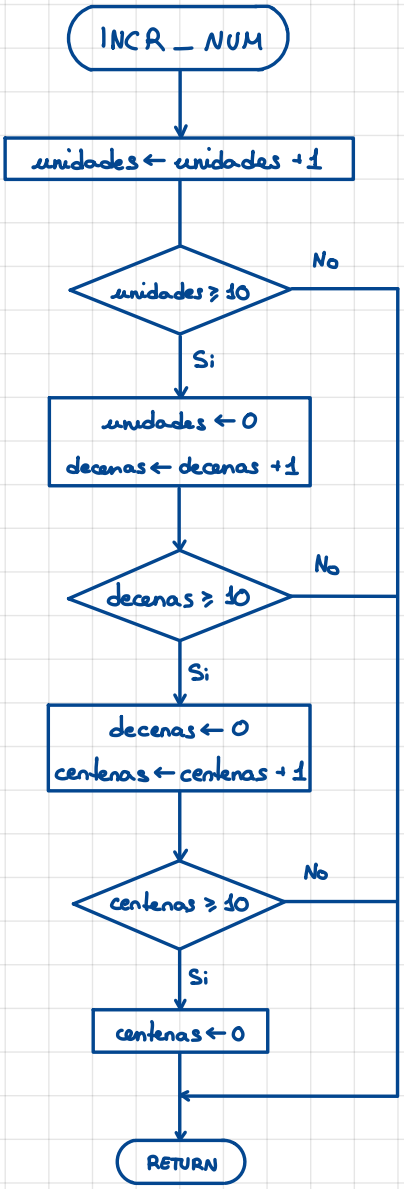
Ejercicios Desarrollo de Software

1 Realizar el diagrama de flujo de una subrutina que, cada vez que se ejecuta, incrementa en uno un número almacenado en memoria en formato BCD.* Se emplean tres bytes en memoria para representar dicho número en formato BCD, un primer byte para representar las unidades, un segundo byte para representar decenas y un tercer byte para representar centenas.




```
INCR_NUM:
    ;Se incrementan las unidades
    INCR unidades
    ;Se guarda el valor '10' en el ACUMULADOR (registro A)
    MOV A, #0Ah
    ;limpiar el CARRY** (registro C)
    CLR C
    ;Se compara el valor de las unidades con el acc. Si son iguales,
    se ejecuta la siguiente línea. Si son diferentes, salta a FIN
    CJNE A, unidades, FIN
    ;unidades ≠ 10. Comprobar CARRY para ver si 'unidades' es
    ;'> 10' o '< 10'. Si 'unidades > 10', 'C=1' y se ejecuta la
    ;siguiente línea. Si 'unidades < 10', 'C=0' y salta a FIN.
    JNC FIN
    ;unidades = 0
    MOV unidades, #00h
    ;Se incrementan las decenas
    INR decenas
        .
        .
        .
    MOV centenas, #00h

FIN:
    RET
```

REPETIR CON 'decenas' Y 'centenas'

