RET100MS



LEEB  MOVF   PORTC,W                ;LEER MULB
            COMF  WREG
            btfsc WREG,7                  ;CHECAR SIGNO DE MULB
            CALL  COM2B
            MOVWF MULB

      ;     BSF         PORTA,1
            BSF         PORTA,0
```

```
            CALL  RET100MS



LEEH  MOVF   PORTB,W

            MOVWF H_BYTE

            CALL  RET100MS
            CALL  RET100MS


            BCF         PORTA,0

            CALL  RET100MS
            CALL  RET100MS
            CALL  RET100MS



LEEL  MOVF   PORTB,W
            MOVWF L_BYTE

            CALL  RET100MS

            BSF         PORTA,0

            MOVF  PORTB,W
            btfsc WREG,7
            call  COM216

            BCF         PORTA,0



            MOVLW 0X80
            CALL LCDCOM

            MOVLW "M"                           ;MOSTRAR "MULTIPLICACION"
            CALL  LCDDAT
            MOVLW "U"
            CALL  LCDDAT
            MOVLW "L"
            CALL  LCDDAT
            MOVLW "T"
            CALL  LCDDAT
            MOVLW "I"
            CALL  LCDDAT
```

```
            MOVLW "P"
            CALL  LCDDAT
            MOVLW ":"
            CALL  LCDDAT



            BTFSC AUX1,0
            CALL  MENOS                 ;SI  AUX1,0  ESTA  ENCENDIDO  SE
IMPRIME "-"

            MOVFF MULA,BIN
            CALL  BIN2BCD
            MOVF  BCDH,W
            CALL  BYTEDIS
            MOVF  BCDL,W
            CALL BYTEDIS                      ;MOSTRAR MULA


            MOVLW "x"
            CALL  LCDDAT

            MOVLW " "
            CALL  LCDDAT

            MOVLW 0XC0
            CALL  LCDCOM

            BTFSC AUX1,1
            CALL  MENOS                 ;SI  AUX1,0  ESTA  ENCENDIDO  SE
IMPRIME "-"

            CLRF  BIN


            MOVFF MULB,BIN
            CALL  BIN2BCD
            MOVF  BCDH,W
            CALL  BYTEDIS
            MOVF  BCDL,W
            CALL  BYTEDIS                     ;MOSTRAR MULB

            MOVLW "="
            CALL  LCDDAT

            BTFSC AUX1,2
            CALL  MENOS

            CALL  B2_BCD
```

```
            MOVF  R0,W
            CALL  BYTEDIS
            MOVF  R1,W
            CALL  BYTEDIS
            MOVF  R2,W
            CALL  BYTEDIS                ;MOSTRAR RESULTADO

            MOVLW " "
            CALL  LCDDAT

            GOTO START
            NOP
            NOP

            GOTO INICIO
            nop
            nop


;TRABA      GOTO  TRABA


;*********************************************************************
******
;                       SUBRUTINAS
;*********************************************************************
******

MENOS MOVLW "-"
            CALL LCDDAT
            RETURN

INILCD      CLRF  TRISD             ; Puerto D como salida
            MOVLW b'00000000'       ; RS = 0, RW = 0
            MOVWF PORTD
            MOVLW b'00101000'       ; LCD en 4 bits, E=1
            MOVWF   PORTD
            NOP
            NOP                                 ; Minimo 220 ns
            MOVLW b'00100000'       ; E=0
            MOVWF   PORTD

            MOVLW b'00101000'       ; Funtion Set: 4 bits, 2 line, 5x7 dot
            CALL  LCDCOM
            ;MOVLW      b'00001110'       ;         Display         ON/OFF
Control:Display/Cursor ON/Blink ON
        MOVLW     b'00001111'       ;            Display            ON/OFF
Control:Display/Cursor OFF/Blink O
```

```
            CALL  LCDCOM
            MOVLW b'00000001'       ; Clear Display
            CALL  LCDCOM
            RETURN

LCDDAT      MOVWF DATO
            CALL  BUSSY
            MOVLW b'00000100'       ; RS=1, RW=0
            MOVWF   PORTD
            MOVF  DATO, W
            ANDLW b'11110000'       ; Parte Alta del dato en W
            IORLW   b'00001100'           ;  W  con  parte  alta  del  dato,
RW=0, RS=1, E=1
            MOVWF PORTD
            ANDLW b'11110111'
            NOP
            MOVWF PORTD             ; E=0
            SWAPF DATO,W                  ; Inversion Nibble
            ANDLW b'11110000'       ; Parte baja en W
            IORLW   b'00001100'           ;  W  con  parte  baja  del  dato,
RW=0, RS=1, E=1
            MOVWF PORTD
            ANDLW b'11110111'
            NOP
            MOVWF PORTD             ; E=0
            RETURN

