Programa del pic

```
LIST P=16F874

porta EQU 0X05      ; Definim els ports que haurem d'utilitzar i les seves corresponents adreces
TRISA EQU 0X05      ; (les quals que podem veure en la taula que anteriorment els hem mostrat
portb EQU 0X06      ; dels bancs de registres de la memòria de dades).
TRISB EQU 0X06
portc EQU 0X07
TRISC EQU 0X07
portd EQU 0X08
TRISD EQU 0X08

INTCON EQU 0X0B     ; Aquests són registres que haurem d'utilitzar durant el programa de memòria,
ADCON1 EQU 0X9F     ; són registres especials que serveixen per definir el comportament del PIC
OPCION EQU 0X01     ; o per obtenir informació sobre aquest (p.ex. l'OPCION ens permet saber si
STATUS EQU 0X03     ; el resultat d'una operació és 0, té carry, etc).
pc EQU 0X02

; Tots els següents registres són de propòsit general i s'utilitzen com a variables.

ANTERIOR EQU 0X31   ; Aquests dos serviran perl'encoder.
ACTUAL EQU 0X32

DIG0 EQU 0X33       ; Aquests cinc serviran pel display.
DIG1 EQU 0X34
DIG2 EQU 0X35
DIG3 EQU 0X36
DIG EQU 0X37

guarda1 EQU 0X38    ; Aquests dos serviran per guardar els valors de W i de l'STATUS en cas que es
guarda2 EQU 0X39    ; produeixi una interrupció i així poder-los recuperar quan l'interrupció s'acabi.

N1 EQU 0X040  ; Aquests set serviran pel PLL.
N11 EQU 0X041
N2 EQU 0X042
N22 EQU 0X043
BIT EQU 0X044

ORG 0X00
                    ; Fem un salt a la memòria de programa 0x05 (inici) ja que cal que saltem la posició de
    GOTO inici      ; memòria 0x04 que és on es situa el punter de programa en cas que es produeixi una
                    ; interrupció i al iniciar el programa no hem d'anar a la rutina d'interrupció (RSI).

ORG 0X04
                    ; Salt a la rutina d'interrupció, doncs si es produeix una interrupció el punter de
    GOTO RSI        ; programa es situa aqui (0x04) i cal fer un salt per tractar les interrupcions en un
                    ; altre punt de la memòria de programa (RSI).

ORG 0X05

 inici
    BSF STATUS,5        ; Ens situem al banc 1 per inicialitzar els registres d'aquest banc.

    MOVLW 0X06          ; Inicialitzem el registre ADCON1 que serveix per decidir si el port a treballa
    MOVWF ADCON1        ; de forma digital o analògica. Volem que sigui digital.

    CLRF TRISA      ; Definim el port A com a sortida.
    CLRF TRISC      ;  Definim el port C com a sortida.
    CLRF TRISD      ;  Definim el port D com a sortida.

    MOVLW 0XFF          ;  Definim el port B com a sortida.
    MOVWF TRISB
```

```
MOVLW 0XE0      ; Inicialitzem el registre OPCION, desabilitem els pull-ups, i definim que les
MOVWF OPCION    ; interrupcions de timer 0 i de wach dog es produeixin en el flanc de pujada.

BCF STATUS,5    ;  Ens situem al banc 0.

MOVLW 0X88      ; Inicialitzem el registre INTCON, habilitem la interrupció de canvi en el port B
MOVWF INTCON    ; (RB4 - 7) i deshabiliem la resta (les perifèriques, la de timer 0 i la de RB0).

CLRF portd      ; Posem el port D a 0 perquè cap led del display estigui il·luminat.

MOVLW 0XFF      ; Encenem el led que indica que l'aparell està engegat i inicialitzem el display, cap
MOVWF porta     ; dels dígits del display està il·luminat.

MOVLW 0X00      ; Iniciaitzem els dígits del display al número 100,0.
MOVWF DIG0
MOVLW 0X00
MOVWF DIG1
MOVLW 0X00
MOVWF DIG2
MOVLW 0X01
MOVWF DIG3

MOVWF DIG   ; Posem la el registre DIG a 1 per tal qe comensi a escriure pel 1r dígit del display.

MOVFW portb         ; Llegim els valors del port B, és a dir de l'encoder , doncs ens caldrà
MOVWF ANTERIOR          ; el valor inicial de l'"encoder" perquè quan varii podem relacionar el
                    ; nou valor que ens dóna amb l'anterior i poguem saber a quina direcció
                    ; ens cal girar

MOVLW 0X30          ; Fem una and per eliminar aquells valors del port B que no ens interessen
ANDWF ANTERIOR,1    ; doncs només ens calen RB4 i 5 per a utilitzar l'encoder.

CLRF portc      ; Posem el clock i l'LE del PLL a 0, de moment no introduïm els valors dels divisors.

BSF portc,3     ;Posem el PWDN a nivell alt, doncs no volem que el PLL estigui en mode baix consum.

MOVLW 0XE8      ; Inicialitzem el valor del divisor N i N2 per tal que el PLL emeti inicialment a 100MHz.
MOVWF N1
MOVLW 0X03
MOVWF N2

CLRF BIT    ; Inicialitzem el valor del registre BIT que ens servirà per saber quin bit estem enviant al PLL.

CALL escriureN ; Cridem la subrutina que envia al PLL el valor del divisor N.

; La següent seqüència serveix per inicialitzar el valor del divisor R (que serà 200 per tenir una sensibilitat de
; 100KHz. ), recordem que cal enviar els bits en sèrie.

BSF portc,1     ; Escriure el bit 15 del divisor R.
BSF portc,0     ; Posem el clock del PLL a 1 i després a 0 perquè el PLL llegeixi el valor del bit.
BCF portc,0

BCF portc,1     ; Escriure el bit 14 del divisor R.
BSF portc,0     ; Posem el clock del PLL a 1 i després a 0 perquè el PLL llegeixi el valor del bit.
BCF portc,0
```

```
BCF portc,1     ; Escriure el bit 13 del divisor R.