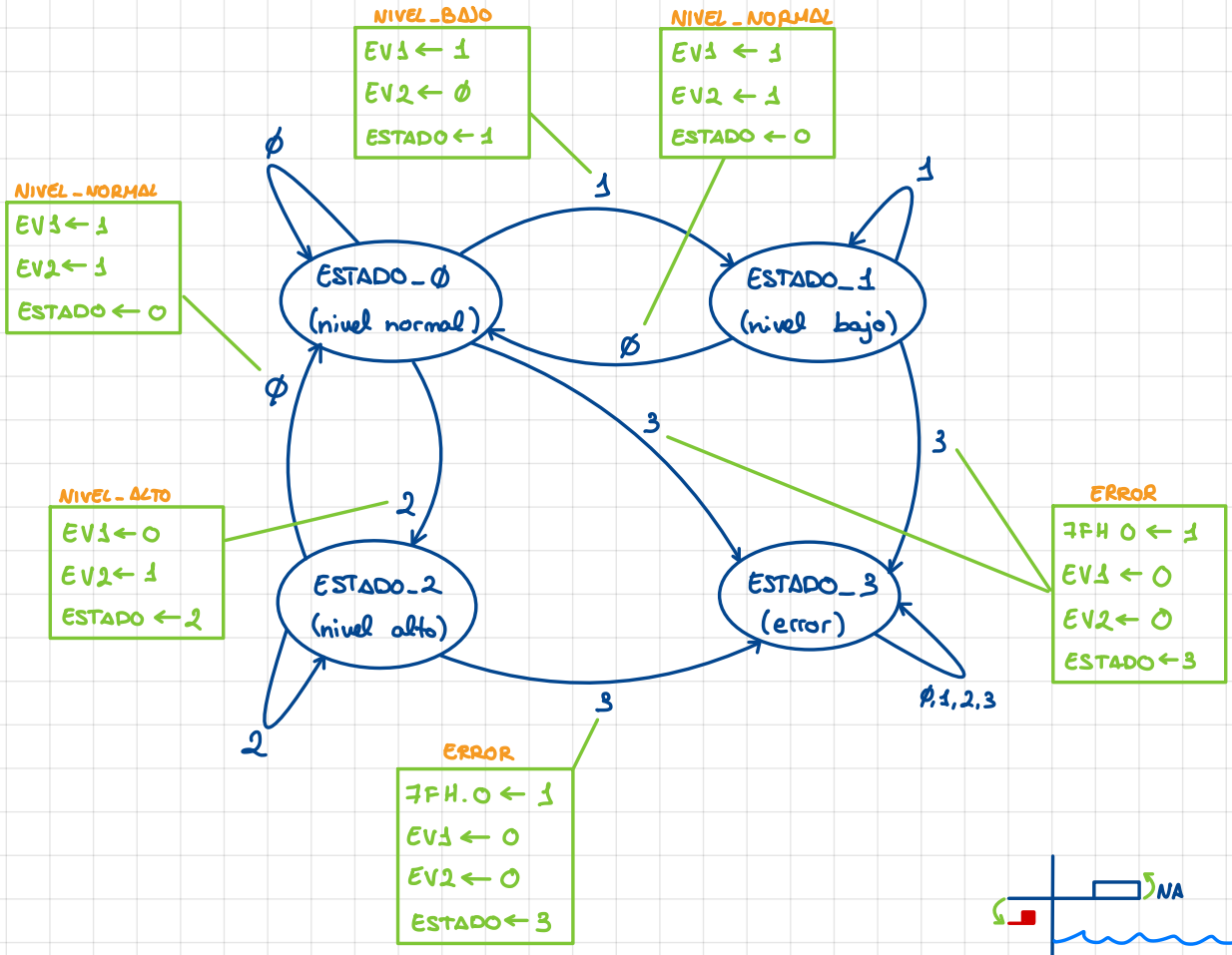
* BCD: 'Binary-Coded Decimal'. Cada cifra decimal se codifica en binario por separado. Por ejemplo, 45 en binario es 101101, pero en BCD es 0100 0101 (4 y 5)

** CARRY: Es un registro que usa el micro en operaciones aritméticas como restas, para indicar que el resultado es negativo. Por ejemplo, en la resta '5-4', el CARRY tendría el valor 0 porque el resultado es positivo. Pero al hacer '4-5' el CARRY cambiaría a 1.

2 Realizar el diagrama de estados, eventos y acciones del control de nivel de fluido de un tanque de almacenamiento de agua. Las especificaciones del programa requerido para el control del nivel son las siguientes:

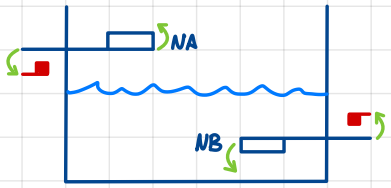
1) Se efectuará la regulación del nivel por medio de dos detectores de nivel (NB: nivel bajo, NA: nivel alto) del depósito y dos electroválbulas, una de admisión de agua (EV1) y otra de distribución de agua (EV2). Los detectores de nivel NB y NA están asociados a las entradas P1.0 y P1.1, respectivamente. Las activaciones de las electroválbulas EV1 y EV2 están asociadas a las salidas P1.4 y P1.5, respectivamente. Las electroválbulas se activan poniendo los puertos de salida a 1 y se desactivan poniendo dichos puntos a 0.

