Universidad de Costa Rica

Escuela de Ingeniería Eléctrica

Microprocesadores

IE0623

Tarea 5

Autores:
Robin González Ricz B43011
Michelle Gutiérrez Muñoz
B43195

Profesor:
Esteban Badilla
Asistente:
Mariela Hernandez

14 de febrero de 2021

Índice

Índice de figuras

1. Estructuras de datos

Banderas: x:x:x:CambMod:ModActual:ARRAY_OK:TCL_LEIDA:TCL_LISTA

MAX_TCL: Constante, Indica el tamaño máximo que se acepta con la Tarea Teclado

Tecla: Byte, se utiliza en las subrutinas de la tarea 4

Tecla_IN: Byte, se utiliza en las subrutinas de la tarea 4

Cont_Reb: Byte, contador de rebotes

Cont_TCL: Byte, contador de cuántas teclas se han ingresado

Patron: Byte, se utiliza en Tarea Teclado

Num_Array: Arreglo, Tabla donde se guardan los valores de las Teclas ingresadas.

CUENTA: Byte, se utiliza para llevar el conteo de "tornillos.[en] el modo RUN

AcmPQ: Byte, se utiliza para llevar el conteo de los paquetes de tornillos en el modo RUN.

CantPQ: Byte, es la variable en la cuál se ingresa el valor máximo para los paquetes.

TIMER_CUENTA: Byte, es utilizada para dar cadencia de incremento a Cuenta.

LEDS: Byte,

BRILLO: Byte, variable para aumentar o decrementar brillo (de forma inversa).

CONT_DIG: Byte, se utiliza en la subrutina OC4 para configurar el Brillo.

CONT_TICKS: Byte, se utiliza en la subtutina OC4 para configurar el Brillo.

DT: Byte, se utiliza en la subtutina OC4 para configurar el Brillo.

BIN1: Byte, se utiliza para guardar el valor de cuenta a desplegar en la pantalla de 7 SEG.

BIN2: Byte, se utiliza para guardar el valor de AcmPQ a desplegar en la pantalla de 7 SEG.

BCD_L: Byte, se utiliza en la subrutina BIN_BCD para guardar el resultado.

LOW: Byte, se utiliza en la subrutina BIN_BCD.

TEMP: Byte, variable auxiliar.

BCD1: Byte, se utiliza en difetentes secciones del código para guardar los valores de interés en BCD.

BCD2: Byte, se utiliza en difetentes secciones del código para guardar los valores de interés en BCD.

DISP1: Byte, se utiliza para enviar el valor correspondiente al dislay 1 de la pantalla.

DISP2: Byte, se utiliza para enviar el valor correspondiente al dislay 2 de la pantalla.

DISP3: Byte, se utiliza para enviar el valor correspondiente al dislay 3 de la pantalla.

DISP4: Byte, se utiliza para enviar el valor correspondiente al dislay 4 de la pantalla.

CONT_7SEG: Word, variable utilizada en OC4 para controlar el tiempo.

Cont_Delay: Byte, ontador para el delay.

D2mS: Constante, para crear retardo de 2 mS.

D260uS: Constante, para crear retardo de 260 uS.

D40uS: Constante, para crear retardo de 40 uS.

Clear_LCD: Constante, para borrar pantalla LCD.

ADD_L1: Constante, para agregar línea 1.

ADD_L2: Constante, para agregar línea 2.

Teclas: Arreglo, contiene la asignación correspondiente para los valores del teclado.

SEGMENT: Arreglo, contiene la asignación correspondiente para los segmentos de la pantalla de 7 segmentos.

iniDsp: Arreglo, comandos para iniciarlizar la pantalla LCD.

Msg1_L1: Mensaje 1 de la línea 1.

Msg1_L2: Mensaje 1 de la línea 2.

Msg2_L1: Mensaje 2 de la línea 1.

Msg2_L2: Mensaje 2 de la línea 2.

2. Programa principal

El programa principal en la primera corrida (ya que CantPQ es cero) entra por defecto al Modo Configuración, en el cuál es posible programar el número de tornillos deseados por paquete. Después de lla primera corrida, de acuerdo con la bandera Cambio de Modo y el Modo seleccionado, entra al Modo Configuración o al Modo Run, y también utiliza esta bandera para saber si debe refrescar la pantalla LCD o no, ya que llama a las subrutinas necesarias para cargar los mensajes correspondientes a cada modo en la pantalla LCD.

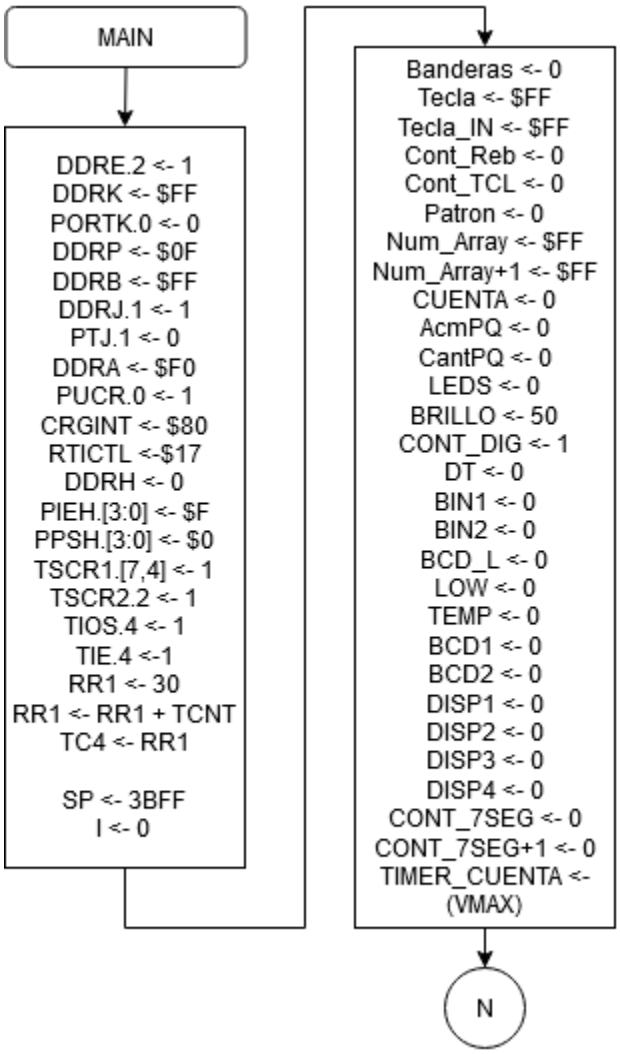

Figura 1: Diagrama de flujo del programa principal, parte 1.