BSF portc,0     ; Posem el clock del PLL a 1 i després a 0 perquè el PLL llegeixi el valor del bit.
BCF portc,0

BCF portc,1     ; Escriure el bit 12 del divisor R.
BSF portc,0     ; Posem el clock del PLL a 1 i després a 0 perquè el PLL llegeixi el valor del bit.
BCF portc,0

BCF portc,1     ; Escriure el bit 11 del divisor R.
BSF portc,0     ; Posem el clock del PLL a 1 i després a 0 perquè el PLL llegeixi el valor del bit.
BCF portc,0

BCF portc,1     ; Escriure el bit 10 del divisor R.
BSF portc,0     ; Posem el clock del PLL a 1 i després a 0 perquè el PLL llegeixi el valor del bit.
BCF portc,0

BCF portc,1     ; Escriure el bit 9 del divisor R.
BSF portc,0     ; Posem el clock del PLL a 1 i després a 0 perquè el PLL llegeixi el valor del bit.
BCF portc,0

BSF portc,1     ; Escriure el bit 8 del divisor R.
BSF portc,0     ; Posem el clock del PLL a 1 i després a 0 perquè el PLL llegeixi el valor del bit.
BCF portc,0

BSF portc,1     ; Escriure el bit 7 del divisor R.
BSF portc,0     ; Posem el clock del PLL a 1 i després a 0 perquè el PLL llegeixi el valor del bit.
BCF portc,0

BCF portc,1     ; Escriure el bit 6 del divisor R.
BSF portc,0     ; Posem el clock del PLL a 1 i després a 0 perquè el PLL llegeixi el valor del bit.
BCF portc,0

BCF portc,1     ; Escriure el bit 5 del divisor R.
BSF portc,0     ; Posem el clock del PLL a 1 i després a 0 perquè el PLL llegeixi el valor del bit.
BCF portc,0

BSF portc,1     ; Escriure el bit 4 del divisor R.
BSF portc,0     ; Posem el clock del PLL a 1 i després a 0 perquè el PLL llegeixi el valor del bit.
BCF portc,0

BCF portc,1     ; Escriure el bit 3 del divisor R.
BSF portc,0     ; Posem el clock del PLL a 1 i després a 0 perquè el PLL llegeixi el valor del bit.
BCF portc,0

BCF portc,1     ; Escriure el bit 2 del divisor R.
BSF portc,0     ; Posem el clock del PLL a 1 i després a 0 perquè el PLL llegeixi el valor del bit.
BCF portc,0

BCF portc,1     ; Escriure el bit 1 del divisor R.
BSF portc,0     ; Posem el clock del PLL a 1 i després a 0 perquè el PLL llegeixi el valor del bit.
BCF portc,0

BSF portc,1     ; Escriure el bit C
BSF portc,0     ; Posem el clock del PLL a 1 i després a 0 perquè el PLL llegeixi el valor del bit.
BCF portc,0

BSF portc,2     ; Posem el bit LE del PLL a 1 i després a 0 per tal de cargar el valor del divisor R
BCF portc,2     ; que acabem d'introduir al lach.
```

```
; Ara repetirem la seqüència però posarem el bit C (l'últim bit) a 0 en lloc de 1, doncs el fabricant
; recomana realitzar aqueses seqüències per introduir el valor del divisor R al PLL.

    BCF portc,1     ; Escriure el bit 15 del divisor R.
    BSF portc,0     ; Posem el clock del PLL a 1 i després a 0 perquè el PLL llegeixi el valor del bit.
    BCF portc,0

    BCF portc,1     ; Escriure el bit 14 del divisor R.
    BSF portc,0     ; Posem el clock del PLL a 1 i després a 0 perquè el PLL llegeixi el valor del bit.
    BCF portc,0

    BCF portc,1     ; Escriure el bit 13 del divisor R.
    BSF portc,0     ; Posem el clock del PLL a 1 i després a 0 perquè el PLL llegeixi el valor del bit.
    BCF portc,0

    BCF portc,1     ; Escriure el bit 12 del divisor R.
    BSF portc,0     ; Posem el clock del PLL a 1 i després a 0 perquè el PLL llegeixi el valor del bit.
    BCF portc,0

    BCF portc,1     ; Escriure el bit 11 del divisor R.
    BSF portc,0     ; Posem el clock del PLL a 1 i després a 0 perquè el PLL llegeixi el valor del bit.
    BCF portc,0

    BCF portc,1     ; Escriure el bit 10 del divisor R.
    BSF portc,0     ; Posem el clock del PLL a 1 i després a 0 perquè el PLL llegeixi el valor del bit.
    BCF portc,0

    BCF portc,1     ; Escriure el bit 9 del divisor R.
    BSF portc,0     ; Posem el clock del PLL a 1 i després a 0 perquè el PLL llegeixi el valor del bit.
    BCF portc,0

    BSF portc,1     ; Escriure el bit 8 del divisor R.
    BSF portc,0     ; Posem el clock del PLL a 1 i després a 0 perquè el PLL llegeixi el valor del bit.
    BCF portc,0

    BSF portc,1     ; Escriure el bit 7 del divisor R.
    BSF portc,0     ; Posem el clock del PLL a 1 i després a 0 perquè el PLL llegeixi el valor del bit.
    BCF portc,0

    BCF portc,1     ; Escriure el bit 6 del divisor R.
    BSF portc,0     ; Posem el clock del PLL a 1 i després a 0 perquè el PLL llegeixi el valor del bit.
    BCF portc,0

    BCF portc,1     ; Escriure el bit 5 del divisor R.
    BSF portc,0     ; Posem el clock del PLL a 1 i després a 0 perquè el PLL llegeixi el valor del bit.
    BCF portc,0

    BSF portc,1     ; Escriure el bit 4 del divisor R.
    BSF portc,0     ; Posem el clock del PLL a 1 i després a 0 perquè el PLL llegeixi el valor del bit.
    BCF portc,0
```

```
    BCF portc,1     ; Escriure el bit 3 del divisor R.