2) El algoritmo de control debe desempeñar las siguientes acciones:
   a) Cuando el nivel es bajo (NB), activar EV1 y desactivar EV2.
   b) Cuando el nivel es alto (NA), desactivar EV1 y activar EV2.
   c) Cuando no hay nivel bajo ni nivel alto activar EV1 y EV2.
   d) Estudiar casos de error y detectar los mismos, poniendo el bit 0 de la dirección de memoria 7FH a 1 en caso de error.

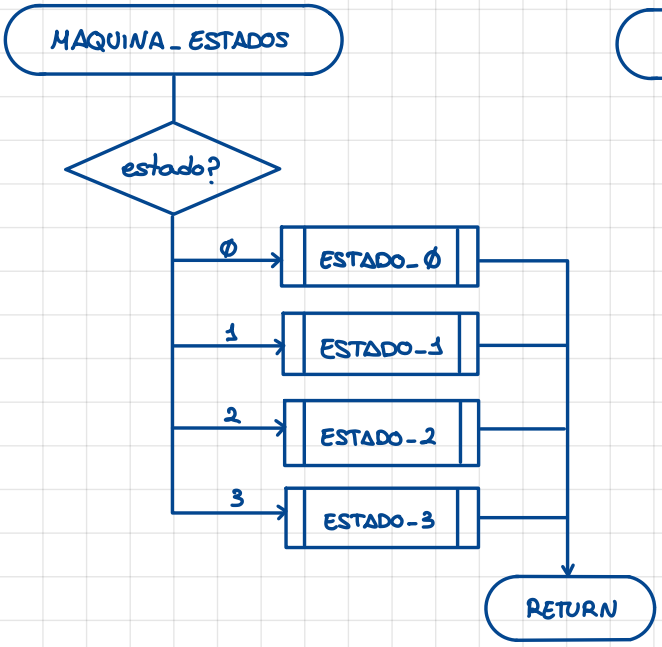


NA	NB	ESTADOS
0	0	0: Nivel normal
0	1	1: Nivel bajo
1	0	2: Nivel alto
1	1	3: Error