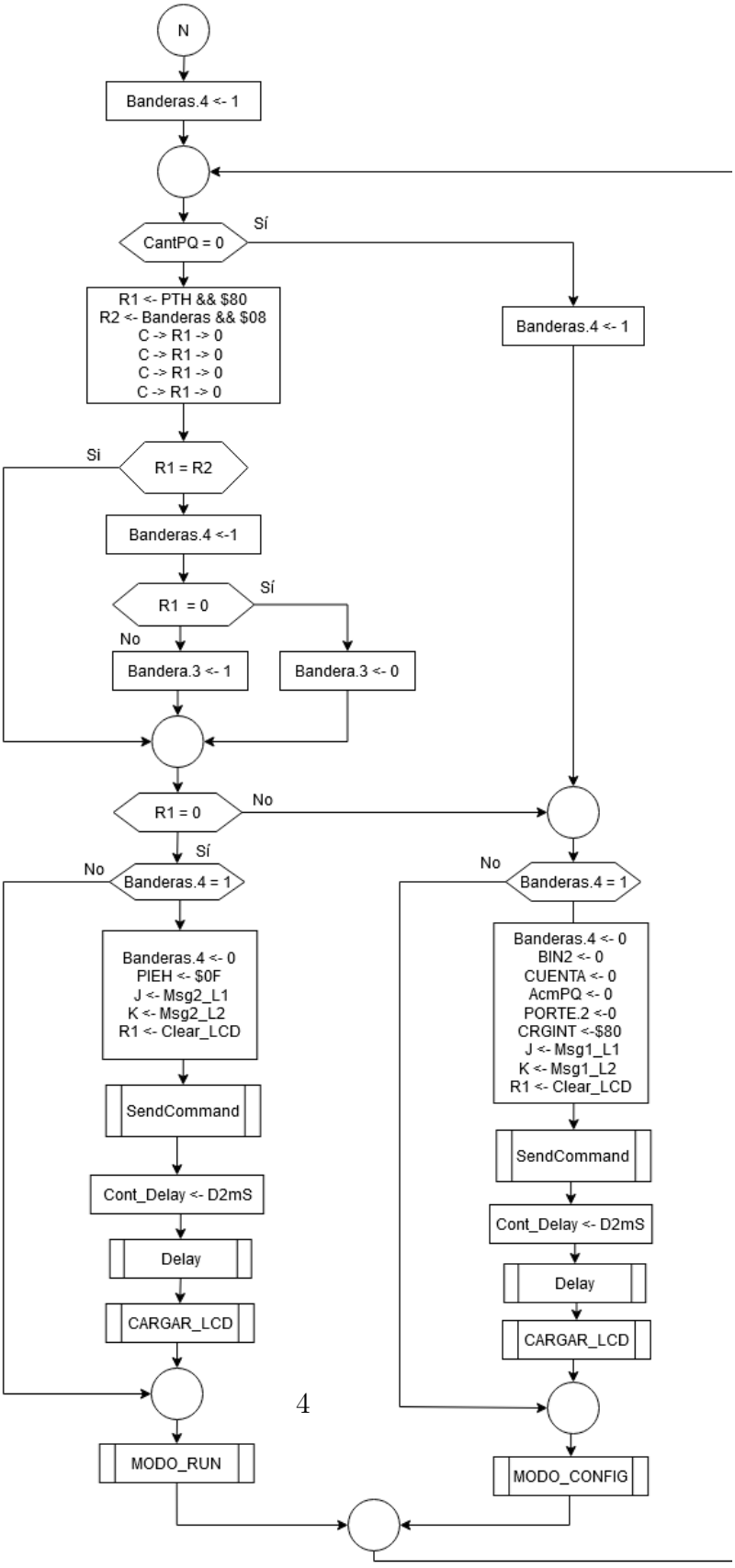

Figura 2: Diagrama de flujo del programa principal, parte 2.

2.1. Memorias de cálculo

2.1.1. Cálculos para RTI

Para obterner un periodo de 1 ms en la subrutina RTI:
M = 1
N = 7
En la dragon el Osc_CLK es 8 MHZ

$$T_{RTI} = \frac{(N+1)\cdot 2^{(M+9)}}{Osc\_CLK} = \frac{8x2^{10}}{8\cdot 10^6} = 1{,}024\cdot 10^{-3}s \tag{1}$$

Por lo cual hay un error debido a la baja granularidad en configuracion de los tiempos de interrupción de:

$$\left|\frac{V_{REAL} - V_{APROXIMADO}}{V_{REAL}}\right| * 100 = \left|\frac{0{,}001 - 0{,}001024}{0{,}001}\right| * 100 = 2{,}4\,\% \tag{2}$$

2.1.2. Cálculos para OC4

$$T_{OC} = \frac{PRSxTC_n}{BusClk} \tag{3}$$

Usando un valor de PRS de 16, entonces, el T_{Cn} es igual a:

$$T_{Cn} = \frac{T_{OC}BusClk}{PRS} = \frac{50KHz^{-1}24MHz}{16} = 30 \tag{4}$$

$$Error = 0\,\% \tag{5}$$

3. Subrutinas

3.1. Cargar_LCD

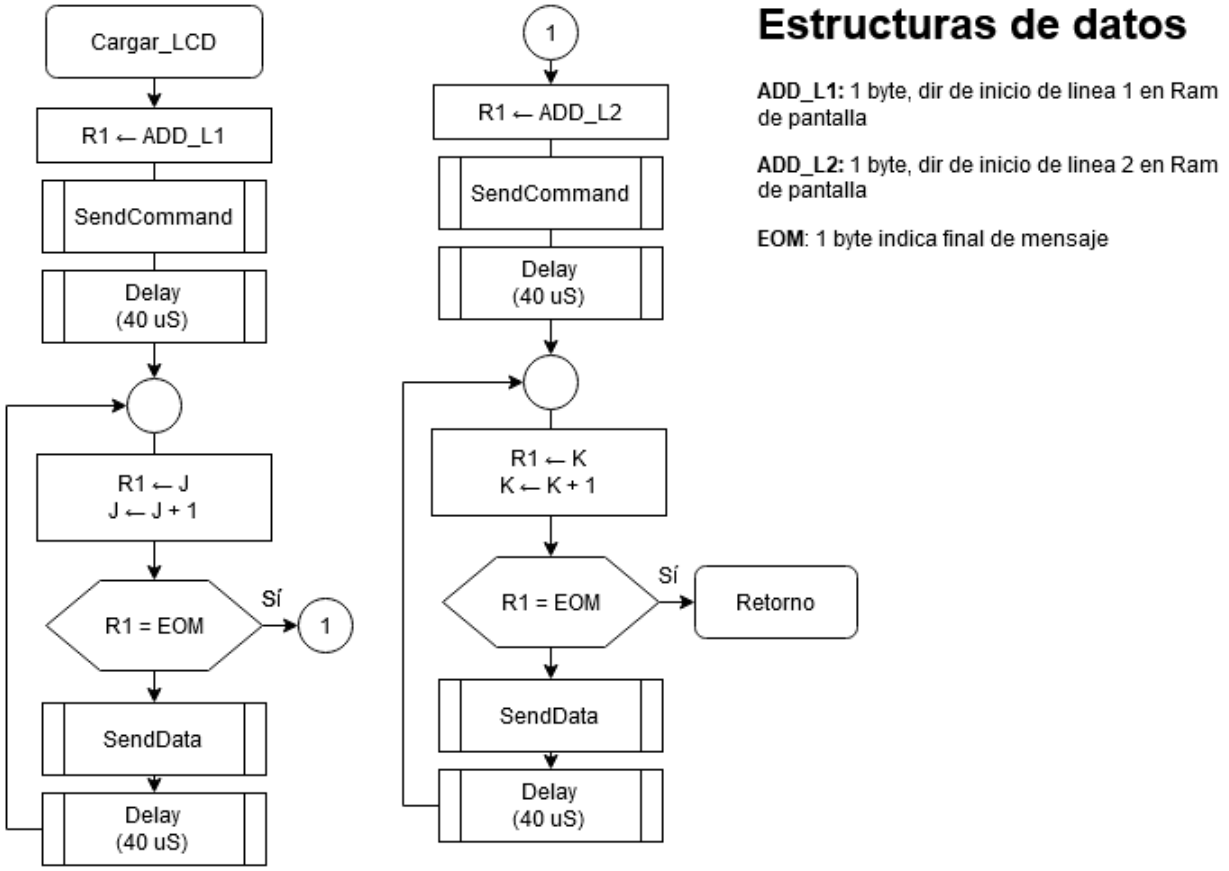

Figura 3: Subrutina Cargar_LCD

3.2. Send Command y Send Data

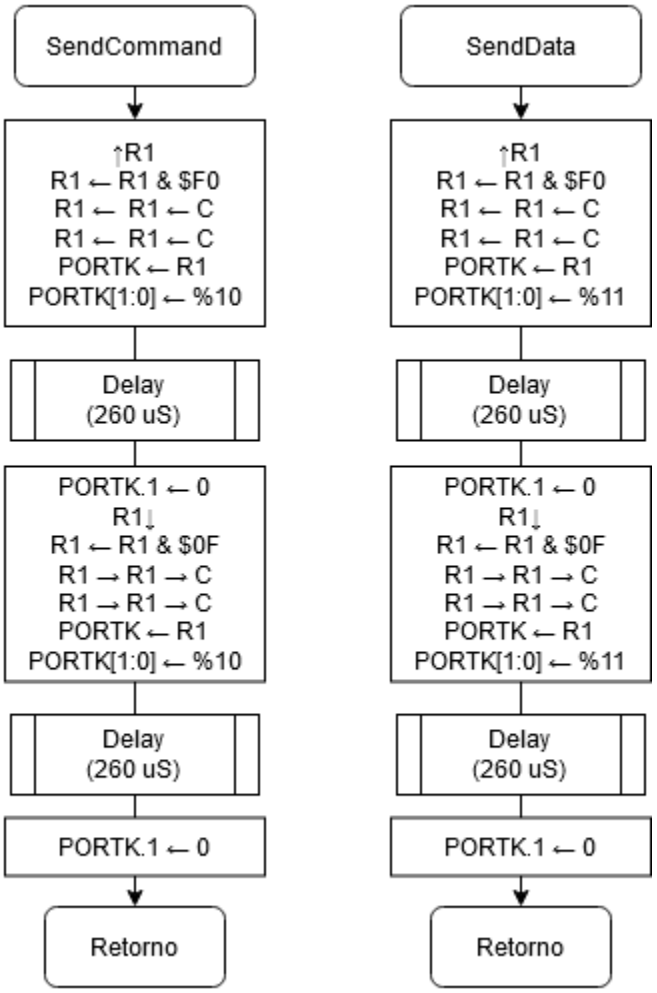

Figura 4: Subrutinas Send Command y Send Data.

3.3. Modo_RUN

En esta subrutina según el valor de Timer Cuenta, incrementa el valor de Cuenta, compara si Cuenta es igual a CantPQ, si lo es, entonces, se detiene, aumenta el contador de paquetes AcmPQ, y activa la Salida (el relé), también si AcmPQ llega a 100, se rebasa y empieza en 0 el contador.