LCDCOM      MOVWF DATO
            CALL  BUSSY
            MOVLW b'00000000'       ; RS=0, RW=0
            MOVWF   PORTD
            MOVF  DATO, W
            ANDLW b'11110000'       ; Parte Alta del dato en W
            IORLW   b'00001000'           ;  W  con  parte  alta  del  dato,
RW=0, RS=0, E=1
            MOVWF PORTD
            ANDLW b'11110111'
            NOP
            MOVWF PORTD             ; E=0
            SWAPF DATO,W                  ; Inversion Nibble
            ANDLW b'11110000'       ; Parte baja en W
            IORLW   b'00001000'           ;  W  con  parte  baja  del  dato,
RW=0, RS=0, E=1
            MOVWF PORTD
            ANDLW b'11110111'
            NOP
            MOVWF PORTD             ; E=0
            RETURN
```

```
BUSSY MOVLW 0x00A0
            MOVWF VALOR1
            MOVLW 0x00A0
            MOVWF VALOR2
LOOP  DECFSZ      VALOR1
            GOTO  LOOP
            DECFSZ      VALOR2
            GOTO  LOOP
            RETURN

;*********************************************************************
***
BIN2BCD
            clrf BCDH
            clrf BCDL
BCD_HIGH
            movlw .100
            subwf BIN,f
            btfss STATUS,C
            goto SUMA_100
            incf BCDH,f
            goto BCD_HIGH
SUMA_100
            movlw .100
            addwf BIN,f
            movlw 0x0F
            movwf BCDL
BCD_LOW movlw .10
            subwf BIN,f
            btfss STATUS,C
            goto SUMA_10
            incf BCDL,f
            movlw 0x0F
            iorwf BCDL,f
            goto BCD_LOW
SUMA_10 movlw .10
            addwf BIN,f
            movlw 0xF0
            andwf BCDL,f
            movf BIN,w
            iorwf BCDL,f
            return
;####################################################################


B2_BCD                                          ;CONVERSION  DE  BINARIO  16
BITS A BCD
    bcf      STATUS,C
    CLRF    count
```

```
    movlw    .16
    movwf    count
    clrf     R0
    clrf     R1
    clrf     R2
Loop16
    rlCf      L_BYTE
    rlCf      H_BYTE
    rlCf      R2
    rlCf      R1
    rlCf      R0
    decfsz   count
    goto     adjDEC
    RETURN
adjDEC
    movlw    R2
    movWf    FSR0L
    call     adjBCD
;
    movlw    R1
    movwf    FSR0L
    call     adjBCD
;
   movlw    R0
    movwf    FSR0L
    call     adjBCD
;
    goto     Loop16
;
      ;movlw R2 ; load R2 as indirect address ptr
      ;movwf FSR0L
      ;call adjBCD

      ;incf FSR0, F
      ;call adjBCD

       ;incf FSR0, F
      ;call adjBCD

      ; goto Loop16
adjBCD
   ; movlw    3
    ;addwf    0,W
  ;  movwf    temp
   ; btfsc    temp,3
   ; movwf    0
    ;movlw    30
    ;addwf    0,W
    ;movwf    temp
```

```
    ;btfsc    temp,7
    ;movwf    0
    ;RETURN
    movfF INDF0,WREG
      addlw 0x03
      btfsc WREG,3 ; test if result > 7
      movwf INDF0
      movfF INDF0,WREG
      addlw 0x30
      btfsc WREG,7 ; test if result > 7
      movwf INDF0 ; save as MSD
      return
;###############################################################################
##

COM2A COMF  WREG                    ;RUTINA PARA HACER COMPLEMENTO A 2 DE
A, Y ACTIVAR BANDERA DE SIGNO
            ADDLW 0X01
            BSF         AUX1,0
            RETURN

COM2B COMF  WREG                    ;RUTINA PARA HACER COMPLEMENTO A 2 DE
B, Y ACTIVAR BANDERA DE SIGNO
            ADDLW 0X01
            BSF         AUX1,1
            RETURN