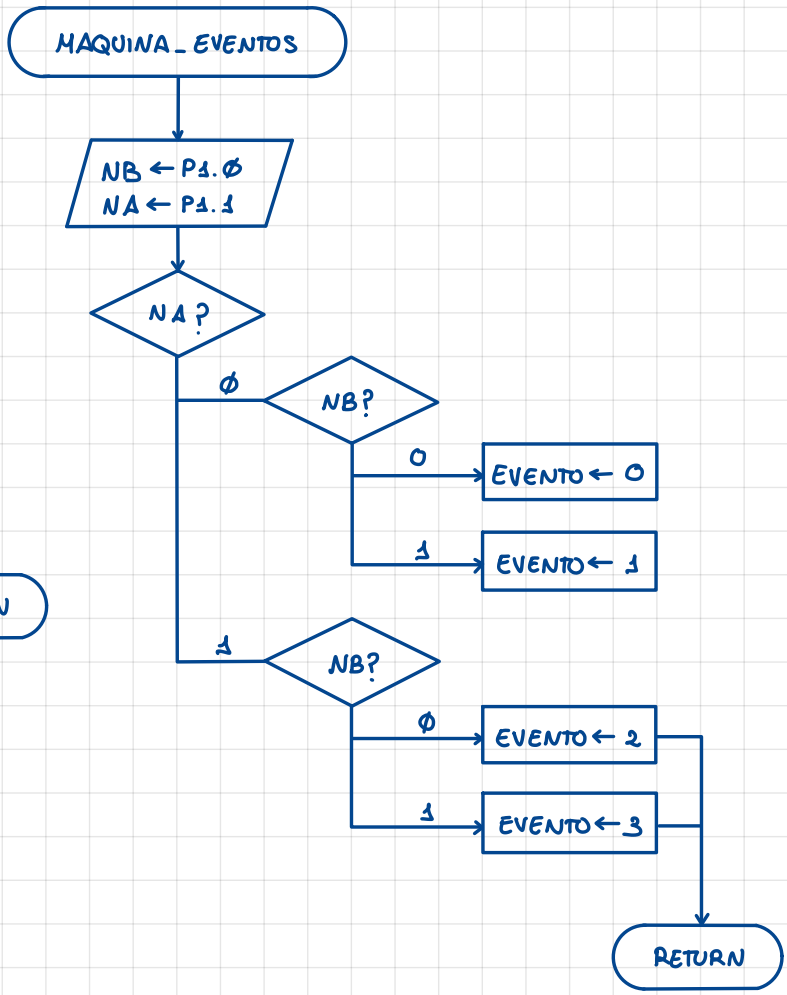
EVENTOS *	
0:	$\overline{NA}$ y $\overline{NB}$
1:	$\overline{NA}$ y NB
2:	NA y $\overline{NB}$
3:	NA y NB



* NB se activa cuando el nivel es bajo, y NA cuando es alto. Si es normal no se activa ninguno, y si se activan los dos a la vez hay un error.