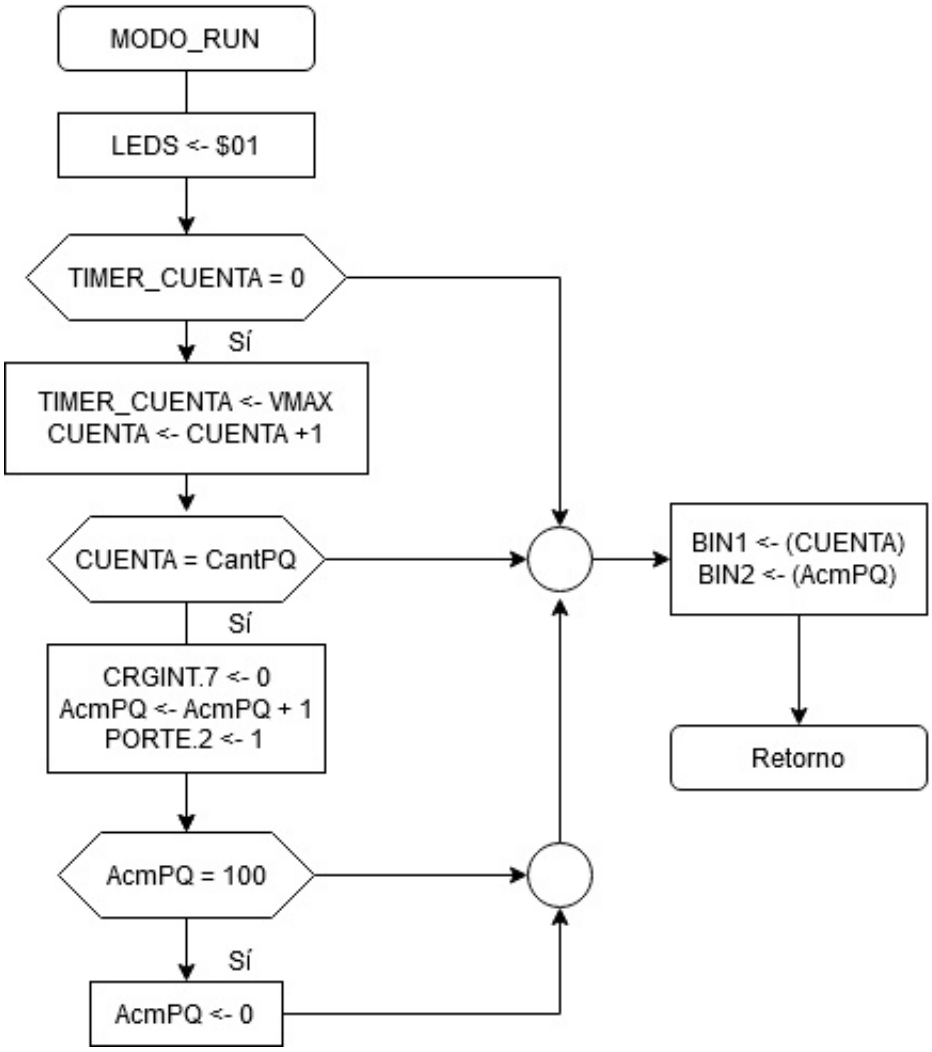

Figura 5: Subrutina MODO_RUN.

3.4. Modo_CONFIG

Esta subrutina llama a Tarea Teclado si la bandera Array_Ok está en cero, es decir, en Num_Array está vacío, Tarea Teclado pone Array_Ok en uno un vez que llena Num_Array, entonces cuando vuelve entrar a Modo Configuración, borra la bandera, y llama a BCD_BIN para que pase los valores de Num_Array de BCD a

Binario y luego se guarden en CantPQ, luego de esto, verifica si el valor es válido, según las especificaciones dadas, en caso de que no lo sea, borra CantPQ y retorna, si es válido, entonces pasa el valor a BIN1, borra Num_Array y retorna.

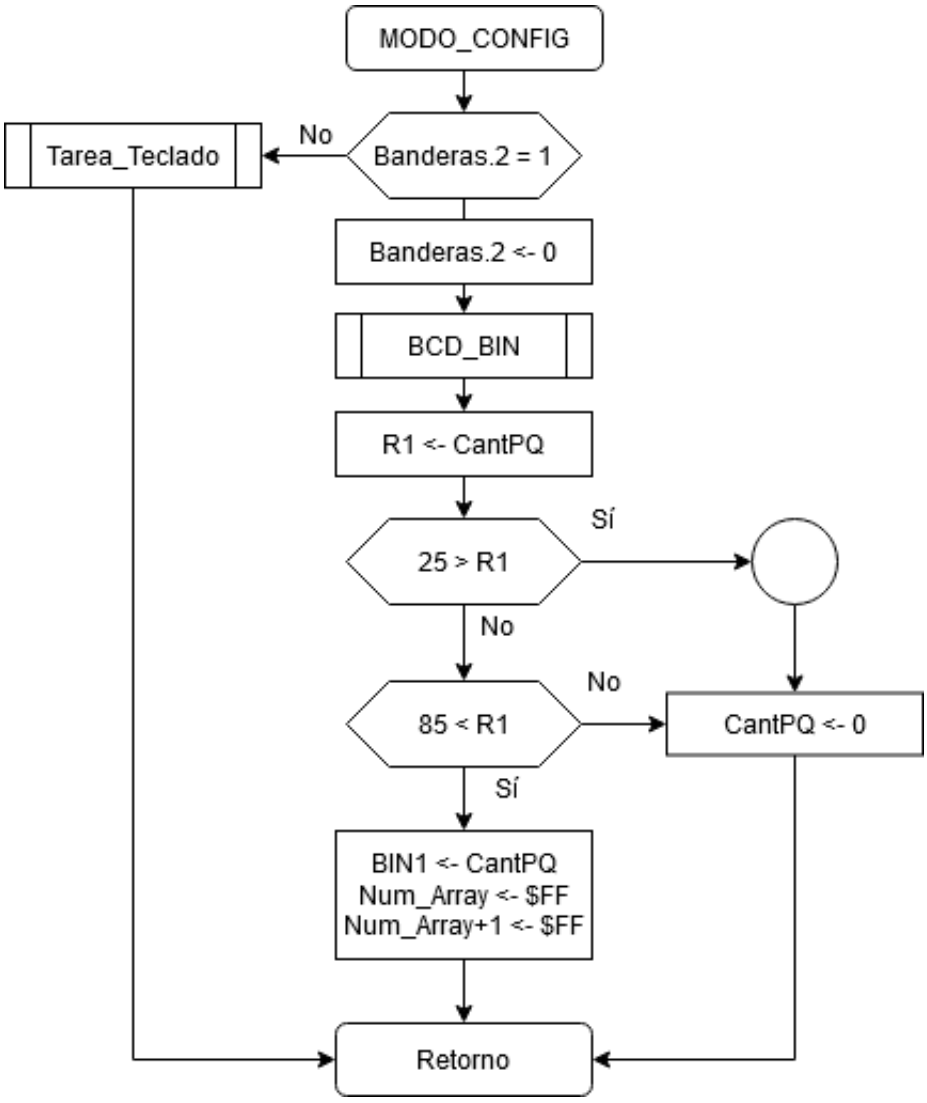

Figura 6: Subrutina MODO_CONFIG.