    BSF portc,0     ; Posem el clock del PLL a 1 i després a 0 perquè el PLL llegeixi el valor del bit.
    BCF portc,0

    BCF portc,1     ; Escriure el bit 2 del divisor R.
    BSF portc,0     ; Posem el clock del PLL a 1 i després a 0 perquè el PLL llegeixi el valor del bit.
    BCF portc,0

    BCF portc,1     ; Escriure el bit 1 del divisor R.
    BSF portc,0     ; Posem el clock del PLL a 1 i després a 0 perquè el PLL llegeixi el valor del bit.
    BCF portc,0

    BSF portc,1     ; Escriure el bit C
    BSF portc,0     ; Posem el clock del PLL a 1 i després a 0 perquè el PLL llegeixi el valor del bit.
    BCF portc,0

    BSF portc,2     ; Posem el bit LE del PLL a 1 i després a 0 per tal de cargar el valor del divisor R
    BCF portc,2     ; que acabem d'introduir al lach, ja tenim els comptadors introduïts al PLL.

bucle   ; Aquí tenim el bucle que s'anirà repetint per mantenir els dígits del display encesos al
        ; al valor desitjat. Recordem que cal fer un escombrat continu erquè el display funcioni.

    BTFSC DIG,0     ; Observem quin dels dígits cal escriure segons on tinguem un 1 en el registre
    GOTO digit0     ; DIG (si està a la primera posició, bit 0, toca escriure el dígit0 si esta al bit1 toca
    BTFSC DIG,1     ; escriure el digit1, etc.). Utilitzem un salt si 0, per tant si hi ha un 1 al bit 0 del
    GOTO digit1     ;  registre DIG no saltarà i farem la següent instrucció, que és un salt per tractar
    BTFSC DIG,2     ; el digit0; si en canvi hi ha un 0, farem un salt i per tant no tractarem el digit0
    GOTO digit2     ; sino que mirarem el bit 1del registre DIG, iigual que abans si hi ha un 1 tractarem
    BTFSC DIG,3     ; (en aquest cas) el digit1 i si hi ha un 0 farem un salt i mirarem si toca escriure
    GOTO digit3     ; el dígit2, etc.

digit0
    MOVFW DIG0  ; Posem el valor del digit0 al registre W.

    CALL taula      ; Criem la taula per saber quina combinació cal escriure pel port D, i així posar
    MOVWF portd ; el N° desitjat al display.

    BCF porta,3     ; Escrivim el dig0 al primer dígit del display.
    RLF DIG         ; Rotem el registre DIG, així el pròxim bucle escriurem el dígit1.
    BSF porta,3     ; Tornem el port A a 0 per tal que el display no mostri cap número.
    CLRF portd      ; Netegem el port D.
    GOTO bucle      ; Tornem al bucle.

digit1
    MOVFW DIG1  ; Posem el valor del digit1 al registre W.

    CALL taula2     ; Criem la taula per saber quina combinació cal escriure pel port D, i així posar
    MOVWF portd ; el N° desitjat al display.

    BCF porta,2     ; Escrivim el digit1 al segon dígit del display.
    RLF DIG         ; Rotem el registre DIG, així el pròxim bucle escriurem el dígit2.
    BSF porta,2     ; Tornem el port A a 0 per tal que el display no mostri cap número.
    CLRF portd      ; Netegem el port D.
    GOTO bucle      ; Tornem al bucle.
```

```
digit2
    MOVFW DIG2  ; Posem el valor del digit2 al registre W.

    CALL taula      ; Criem la taula per saber quina combinació cal escriure pel port D, i així posar
    MOVWF portd ; el Nº desitjat al display.

    BCF porta,1     ; Escrivim el dig2 al tercer dígit del display.
    RLF DIG         ; Rotem el registre DIG, així el pròxim bucle escriurem el dígit3.
    BSF porta,1     ; Tornem el port A a 0 per tal que el display no mostri cap número.
    CLRF portd      ; Netegem el port D.
    GOTO bucle      ; Tornem al bucle.

digit3
    MOVFW DIG3  ; Posem el valor del digit3al registre W.

    CALL taula      ; Criem la taula per saber quina combinació cal escriure pel port D, i així posar
    MOVWF portd ; el Nº desitjat al display.

    BCF porta,0     ; Escrivim el dig3 al quart dígit del display.
    MOVLW 0X01  ; Posem l'1 a la posició 0 al registre DIG, doncs cal tornar a començar el cicle
    MOVWF DIG   ; escrivint el digit0 altre cop.
    BSF porta,0     ; Tornem el port A a 0 per tal que el display no mostri cap número.
    CLRF portd      ; Netegem el port D.
    GOTO bucle      ; Tornem al bucle.

; Seguidament tenim dos taules que s'encarreguen de passar el número que hi ha als registres DIG0,
; DIG1, DIG2 o DIG3 (nombre entre 0 i 9 ja que són cada un dels dígits) a la combinació binària
; corresponent per tal que a la pantalla del display apareixi el nombre que toca.

taula
    ADDWF pc,1  ; W està carregat amb el valor corresponent a un dels dígits (el que estem escrivint)
    RETLW 0XFC  ; com que realitzem una suma entre el nombre que hi ha a W (el que volem escriure)
    RETLW 0X60  ; i el punter de programa (recordem que el punter de programa assenyala la línia de
    RETLW 0XDA  ; memòria de programa que s'està duent a terme) llavors es produeix un salt de línies
    RETLW 0XF2  ; equivalent al nombre que hem sumat. Cada una de les instruccions RETLW retorna
    RETLW 0X66  ; a la crida amb un nou valor a W que correspon a la combinació binària que encén a
    RETLW 0XB6  ; pantalla del display un número determinat, que de fet és igual al  nombre de línies que
    RETLW 0XBE  ; ens hem saltat. Per exemple, si tenim que W s'ha carregat amb el valor tres la suma
    RETLW 0XE0  ; pc+3 ens port a la línia que posa RETLW 0XF2. De forma que retornem a la crida amb
    RETLW 0XFE  ; W=F2, combinació que està calculada perquè si la posem al port D (port que diu al
    RETLW 0XF6  ; display quin nombre ha d'escriure) ens aparegui el nombre 3 a la pantalla del display.