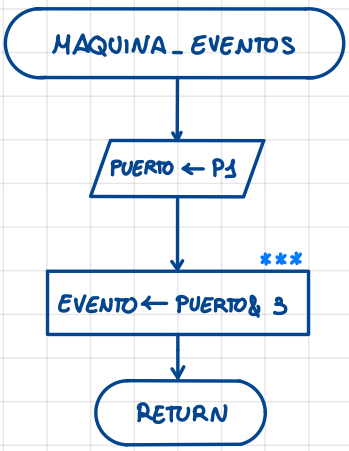


(OPCIÓN 1) **

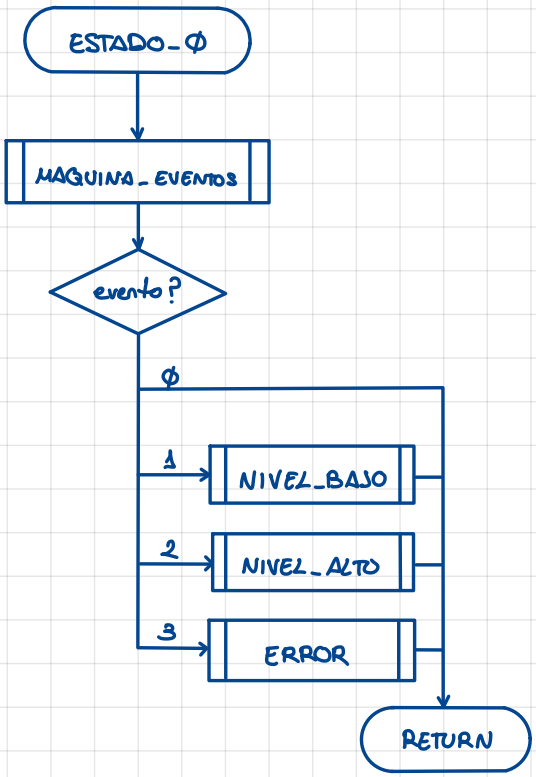


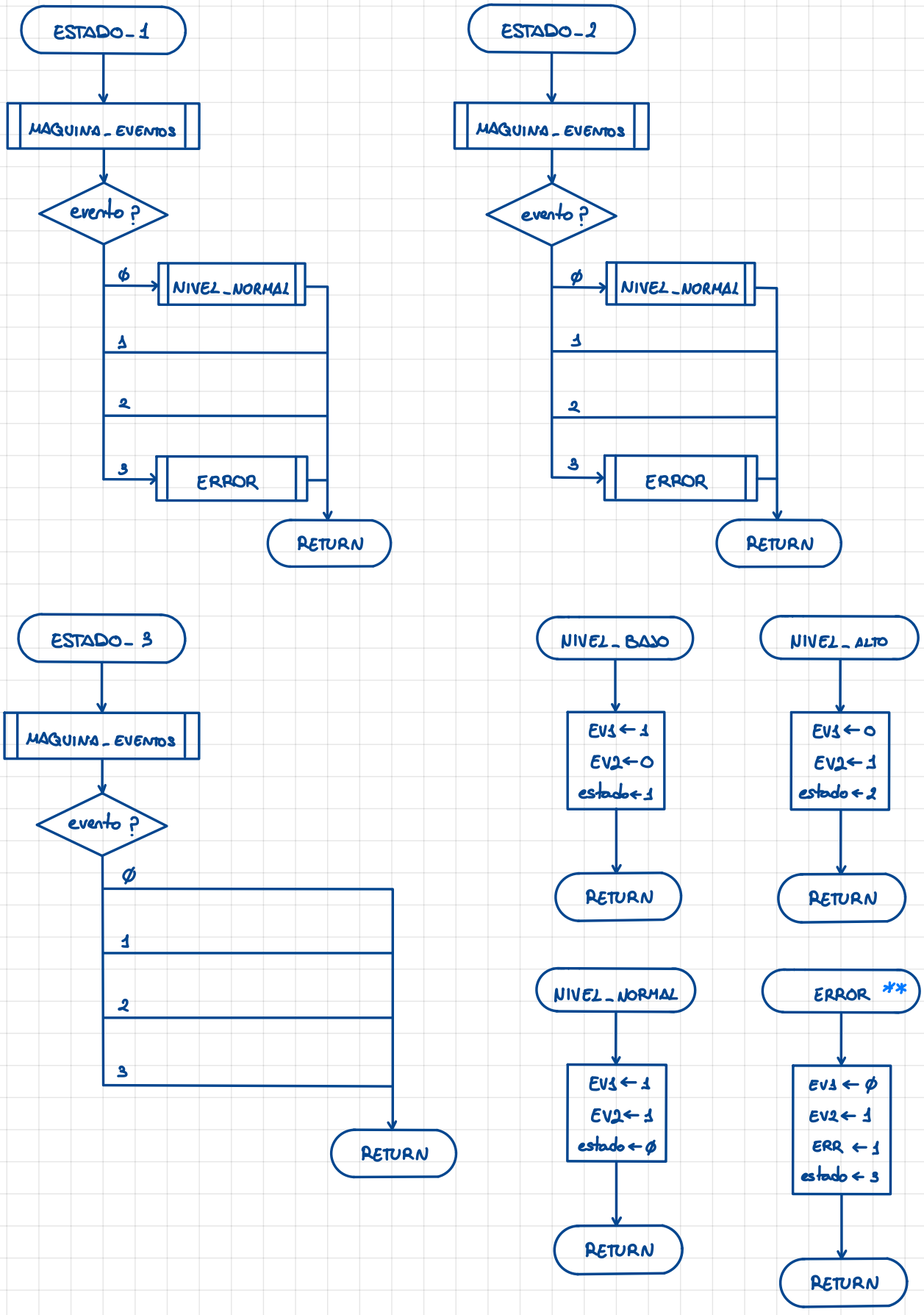
** Las dos opciones son posibles, pero la 2 es mejor.