3.5. BCD_7SEG

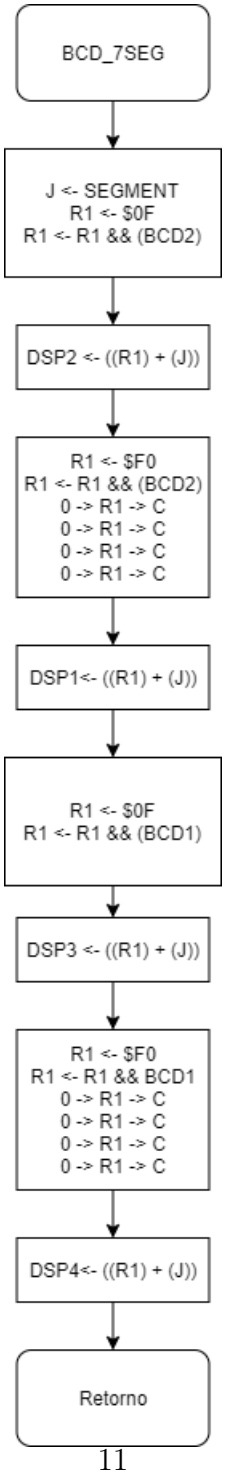

Figura 7: Subrutina BCD_7SEG.

3.6. BCD_BIN

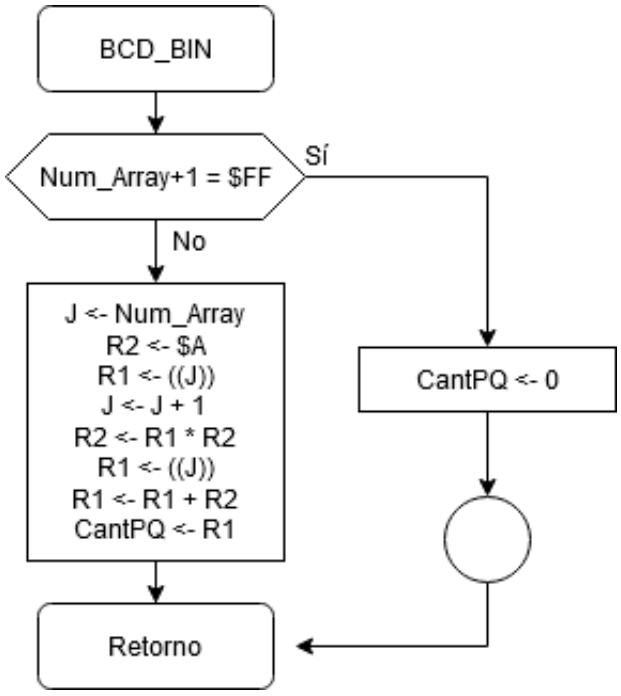

Figura 8: Subrutina BCD_BIN.

3.7. BIN_BCD

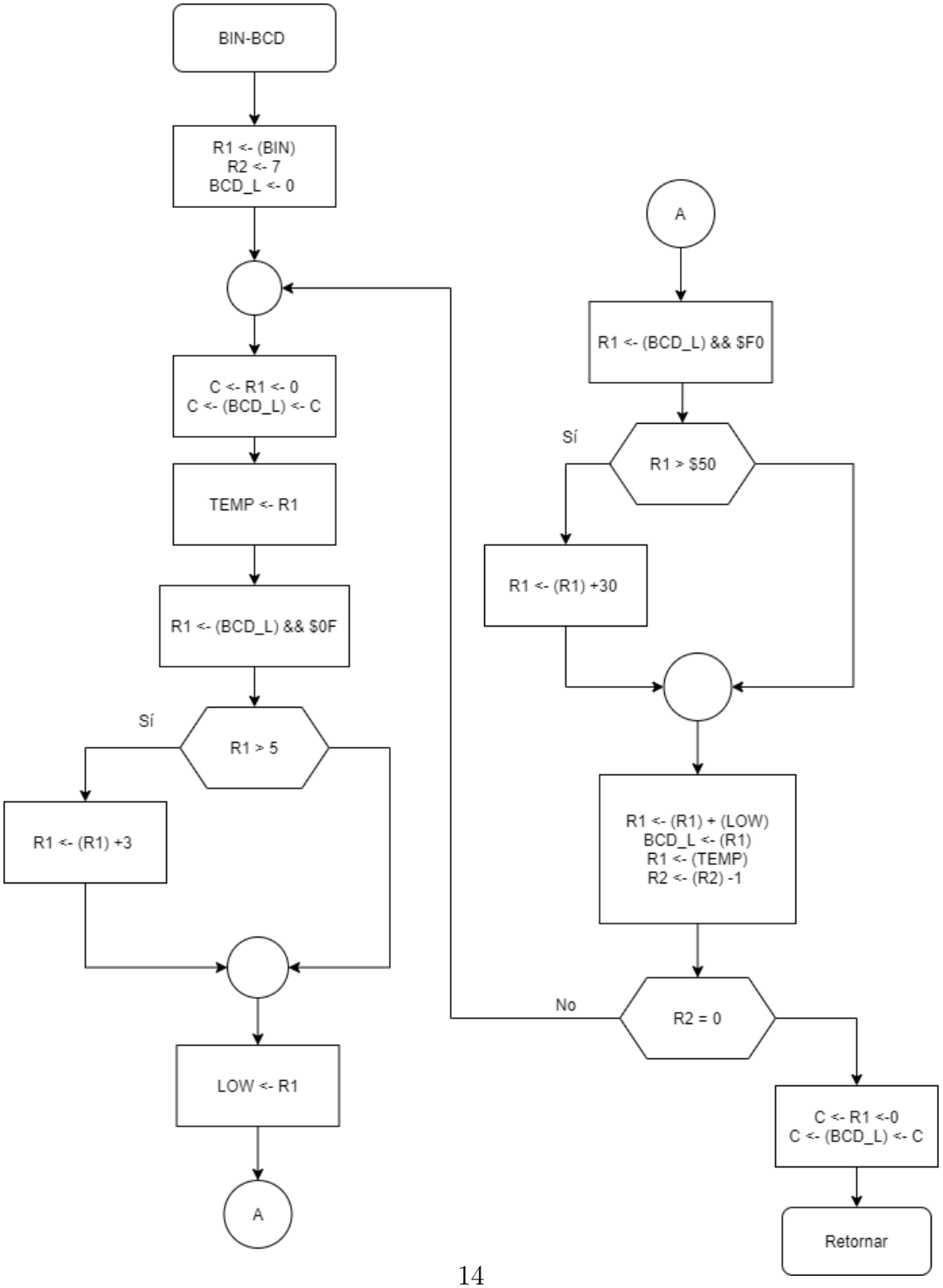

Figura 9: Subrutina BIN_BCD.