taula2          ; Aquesta altra taula té el mateix funcionament que l'anterior però serveix per al segon
    ADDWF pc,1  ; dígit del display (DIG1)  ja que les combinacions estan calculades per retornar la
    RETLW 0XFD  ; combinació que encén el nombre que toqui + el punt DP, que indica que el següent
    RETLW 0X61  ; nombre és un decimal.
    RETLW 0XDB
    RETLW 0XF3
    RETLW 0X67
    RETLW 0XB7
    RETLW 0XBF
    RETLW 0XE1
    RETLW 0XFF
    RETLW 0XF7
```

```
RSI                     ; És la rutina d'interrupció, quan es produeix una interrupció anem a parar aquí!
    MOVWF guarda1       ; Aquí tenim una seqüència que emmagatzema els valors dels registres W i
    SWAPF STATUS,0      ; STATUS per tal de poder tornar al programa quan s'acabi la interrupció sense
    MOVWF guarda2       ; que aquests registres s'hagin modificat, doncs podria ser catastròfic.

    BTFSC INTCON,0      ; Farem un salt en cas qur el bit 0 del registre INTCON sigui 0 ja que significarà
    GOTO interrupcio    ; diferent a la que ens interessa i cal tornar al programa. Si el bit és 1 llavors
                        ; anirem a tractar l'interrupció.
    SWAPF guarda2,0     ; És una rutina de fi d'interrupció, retorna els valors de l'STATUS i de W
    MOVWF STATUS        ; que teníem quan s'ha produït la interrupció per tal que no es produeixi cap
    SWAPF guarda1,1     ; catàstrofe. RETFIE torna a la línia de programa en que s'ha produït la
    SWAPF guarda1,0     ; interrupció sense modificar els registres W i STATUS (que de fet acabem
       RETFIE           ; de posar bé i no volem que variin).

interrupcio
    MOVFW portb         ; Llegim la combinació que rebem de l'encoder pel port B. I tot seguit fem una and
    MOVWF ACTUAL        ; per quedar-nos només amb els bits RB4 i 5, i que els altres bits del port B estiguin
    MOVLW 0X30          ; a 0. Doncs si tinguéssin algun valor ens podria afectar al programa.
    ANDWF ACTUAL,1

    MOVFW ANTERIOR      ; Mirem si al registre ANTERIOR (que indica el valor que tenia abans l'encoder) hi
    BTFSC STATUS,2      ; havia un 00 i llavors saltem a una subrutina que tractarà aquest cas. Per fer-ho fem
    GOTO zero           ; passar el  registre ANTERIOR al W, si el bit 2 de l'STATUS és 1 llavors teníem un 00.

    MOVLW 0X10          ; Mirem si al registre ANTERIOR (que indica el valor que tenia abans l'encoder) hi
    SUBWF ANTERIOR,0    ; havia un 0x10 i llavors saltem a una subrutina que tractarà aquest cas. Per fer-ho restem
    BTFSC STATUS,2      ; 0x10 al registre ANTERIOR, si ens el resultat és 0, el bit 2 de l'STATUS es posa a 1 i
    GOTO un             ; vol dir que efectivament l'ANTERIOR era 0x10.


    MOVLW 0X20          ; Mirem si al registre ANTERIOR (que indica el valor que tenia abans l'encoder) hi
    SUBWF ANTERIOR,0    ; havia un 0x20 i llavors saltem a una subrutina que tractarà aquest cas. Per fer-ho restem
    BTFSC STATUS,2      ; 0x20 al registre ANTERIOR, si ens el resultat és 0, el bit 2 de l'STATUS es posa a 1 i
    GOTO dos            ; vol dir que efectivament l'ANTERIOR era 0x20.


    MOVLW 0X30          ; Mirem si al registre ANTERIOR (que indica el valor que tenia abans l'encoder) hi
    SUBWF ANTERIOR,0    ; havia un 0x30 i llavors saltem a una subrutina que tractarà aquest cas. Per fer-ho restem
    BTFSC STATUS,2      ; 0x30 al registre ANTERIOR, si ens el resultat és 0, el bit 2 de l'STATUS es posa a 1 i
    GOTO tres           ; vol dir que efectivament l'ANTERIOR era 0x30.

zero
   MOVLW 0X10       ; Sabem que el registre ANTERIOR era 0x00 i ara mirem pel mateix sistema que abans si
   SUBWF ACTUAL,0   ; el registre ACTUAL és 0x10, en aquest cas caldrà incrementar, farem un salt a una subrutina.
   BTFSC STATUS,2   ; Sinó mirarem si l'ACTUAL és 0x20 i si es dona el cas decrementarem, farem un salt a una
   GOTO incrementar ;  altra subrutina. Si no es dona cap dels dos casos significa que hi ha hagut un error, doncs
   MOVLW 0X20       ; l'encoder només pot donar una d'aquestes dues combinacions quan anteriorment tenia un
   SUBWF ACTUAL,0   ;  0x00. Caldrà entendre que hi ha hagut un error i saltarem a una subrutina que tracta aquest
   BTFSC STATUS,2   ; fallo.
   GOTO decrementar
   GOTO fallo

un
   MOVLW 0X30       ; Sabem que el registre ANTERIOR era 0x10 i ara mirem pel mateix sistema que abans si
   SUBWF ACTUAL,0   ; el registre ACTUAL és 0x30, en aquest cas caldrà incrementar, farem un salt a una subrutina.
   BTFSC STATUS,2   ; Sinó mirarem si l'ACTUAL és 0x00 i si es dona el cas decrementarem, farem un salt a una
   GOTO incrementar ;  altra subrutina. Si no es dona cap dels dos casos significa que hi ha hagut un error, doncs
   MOVFW ACTUAL     ; l'encoder només pot donar una d'aquestes dues combinacions quan anteriorment tenia un
   BTFSC STATUS,2   ;  0x00. Caldrà entendre que hi ha hagut un error i saltarem a una subrutina que tracta aquest
   GOTO decrementar ; fallo.