(OPCION 2) **



*** & XXXXXXXX : puerto
00000011 : mascara
00000XX
P1.1 P1.0





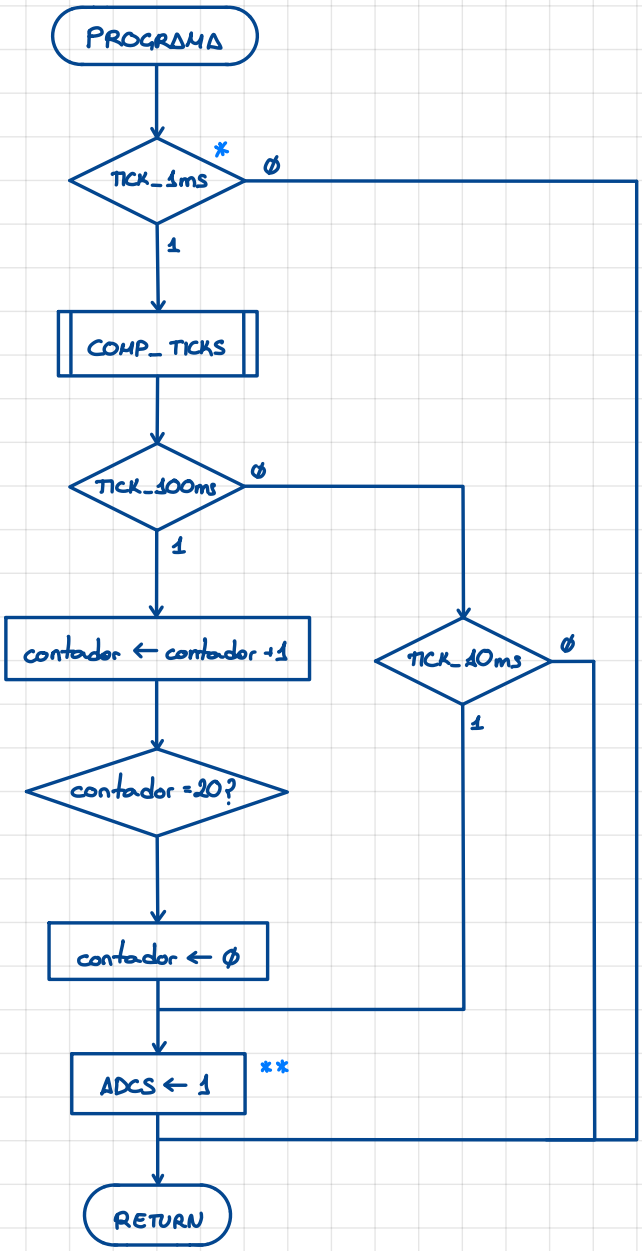
** No se especifica qué hacer con las electroválvulas en caso de error, pero por seguridad he decidido abrir la válvula de salida de agua. Hay más opciones posibles.