3.8. CONV_BIN_BCD

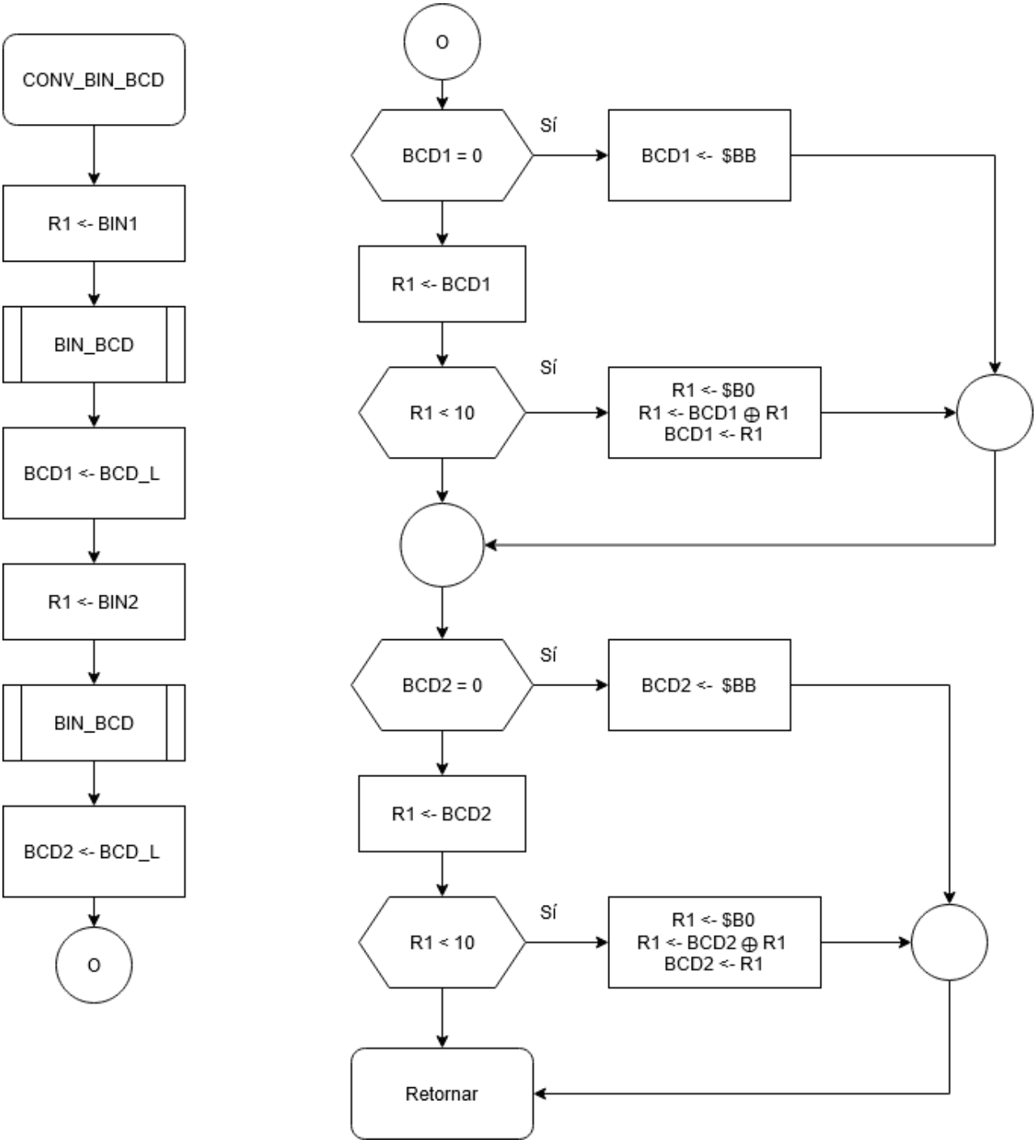

Figura 10: Subrutina CONV_BIN_NCD.

3.9. Delay

Delay unicametne se encarga de esperar que el contador sae 0 y no hacer nada más.

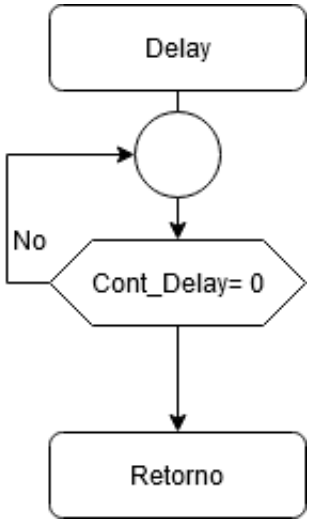

Figura 11: Subrutina Delay.

4. Subrutinas de interrupción

4.1. Real Time Interrupt

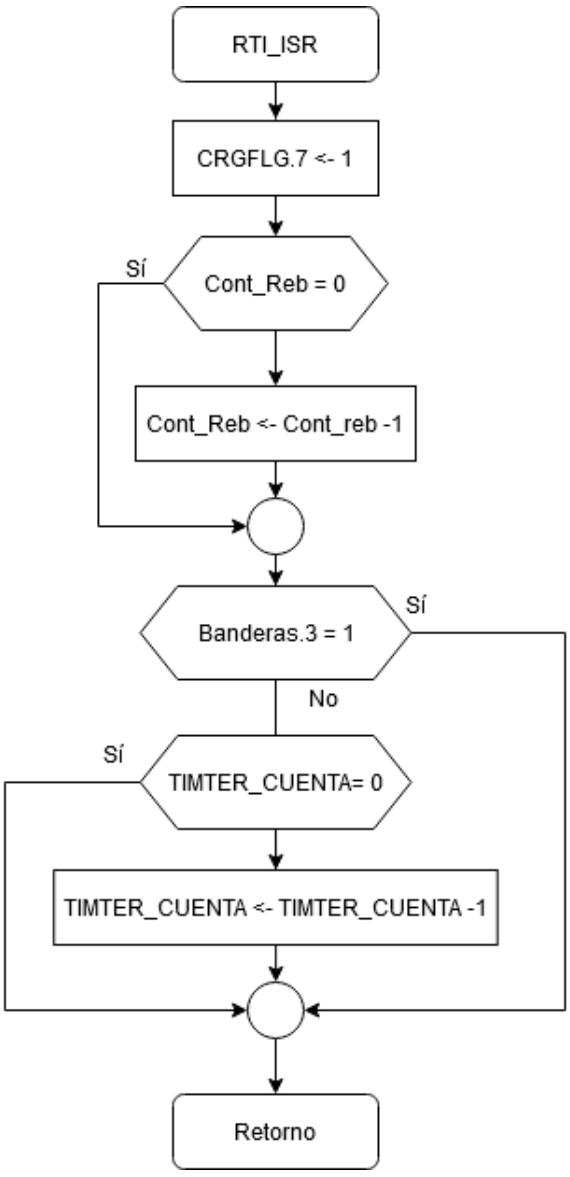

Figura 12: Subrutina RTI_ISR.