   GOTO fallo
```

```
dos
    MOVFW ACTUAL  ; Sabem que el registre ANTERIOR era 0x20 i ara mirem pel mateix sistema que abans si
    BTFSC STATUS,2   ; el registre ACTUAL és 0x00, en aquest cas caldrà incrementar, farem un salt a una subrutina.
    GOTO incrementar  ; Sinó mirarem si l'ACTUAL és 0x30 i si es dona el cas decrementarem, farem un salt a una
    MOVLW 0X30       ;  altra subrutina. Si no es dona cap dels dos casos significa que hi ha hagut un error, doncs
    SUBWF ACTUAL,0; l'encoder només pot donar una d'aquestes dues combinacions quan anteriorment tenia un
    BTFSC STATUS,2   ;  0x00. Caldrà entendre que hi ha hagut un error i saltarem a una subrutina que tracta aquest
    GOTO decrementar  ; fallo.
    GOTO fallo

tres
    MOVLW 0X20       ; Sabem que el registre ANTERIOR era 0x30 i ara mirem pel mateix sistema que abans si
    SUBWF ACTUAL,0; el registre ACTUAL és 0x20, en aquest cas caldrà incrementar, farem un salt a una subrutina.
    BTFSC STATUS,2   ; Sinó mirarem si l'ACTUAL és 0x10 i si es dona el cas decrementarem, farem un salt a una
    GOTO incrementar  ;  altra subrutina. Si no es dona cap dels dos casos significa que hi ha hagut un error, doncs
    MOVLW 0X10       ; l'encoder només pot donar una d'aquestes dues combinacions quan anteriorment tenia un
    SUBWF ACTUAL,0;  0x00. Caldrà entendre que hi ha hagut un error i saltarem a una subrutina que tracta aquest
    BTFSC STATUS,2   ; fallo.
    GOTO decrementar
    GOTO fallo

fallo
    MOVFW portb          ; Si per alguna raó es produeix una seqüència que l'encoder no ha de seguir per a cap de
    MOVWF ANTERIOR       ; les direccions de gir (esquerre i dreta) vol dir que per algun motiu s'ha produït un error.
    MOVLW 0X30           ; Llavors llegim l'encoder per actualitzar el valor un altre cop i no fem res, ni incrementem
    ANDWF ANTERIOR,1     ;  ni decrementem, només fem un salt a la subrutina de final d'interrupció
    GOTO fiint

incrementar
    MOVFW DIG0       ; El primer que fem per incrementar serà comprovar si estem a l'ultim nombre possible (110.0).
    BTFSS STATUS,2   ; Per fer-ho mirem si digit0 és un 0, fem passar el digit 0 pel registre W i llavors fem un salt si
    GOTO incd0       ; 1 en cas que el bit 2 de l'STATUS sigui 1. Si aquest bit de l'STATUS és 0, no farem salt i
    MOVFW DIG1       ; per tant passarem a la següent instrucció que és un salt a la subrutina incd0. Si el bit fos un
    BTFSS STATUS,2   ; 1 llavors saltariem la instrucció GOTO i passaríem a mirar si el segon digit també resulta ser
    GOTO incd0       ; un 0 pel mateix sistema. I també ho fem per al tercer dígit.
    MOVLW 0X01
    SUBWF DIG2,0
    BTFSS STATUS,2
    GOTO incd0
    MOVLW 0X01       ; Per mirar si el primer dígit és un 1 fem la resta del digit meny 0x01, si el resultat dóna zero el
    SUBWF DIG3,0     ; bit 2 de l'STATUS serà 1 i voldrà dir que estem al nombre màxim, sino el bit serà zero i igual
    BTFSS STATUS,2   ; que abans anirem a la subrutina incrementar2.
    GOTO incd0
    CLRF DIG0        ; Si arribem a aquesta instrucció significa que hem saltat la subrutina incd0 i que ens
    MOVLW 0X05       ; trobem que estem al nombre màxim, en lloc d'incrementar el que farem és anar al nombre
    MOVWF DIG1       ; mínim, per tant inicialitzem els digits i el divisor N per tal que ens situem a la freqüència
    MOVLW 0X08       ; mínima, 085.0MHz.
    MOVWF DIG2
    CLRF DIG3
    MOVLW 0X52
    MOVWF N1
    MOVLW 0X03
    MOVWF N2
    CALL escriureN
    GOTO fiint

incd0
    MOVLW 0X09       ; Si hem saltat a aquesta subrutina significa que hem d'incrementar, mirem si cal incrementar
    SUBWF DIG0,0     ; el digit0, per fer-ho cal saber si aquest és un 9. Ho fem com sempre, fem la resta de DIG0 i
    BTFSC STATUS,2   ; 0x09, si el bit dos de l'STATUS es posa a 1, vol dir que el resultat ha donat  0 i per tant cal
    GOTO incd1       ; que mirem el digit1, anirem a una subrutina incd1 (incrementar digit 1). Si pel contrari el bit
    INCF DIG0        ; 2 de l'STATUS és 0, saltem la instrucció GOTO i incrementem en 1 el digit0, doncs no és un
    CALL incN        ; 9 per tant el podem incrementar. També cal incrementar el divisor N, farem una crida amb un
    CALL escriureN   ; CALL que durà a terme aquesta tasca i una altra per passar la nova informació del divisor al
    GOTO fiint       ; PLL. . Llavors anirem a la subrutina de final d'interrupció.
```

```
incd1
    MOVLW 0X09      ; Si estem en aquesta subrutina vol dir que el primer digit és un 9 i com que hem d'incrementar
    SUBWF DIG1,0    ; el segon dígit es veurà afectat. Caldrà que observem, igual que hem fet amb el primer dígit, si
    BTFSC STATUS,2  ; el segon també és un 9, doncs llavors caldria modificar el tercer dígit. Si no és nou, voldra dir
    GOTO incd2      ; que hem d'incrementar el segon digit i posar el primer a 0, també caldrà que incrementem N
    INCF DIG1       ; i que passem la informació nova al PLL  (amb les mateixa crides, CALL, que abans). Finalment
    CLRF DIG0       ; anirem a la subrutina de final d'interrupció.