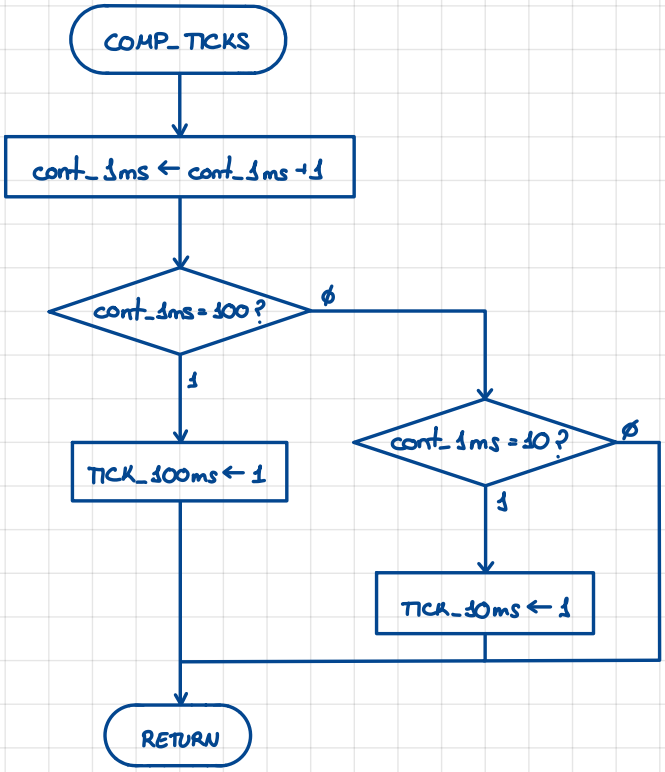
```
; ETIQUETAS
; variables
estado    EQU   R7
evento    EQU   R6
nivel     EQU   R5
ERR       EQU   7FH.0
; puertos de entrada
puerto    EQU   P1
; puertos de salida
EV1       EQU   P3.4
EV2       EQU   P3.5

ORG  0X00H
   AJMP  INICIO

ORG  0X30H
   INICIALIZAR:
      MOV   estado , #00H
      MOV   evento , #00H
      MOV   nivel , #00H
      CLR   ERR
      CLR   EV1
      CLR   EV2
      ; NO se escribe en
      ; los puertos de entrada
      RET

   INICIO:
      ACALL   INICIALIZAR
   BUCLE:
      ACALL   MAQ_ESTADOS
      AJMP    BUCLE

MAQ_ESTADOS:
   MOV   A, # estado
   RL    A
   MOV   DPTR, #TBL_ESTADOS
   JMP   @A+DPTR

TBL_ESTADOS:
   AJMP  ESTADO_0
   AJMP  ESTADO_1
   AJMP  ESTADO_2
   AJMP  ESTADO_3

GEN_EVENTOS:
   MOV   nivel , # puerto
   MOV   A, #03H
   ANL   A, nivel
   MOV   evento , A
   RET

ESTADO_0:
   ACAL  GEN_EVENTOS
   MOV   A, #evento
   RL    A
   MOV   DPTR, #TBL_EVENTOS_0
   JMP   @A+DPTR

TBL_EVENTOS_0:
   AJMP  NADA
   AJMP  NIVEL_BAJO
   AJMP  NIVEL_ALTO
   AJMP  ERROR

ESTADO_1:
   ACALL  GEN_EVENTOS
   MOV   A, # evento
   RL A
   MOV   DPTR, #TBL_EVENTOS_1
   JMP   @A+DPTR

TBL_EVENTOS_1:
   AJMP  NIVEL_NORMAL
   AJMP  NADA
   AJMP  NADA
   AJMP  ERROR

EVENTO_2:
   ACALL  GEN_EVENTOS
   MOV   A, # evento
   RL A
   MOV   DPTR, #TBL_EVENTOS_2
   JMP   @A+DPTR

TBL_EVENTOS_2:
   AJMP  NIVEL_NORMAL
   AJMP  NADA
   AJMP  NADA
   AJMP  ERROR

EVENTO_3:
   ACALL  GEN_EVENTOS
   MOV   A, # evento
   RL A
   MOV   DPTR, #TBL_EVENTOS_3
   JMP   @A+DPTR
```

```
TBL_EVENTOS_3:
    AJMP  NIVEL_NORMAL_ERROR
    AJMP  NIVEL_BAJO_ERROR
    AJMP  NIVEL_ALTO_ERROR
    AJMP  NADA

NIVEL_NORMAL:
    SETB  EV1
    SETB  EV2
    MOV   estado, #00H
    RET

NIVEL_BAJO:
    SETB  EV1
    CLR   EV2
    MOV   estado, #01H
    RET

NIVEL_ALTO:
    CLR   EV1
    SETB  EV2
    MOV   estado, #02H
    RET

ERROR:
    CLR   EV1
    CLR   EV2
    SETB  ERR
    MOV   estado, #03H
    RET

NIVEL_NORMAL_ERROR:
    SETB  EV1
    SETB  EV2
    CLR   ERR
    MOV   estado, #00H
    RET

NIVEL_BAJO_ERROR:
    SETB  EV1
    CLR   EV2
    CLR   ERR
    MOV   estado, #01H
    RET

NIVEL_ALTO_ERROR:
    CLR   EV1
    SETB  EV2
    CLR   ERR
    MOV   estado, #02H
    RET

NADA:
    RET
```