4.2. Key wakeups puerto H

Esta subrutina se encarga de anteder diferentes funciones con interrupciones según el botón presionado, con PH0, hace un borrado de Cuenta para iniciar de nuevo el conteo, con PH1, hace un borrado de AcmPQ, con PH2 y PH3, incrementa y decrementa el brillo de la pantalla de 7 segmentos.

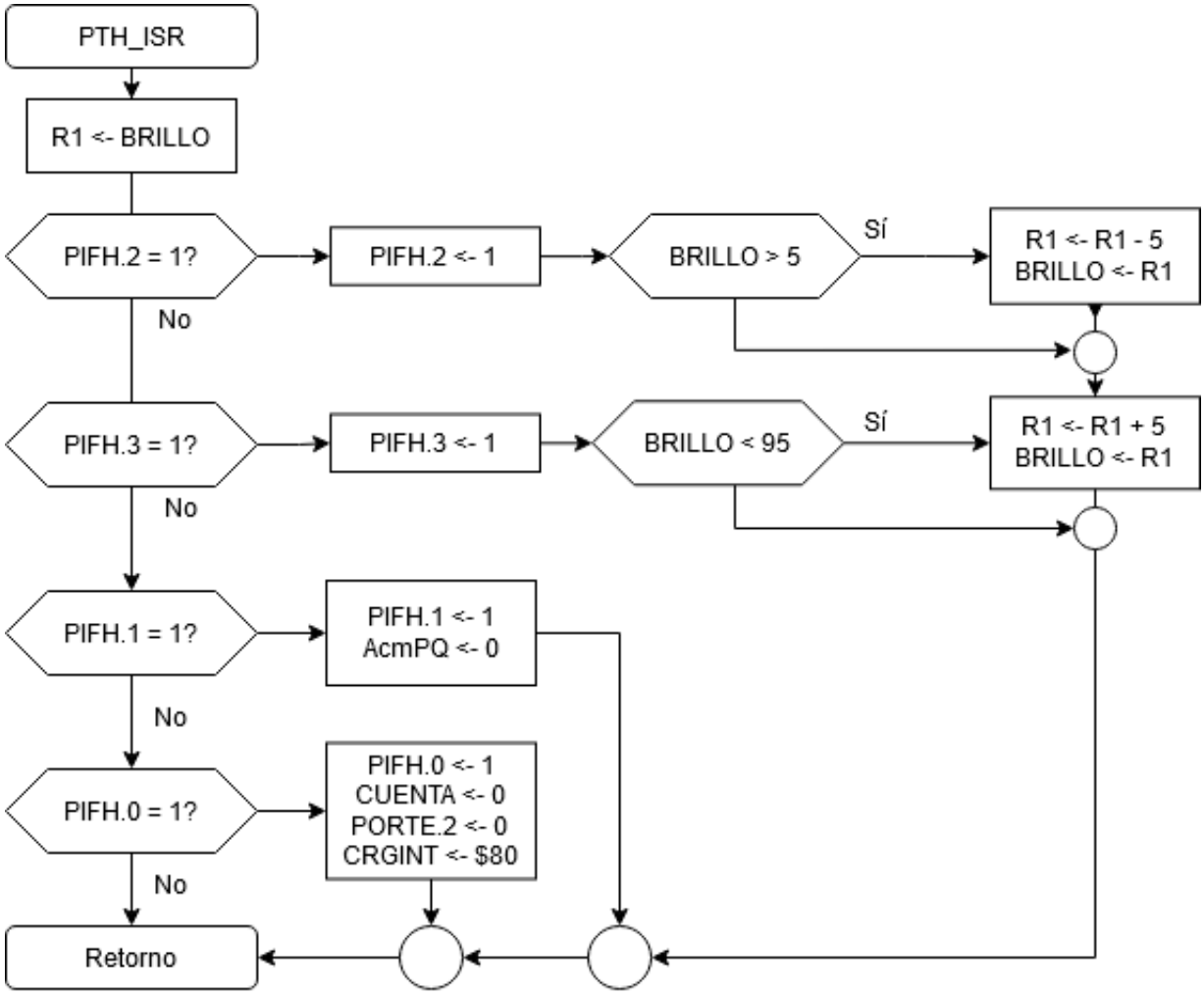

Figura 13: Subrutina PTH_ISR.

4.3. Output Compare canal 4

Esta subrutina de interrupción se encarga de descontar tres valores: Cont_Delay, para el conteo de tiempo en Delay, Cont_Ticks, para la configuración del brillo y Cont_7seg para la manipulación de la pantalla de 7 segmentos. Además, se encarga de desplegar en los displays de la pantalla de 7 segmentos los valores correspondientes.