COM216      COMF  L_BYTE,F                      ;RUTINA     PARA     HACER
COMPLEMENTO A 2 DE RESULTADO Y ACTIVAR BANDERA DE SIGNO
            INCF  L_BYTE,F
            BTFSC STATUS,Z
            DECF  H_BYTE,F
            COMF  H_BYTE,F
            BSF         AUX1,2
            RETURN

;*********************************************
;subrutina de escritura de un byte hex a display
;ENTRADA : DATO EN W. SALIDA: NINGUNA
;*********************************************
;SUBRUTINA QUE CONVIERTE DATOS DEL CONVERTIDOR A HEXADECIMAL Y LOS MANDA
AL DISPLAY

BYTEDIS
      CLRF  TEMP1
      CLRF  TEMP2
      MOVWF TEMP1
      SWAPF TEMP1,W
      ANDLW 0X0F
```

```
      MOVWF TEMP2
      MOVLW 0X0A
      CPFSLT      TEMP2
      GOTO  BYTEDIS1
      MOVF  TEMP2,W
      ADDLW 0X30
      CALL  LCDDAT
      GOTO  BYTEDIS2
BYTEDIS1    MOVF  TEMP2,W
      ADDLW 0X37
      CALL  LCDDAT
BYTEDIS2    MOVF  TEMP1,W
      ANDLW 0X0F
      MOVWF TEMP2
      MOVLW 0X0A
      CPFSLT      TEMP2
      GOTO  BYTEDIS3
      MOVF  TEMP2,W
      ADDLW 0X30
      CALL  LCDDAT
      RETURN
BYTEDIS3    MOVF  TEMP2,W
      ADDLW 0X37
      CALL  LCDDAT
      RETURN


BYTEDISHH   MOVWF TEMP1
      SWAPF TEMP1,W
      ANDLW 0X0F
      MOVWF TEMP2
      MOVLW 0X0A
      CPFSLT      TEMP2
      GOTO  BYTEDIS1H
      MOVF  TEMP2,W
      ADDLW 0X30
      CALL  LCDDAT
      RETURN
BYTEDIS1H   MOVF  TEMP2,W
      ADDLW 0X37
      CALL  LCDDAT
      RETURN


BYTEDISLH   MOVWF TEMP1
      MOVF  TEMP1,W
      ANDLW 0X0F
      MOVWF TEMP2
      MOVLW 0X0A
```

```
      CPFSLT      TEMP2
      GOTO  BYTEDIS3L
      MOVF  TEMP2,W
      ADDLW 0X30
      CALL  LCDDAT
      RETURN
BYTEDIS3L   MOVF  TEMP2,W
      ADDLW 0X37
      CALL  LCDDAT
      RETURN


RET20MS
      MOVLW 0X9A
      MOVWF RETRA2,A
      CLRF  RETRA1,A
RET20_1     DECFSZ      RETRA1,F,A
      GOTO  RET20_1
      DECFSZ      RETRA2,F,A
      GOTO  RET20_1
      RETURN

RET100MS
      MOVLW 0X05
      MOVWF RETRA3,A
RET100MS_1  CALL  RET20MS
             DECFSZ    RETRA3,F,A
          GOTO      RET100MS_1
          RETURN

                NOP
                NOP
                RETURN



;*********************************************************************
******
;                 RUTINAS DE INTERRUPCION
;*********************************************************************
******
; RUTINA DE INTERRUPCION DE ALTA PRIORIDAD


HighInt:

;     *** CODIGO DE RUTINA DE ALTA PRIORIDAD VA AQUI ***
```

```
            retfie      FAST


;*********************************************************************
******
; RUTINA DE INTERRUPCION DE BAJA PRIORIDAD


LowInt            movff STATUS,STATUS_TEMP      ;save STATUS register
            movff WREG,WREG_TEMP          ;save working register
            movff BSR,BSR_TEMP            ;save BSR register

;     *** CODIGO DE RUTINA DE BAJA PRIORIDAD VA AQUI ***


            movff BSR_TEMP,BSR            ;restore BSR register
            movff WREG_TEMP,WREG          ;restore working register
            movff STATUS_TEMP,STATUS      ;restore STATUS register
            retfie


;*********************************************************************
******
;FIN DEL PROGRAMA. REQUIERE LA DIRECTIVA END

            END
```