3 Se debe representar mediante un diagrama de flujo un programa que se ejecuta cada 1 ms y que realice las siguientes tareas:

a) Cada 10 ms, el programa debe solicitar inicio de lectura del convertidor AD.
b) Cada 100 ms, el programa debe incrementar un contador. Cuando el valor del contador sea veinte, este contador se debe resetear y, posteriormente, debe continuar contando.





* Se supone que este programa forma parte de uno más grande, que tendrá un timer en funcionamiento, generando interrupciones cada 1 ms. Cada vez que ocurra esta interrupción el programa (principal) saltará a una rutina de atención a la interrupción del timer. En esa rutina se activará el TICK_1ms, una variable (bool) que indica que ha transcurrido 1 ms.

** ADCS (ADC Start) es el registro que inicia la conversión del ADC. Pero antes de encenderlo hay que configurar el canal y el modo del ADC, y activar las interrupciones. Si sólo va a medirse una cosa (p.e. peso) se puede configurar en las inicializaciones, pero si se leen varias cosas (p.e. peso, temperatura,...) hay que hacerlo justo antes de usarlo. Las interrupciones siempre hay que activarlas antes de encenderlo.

Se quiere configurar el timer para crear interrupciones cada 1 ms: $T_{int} = 1$ ms

$$f_{int} = \frac{f_{osc}}{pres \cdot cont} \rightarrow \frac{1}{T_{int}} = \frac{f_{osc}}{pre \cdot cont} \rightarrow T_{int} = \frac{pre \cdot cont}{f_{osc}}$$

$$cont = \frac{T_{int} \cdot f_{osc}}{pre} = \frac{1 \cdot 10^{-3} \cdot 24 \cdot 10^{6}}{12} = 2\,10^{3} = 2000$$

$$TMR = Mod - cont = 2^{16} - 2000 = 63536 \rightarrow F830h \rightarrow TH0 = F8h,\ TL0 = 30h$$

```
        ⋮
; contadores
cont_1ms    EQU   0X20
contador    EQU   0X21
; flags
TICK_1ms    EQU   0X22.0
TICK_10ms   EQU   0X22.1
TICK_100ms  EQU   0X22.2
TICK_ADC    EQU   0X22.3
        ⋮

ORG 0X00
   AJMP INICIO

ORG 0X0B
   PUSH ACC
   PUSH PSW
   MOV TH0, #0F8h
   MOV TL0, #030h
   SETB TICK_1ms
   POP PSW
   POP ACC
   RETI

ORG 0X53
   SETB TICK_ADC
   RETI

ORG 0X7B
   INICIALIZAR
        ⋮
      MOV cont_1ms, #00h
      MOV contador, #00h
      CLR TICK_1ms
      CLR TICK_10ms
      CLR TICK_100ms
      ;configurar TIMER0
      MOV TH0, #0F8h
      MOV TL0, #030h
      MOV TMOD, #01h
      ;configurar ADC
      ANL ADCON, #0F8h
      CLR ADEX
      ;activar interrupciones timer
      SETB ET0
      SETB EA
      ;encender timer
      SETB TR0
      RET

INICIO:
   ACALL INICIALIZAR
BUCLE:
   ACALL MAQ_ESTADOS
   AJMP BUCLE

MAQ_ESTADOS:
   MOV A, #ESTADO
   RL A
   MOV DPTR, #TBL_ESTADOS
   JMP @A+DPTR

TABLA_ESTADOS:
        ⋮

ESTADO_X:
   ACALL GEN_EVENTOS_X
   MOV A, EVENTO
   RL A
   MOV DPTR, #TBL_EVENTOS
   JMP @A+DPTR

GEN_EVENTOS:
   JB TICK_1ms, GE_1ms
```