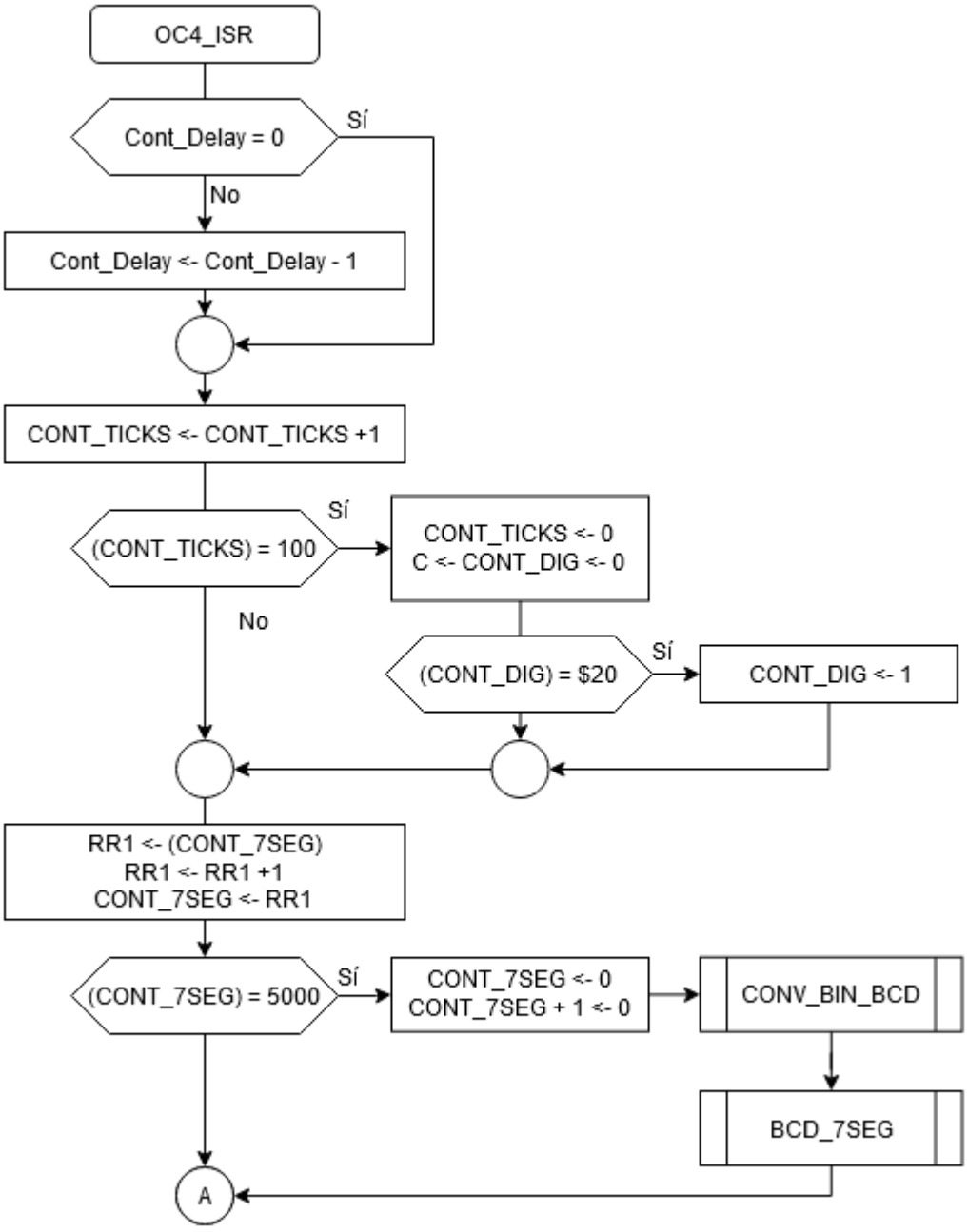

Figura 14: Subrutina OC4_ISR primera parte.

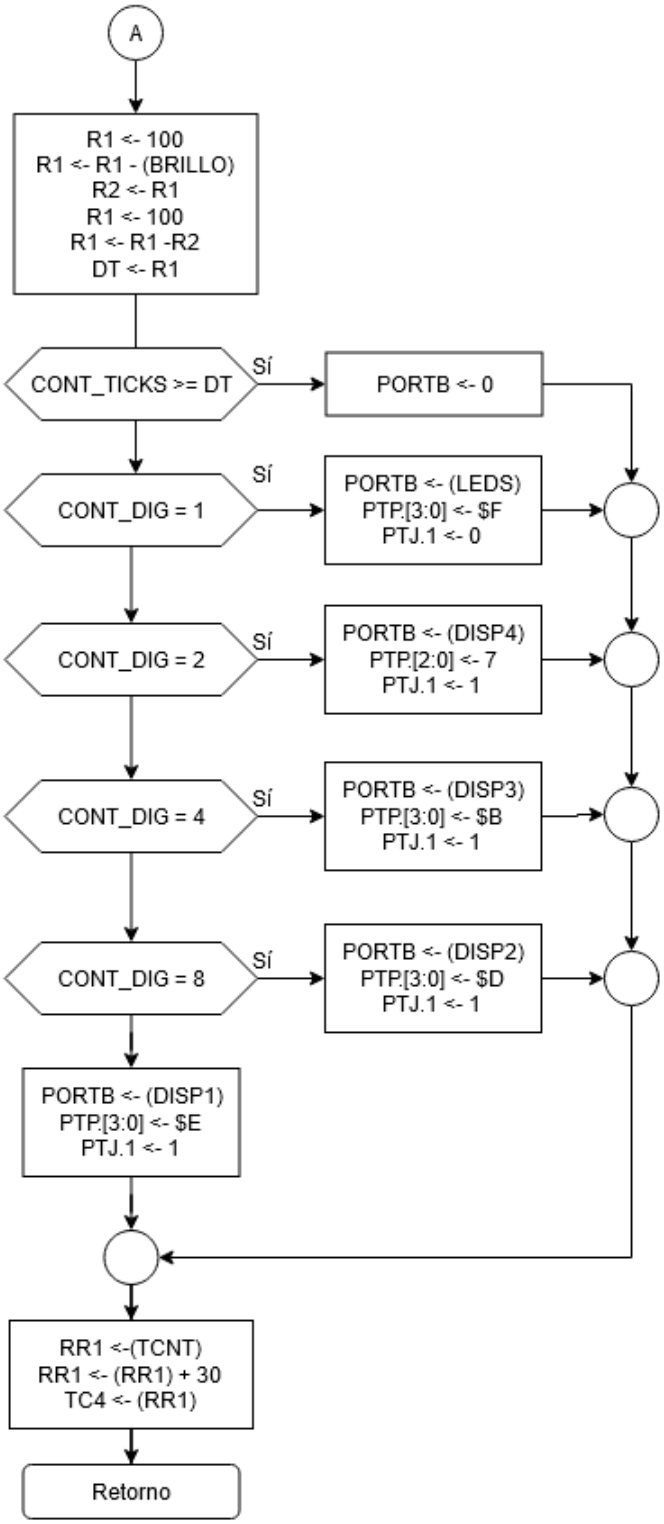

Figura 15: Subrutina OC4_ISR continuación.

5. Pruebas de funcionalidad

Se probaron de manera independiente en un archivo cada una de las subrutinas implementadas en esta tarea. Se comprobó la funcionalidad de todas y cada una de ellas.
Para probar el funcionamiento de la tarea completa primero se inició con modo config poniendo diferentes valores en casos extremos en el teclado verificando su funcionalidad y la de la pantalla. Una vez en modo run se verificó el conteo de tornillos y de cajas y la activación de la salida al llegar a CantPQ, también que PTH_ISR cumple las funciones de reiniciar cuentas y de cambiar el brillo de las pantallas.
Nuevamente se ingresó a modo config donde fue posible cambiar nuevamente el valor de CantPQ e igualmente manipular el brillo de la pantalla y despues de eso volver a modo run para que se ejecutara la cuenta con el nuevo limite de tornillos por caja.