    CALL incN
    CALL escriureN
    GOTO fiint

incd2
    MOVLW 0X09      ; Si estem en aquesta subrutina vol dir que el primer  i el segón digits són 9 i com que hem
    SUBWF DIG2,0    ; d'incrementar el tercer dígit es veurà afectat. Caldrà que observem, igual que hem fet amb el
    BTFSC STATUS,2  ; primer i el segon dígit, si el tercer també és un 9, doncs llavors caldria modificar el quart dígit
    GOTO incd3      ; (en aquest cas estaríem al número 99.9). Si no és nou, voldra dir.que hem d'incrementar el
    INCF DIG2       ; tercer digit i posar el primer i el segón a 0, també caldrà que incrementem N i que passem la
    CLRF DIG0       ; informació nova al PLL  (amb les mateixa crides, CALL, que abans). Finalment anirem a la
    CLRF DIG1       ; subrutina de final d'interrupció.
    CALL incN
    CALL escriureN
    GOTO fiint

incd3
    INCF DIG3       ; Si estem en aquesta subrutina vol dir que el primer, el segón i el tercer digits són 9 (estem al
    CLRF DIG0       ; nombre 99.9) per tant cal que escrivim el 100.0, posem els tres primers dígits a 0 i incrementem
    CLRF DIG        ; el 4t. També caldrà que incrementem N i que passem la informació nova al PLL  (amb les
    CLRF DIG2       ; mateixa crides, CALL, que abans). Finalment anirem a la subrutina de final d'interrupció.
    CALL incN
    CALL escriureN
    GOTO fiint

decrementar
    MOVFW DIG0      ; El primer que fem per decrementar serà comprovar si estem al primer nombre possible (85.0).
    BTFSS STATUS,2  ; Per fer-ho mirem si digit0 és un 0, fem passar el digit 0 pel registre W i llavors fem un salt si
    GOTO decd0      ; 1 en cas que el bit 2 de l'STATUS sigui 1. Si aquest bit de l'STATUS és 0, no farem salt i
    MOVLW 0X05      ; per tant passarem a la següent instrucció que és un salt a la subrutina incd0. Si el bit fos un
    SUBWF DIG1,0    ; 1 llavors ens saltariem la instrucció GOTO i passaríem a mirar si el segon digit resulta ser un
    BTFSS STATUS,2  ; 5. Per fer-ho farem una resta entre 0x05 i el digit1, si la resta dóna 0 vol dir que el segon digit
    GOTO decd0      ; és un 5 i això ens ho indicaria el bit 2 de l'STATUS que es posaria a 1. En aquest cas miraríem
    MOVLW 0X08      ; si el següent dígit és un 8 pel mateix sistema, si ho és voldra dir que efectivamet estem al mínim
    SUBWF DIG2,0    ; valor possible de freqüència.
    BTFSS STATUS,2
    GOTO decd0
    MOVFW DIG0
    BTFSS STATUS,2
    GOTO decd0
    CLRF DIG0       ; Si arribem a aquesta instrucció significa que hem saltat la subrutina decd0 i que ens
    CLRF DIG1       ; trobem que estem al nombre mínim, en lloc de decrementar el que farem és anar al nombre
    MOVLW 0X01      ; màxim, per tant inicialitzem els digits i el divisor N per tal que ens situem a la freqüència
    MOVWF DIG2      ; mínima, 110.0MHz.
    MOVWF DIG3
    MOVLW 0X4C
    MOVWF N1
    MOVLW 0X04
    MOVWF N2
    CALL escriureN
    GOTO fiint
```

```
decd0
    MOVFW DIG0          ; Si hem saltat a aquesta subrutina significa que hem de decrementar, mirem si cal decrementar
    BTFSC STATUS,2      ; el digit0, per fer-ho cal saber si aquest és un 0. Ho fem com sempre, passem el valor pel
    GOTO decd1          ; registre W, si el bit dos de l'STATUS es posa a 1, vol dir que el resultat ha donat  0 i per tant
    DECF DIG0           ; cal que mirem el digit1, anirem a una subrutina decd1 (decrementar digit 1). Si pel contrari el bit
    CALL decN           ; 2 de l'STATUS és 0, saltem la instrucció GOTO i decrementem en 1 el digit0, doncs no és un
    CALL escriureN      ; per tant el podem decrementar. També cal decrementar el divisor N, farem una crida amb un
    GOTO fiint          ; CALL que durà a terme aquesta tasca i una altra per passar la nova informació del divisor al
                        ; PLL. . Llavors anirem a la subrutina de final d'interrupció.

decd1
    MOVFW DIG1          ; Si estem en aquesta subrutina vol dir que el primer digit és un 0 i com que hem de decrementar
    BTFSC STATUS,2      ; el segon dígit es veurà afectat. Caldrà que observem, igual que hem fet amb el primer dígit, si
    GOTO decd2          ; el segon també és un 0, doncs llavors caldria modificar el tercer dígit. Si no és zero, voldra dir
    DECF DIG1           ; que hem de decrementar el segon digit i posar el primer a 9, també caldrà que decrementem N
    MOVLW 0X09          ; i que passem la informació nova al PLL  (amb les mateixa crides, CALL, que abans). Finalment
    MOVWF DIG0          ; anirem a la subrutina de final d'interrupció.
    CALL decN
    CALL escriureN
    GOTO fiint

decd2
    MOVFW DIG2          ; Si estem en aquesta subrutina vol dir que el primer  i el segón digits són 0 i com que hem
    BTFSC STATUS,2      ; de decrementar el tercer dígit es veurà afectat. Caldrà que observem, igual que hem fet amb el
    GOTO decd3          ; primer i el segon dígit, si el tercer també és un 0, doncs llavors caldria modificar el quart dígit
    DECF DIG2           ; (en aquest cas estaríem al número 100.0). Si no és 0, voldra dir.que hem de decrementar el
    MOVLW 0X09          ; tercer digit i posar el primer i el segón a 9, també caldrà que decrementem N i que passem la
    MOVWF DIG0          ; informació nova al PLL  (amb les mateixa crides, CALL, que abans). Finalment anirem a la
    MOVWF DIG1          ; subrutina de final d'interrupció.
    CALL decN
    CALL escriureN
    GOTO fiint

    INCF DIG3       ; Si estem en aquesta subrutina vol dir que el primer, el segón i el tercer digits són 9 (estem al
    CLRF DIG0       ; nombre 99.9) per tant cal que escrivim el 100.0, posem els tres primers dígits a 0 i incrementem
    CLRF DIG        ; el 4t. També caldrà que incrementem N i que passem la informació nova al PLL  (amb les
    CLRF DIG2       ; mateixa crides, CALL, que abans). Finalment anirem a la subrutina de final d'interrupció.
    CALL incN
    CALL escriureN
    GOTO fiint

decd3
    DECF DIG3       ; Si estem en aquesta subrutina vol dir que el primer, el segón i el tercer digits són 0 (estem al
    MOVLW 0X09      ; nombre 100.0) per tant cal que escrivim el 99.9, posem els tres primers dígits a 9 i decrementem
    MOVWF DIG0      ; el 4t. També caldrà que decrementem N i que passem la informació nova al PLL  (amb les
    MOVWF DIG1      ; mateixa crides, CALL, que abans). Finalment anirem a la subrutina de final d'interrupció.
    MOVWF DIG2
    CALL decN
    CALL escriureN
    GOTO fiint

fiint
    MOVLW 0X01      ; Aquesta és la rutina de final d'interrupció. Com que si estem aquí possiblement vol dir qur hem
    MOVWF DIG       ; hagut de modificar el nombre del display i del divisor el que primer fem és posar DIG a 0x01, així
    BCF INTCON,0    ; tonarà a començar el bucle escrivint el primer dígit. Llavors posem el bit 0 del registre INTCON
    BSF INTCON,3    ; a 0, doncs és el bit que indica que hi ha hagut una interrupció de canvi en el port B i cal
                    ; inicialitzar-lo manulament. També hem de posar el bit 3 del d'aquest registre a 1, doncs habilita
                    ; aquesta mateixa interrupció.

    SWAPF guarda2,0 ; És una rutina de fi d'interrupció, retorna els valors de l'STATUS i de W
    MOVWF STATUS    ; que teníem quan s'ha produït la interrupció per tal que no es produeixi cap
    SWAPF guarda1,1 ; catàstrofe. RETFIE torna a la línia de programa en que s'ha produït la
    SWAPF guarda1,0 ; interrupció sense modificar els registres W i STATUS (que de fet acabem
      RETFIE        ; de posar bé i no volem que variin).
```

```
escriureN
    MOVFW N2     ; Aquesta és una subrutina de crida, el que fa és passar el valor del divisor del PLL a aquest
    MOVWF N22    ; perquè segurament l'hem hagut de variar (ja sigui incrementant-lo o decrementant-lo). Primer
    MOVLW 0X03   ; posem el valor de N2 en un altre registre N22, doncs l'algorisme que fem servir per passar la
    MOVWF BIT    ; informació al PLL destruirà el valor guardat a N2 i necessitem tenir-lo emmagatzemat. Llavors
    RLF N22,1    ; inicialitzem el registre BIT a 0x03, doncs ens servirà de contador per saber quants cops hem
    RLF N22,1    ; rotat. Això és degut a que l'algorisme esmentat dóna la informació al PLL per un bit de forma
    RLF N22,1    ; sèrie per la qual cosa el que fem és rotar per anar obtenint els valors del registre un a un en
    RLF N22,1    ; sèrie. Com que del registre N2 només ens interessen els 3 últims bits farem rotar N22 cinc cops.
    RLF N22,1

    seguimN2
        MOVFW BIT        ; Passem el contador (registre BIT) al registre W per mirar si és un 0, si es dóna el cas
        BTFSC STATUS,2   ; anirem a la rutina escriure N1 ja que significarà que ja hem acabat de passar tota la
        GOTO escriureN1  ; informació de N22. Sinó seguirem rotant N22 per passar la informació al PLL.

         rotarN2
        DECF BIT,1       ; Decrementem el contador BIT, llavors rotem N22, si el bit obtingut és un 0, llavors
        RLF N22,1        ; el bit 0 de l'STATUS (bit de carry) serà 0 per tant anirem a la subrutina N2 és un 0,
        BTFSS STATUS,0   ; que li passarà un 0 al PLL. Si pel contrari N2 és un 1, li passarem un 1 al PLL. Per
        GOTO N2esun0     ; fer-ho el que fem és posar un 1 al bit 1 del port C (bit de data del PLL) i llavors creem
        BSF portc,1      ; un pols de rellotge del PLL (bit 0 del port C) per tal que es produeixi un flanc de
        BSF portc,0      ; pujada al rellotge del PLL i aquest llegeixi aquest 1 que li hem posat a la pota de
        BCF portc,0      ; data (on el PLL rep els bits del divisor). Finalment tornem al bucle de rotació de N22.
        GOTO seguimN2

    N2esun0
        BCF portc,1      ; En aquest subrutina tenim que cal enviar un 0 al divisor del PLL per tant posem la pota
        BSF portc,0      ; de data del PLL (bit 1 del port C) a 0 i llavors fem un pols de rellotge (bit 0 del port C).
        BCF portc,0      ; Quan hem acabat aquesta seqüència que dóna un 0 al PLL, seguirem el bucle de roatció
        GOTO seguimN2    ; de N22.

    escriureN1           ; Si ja hem arribat a aquesta subrutina significa que hem acabat d'enviar N2 i ara cal enviar
        MOVFW N1         ; N1. Igual que abans passem les dades de N1 a un altre registre (N11) per evitar perdre la
        MOVWF N11        ; informacióde N1 al fer l'algorisme de rotació. A més aquest cop inicialitzem el contador
        MOVLW 0X08       ; BIT a 8, doncs del registre N11 cal aprofitar tots es bits (és un registre de 8 bits).
        MOVWF BIT

    seguimN1             ; Aquest és l'algorisme de rotació de N1 que permet extreure'n els bits un a un per
        MOVFW BIT        ; enviar-los en sèrie al PLL. Primer mirem si el contador, BIT, ja és 0 (s'hauria acabat)
        BTFSC STATUS,2   ; d'enviar N1 i passaríem a enviar el bit anomenat C del PLL mitjançant la subrutina
        GOTO ultimsbits  ;últims bits.

    rotarN1
        DECF BIT,1       ; Si estem en aquesta subrutina cal que seguim enviant N11, igual que per a N22 el
        RLF N11,1        ; que primer fem és decremenatr el contador BIT. Després rotem N11 per obtenir els
        BTFSS STATUS,0   ; bits que el componen un a un. Si és un 0 (ho sabem a aprtir del bit de carry del
        GOTO N1esun0     ; registre STATUS) anem a la subrutina és un 0. Sinó enviem un 1 al la pota de data
        BSF portc,1      ; del PLL (bit 1 del port C) i llavors fem un pols de rellotge del PLL (bit 0 del port C)
        BSF portc,0      ; posem la el bit corresponent a 1 i després a 0, així creem un flanc de pujada. Després
        BCF portc,0      ; amb la instrucció GOTO continuem el bucle de rotació de N11.
        GOTO seguimN1

    N1esun0              ; En aquesta subrutina cal enviar un 0 a la pota data del PLL i després crear un pols de
        BCF portc,1      ; rellotge per produir un flanc de pujada i que el PLL adquireixi la dada. Seguidament
        BSF portc,0      ; continuarem el bucle de rotar el registre N11.
        BCF portc,0
        GOTO seguimN1
```

```
ultimsbits          ; Aquesta subrutina envia el bit C i els 7 bits que cal enviar al PLL que de fet no contenen
    MOVLW 0X08      ; informació (podem enviar  el qe volguem) i els enviarem iguals que el bit C, és a dir com
    MOVWF BIT       ; a zeros.Conseqüentment el que hem de fer es traduеix a enviar 8 zeros a la pota data del
    BCF portc,1     ; PLL per tant posem aquesta pota (bit 1 del port C) a zero i llavors creem 8 flancs al rellotge
    BSF portc,0     ; del PLL (bit 0 del port C) és a dir posem aquest bit del port C a 1 i després a zero 0 vuit
    BCF portc,0     ; vegades.
    BSF portc,0
    BCF portc,0
    BSF portc,0
    BCF portc,0
    BSF portc,0
    BCF portc,0
    BSF portc,0
    BCF portc,0
    BSF portc,0
    BCF portc,0
    BSF portc,0
    BCF portc,0
    BSF portc,0
    BCF portc,0

    BSF portc,2     ; Finalment, un cop el lach del PLL ja té tots els bits que necessita perquè pugui saber
    BCF portc,2     ; que li hem donat un nou divisor N, cal que carreguem el lach mitjançant la pota LE del
        RETURN      ; PLL (bit 2 del port C), com que aquesta pota funciona per flanc de pujada li posem un 1
                    ; i després un 0 per crear un pols en aquesta pota. Finalment tornem de la crida mitjançant
                    ; la instrucció RETURN.


incN                ; Aquesta crida s'encarrega d'incrementar el divisor N del PLL. Per fer-ho comencem per
   INCF N1,1        ; incrementar N1. Com que el divisor N del PLL esta dividit en 2 registres N1 i N2, ja que
   BTFSC STATUS,2   ; el nombre del divisor és molt gran i ocupa més d'un registre no n'hi ha prou en
   GOTO incN2       ; incrementar N1 sinó que un cop ho hem fet cal mirar si N1 ha passat a valdre 0, en aquest
       RETURN       ; cas el bit 2 del registre STATUS valdrà 1 per la qual cosa la instrucció BTFSC (salt si 0)
                    ; no es saltarà la instrucció següent que al ser un GOTO ens enviarà a una subrutina que
                    ; s'encarrega d'incrementar N2 (doncs si N1 era tot uns i ha passat a zero cal incrementar N2).
                    ;  Sino com que ja haurem incrementat N1 i no caldrà que augmentar N2 podrem tornar de
                    ; la crida mitjançant la istrucció RETURN inmediatament.

    incN2           ; En aquesta subrutina senzillament cal incrementar N2, ho fem amb la instrucció INCF i
       INCF N2,1    ; llavors retornem de la crida amb la RETURN.
           RETURN


decN                ; Aquesta crida s'encarrega de decrementar el divisor N del PLL. Per fer-ho comencem per
   DECF N1,1        ; decrementar N1. Com que el divisor N del PLL esta dividit en 2 registres N1 i N2, ja que
   MOVLW 0XFF       ; el nombre del divisor és molt gran i ocupa més d'un registre no n'hi ha prou en
   SUBWF N1,0       ; decrementar N1 sinó que un cop ho hem fet cal mirar si N1 ha passat a valdre 0xFF, per
   BTFSC STATUS,2   ; fer-ho restem 0xFF de N1 (un cop ja l'hem decrementat), si el resultat és 0 el bit 2 del registre
   GOTO decN2       ; STATUS valdrà 1 per la qual cosa la instrucció BTFSC (salt si 0) noes saltarà la instrucció
       RETURN       ; següent que al ser un GOTO ens enviarà a una subrutina que s'encarrega d'incrementar N2
                    ; (doncs si N1 era zero i ha passat a ser tot uns cal decrementar N2). Sino com que ja haurem
                    ; decrementat N1 i no caldrà que disminuïm N2 podrem tornar de la crida mitjançant la
                    ; istrucció RETURN inmediatament.

    decN2           ; En aquesta subrutina senzillament cal decrementar N2, ho fem amb la instrucció DECF i
       DECF N2,1    ; llavors retornem de la crida amb la RETURN.
           RETURN
END
```