Proyecto Final: Selector 623

Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
IE0623 Microprocesadores GR001
Fecha de entrega: 09/12/2024

Profesor:
Ing. Geovanny Delgado
M.Sc.E.E

Asistente:
Yoel Moya
Carmona

Estudiante:
Roger Daniel
Piovet García
C15990

Índice

1. Resumen

El Selector 623 es un sistema empleado en la industria siderúrgica para el control y la clasificación de barras de aluminio. Este sistema tiene como objetivo principal garantizar que las barras cumplen con una longitud mínima especificada, al mismo tiempo que identifica aquellas que presenten defectos de fabricación o no alcancen las dimensiones requeridas. El operario podrá seleccionar 1 de 3 modos de operación:

- Seleccionar: Esperar una barra para que sea procesada
- Configurar: Ajustar un nuevo valor de *LongOk* por medio de un teclado matricial
- Stop: No se ejecuta ninguna acción

En el modo seleccionar, el proceso comienza con la detección de imperfecciones en las barras recién producidas. Un detector ultrasónico escanea cada barra en busca de irregularidades en su superficie. Si se detecta algún defecto, el sistema activa automáticamente una máquina de corte que elimina la sección defectuosa. Las barras que pasan esta etapa inicial de inspección se cortan a una longitud máxima estándar de 99 cm, siempre que no presenten defectos evidentes. El Selector 623 cuenta con dos sensores ultrasónicos, denominados *S1* y *S2*. Estos sensores están posicionados para medir la longitud de cada barra al pasar por un punto específico del sistema. Cuando la longitud medida es mayor o igual a un valor mínimo programable, conocido como *LongOK*, la barra se considera apta para su comercialización. De lo contrario, la barra es rechazada, para que un operario la retire de la línea de producción, y el Selector 623 despliega un mensaje de error si la barra tenía una longitud o velocidad inválida, la cual se encuentra en el rango de 10cm/seg y 50cm/seg. En el modo Configurar, el operario puede digitar en un teclado matricial un nuevo valor para *LongOK*. Una vez el operario haya digitado el valor deseado, puede confirmar el valor digitado presionando la tecla Enter, o borrar el valor digitado con la tecla Borrar. El valor digitado es desplegado en un set de displays de 7 segmentos que posee el Selector 623. En el modo STOP, no se ejecuta ninguna acción, y se espera a que el operario escoja el modo Configurar o Seleccionar.
El Selector 623 fue implementado en la Dragon 12+, haciendo uso de los dipswitches PH7:PH6 para seleccionar el modo operación. Además de esto, se hace uso de los botones pulsadores PH3:PH0 para emular el funcionamiento de los sensores, los cuales generan un pulso corto en caso de que sean activados. Por último, se hace uso del microrrelé de la tarjeta para activar y desactivar el rociador que marca las barras de aluminio en su centro longitudinal.

2. Diseño de la aplicación

2.1. Esquema general

El esquema de funcionamiento del Selector 623 se muestra en la Figura 1. Primero se configura el hardware por utilizars, así como las variables del programa. Después, se determina el modo de operación en base a los dipswitches accionados por el usuario, y finalmente se ejecuta el modo de operación.

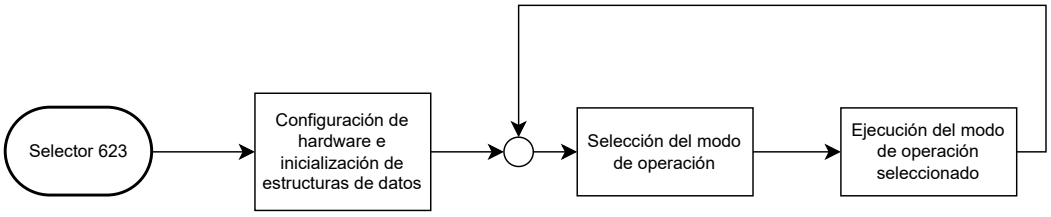

Figura 1: Esquema de funcionamiento del Selector 623

2.2. Estructuras de datos

Las estructuras de datos utilizadas por el Selector 623 se muestran en el Cuadro 2, siendo esta la tabla proporcionada por el profesor. El Selector 623 hace uso de dos variables bandera, denominadas `Banderas_1` y `Banderas_2`. La disposición de los bits de cada una de estas banderas se muestra en el Cuadro 1. En esta tabla, se denota con una `X` aquelllos bits reservados de la variable bandera respectiva.

Nombre del Bit	Posición del Bit
ShortP0	0
LongP0	1
ShortP1	2
LongP1	3
Array_Ok	4
`X`	5
`X`	6
`X`	7

(a): `Banderas_1`

Nombre del Bit	Posición del Bit
RS	0
LCD_Ok	1
FinSendLCD	2
Second_Line	3
`X`	4
`X`	5
`X`	6
`X`	7

(b): `Banderas_2`

Cuadro 1: Variables bandera del Selector 623

VARIABLES	DIRECCIONES	VALORES
Tarea_Teclado		
MAX_TCL	1000	tSupRebTCL
Tecla	1001	
Tecla_IN	1002	
Cont_TCL	1003	
Patron	1004	
Est_Pres_TCL	1005	
Num_Array	1010	
Tarea_PantallaMUX		
Est_Pres_PantallaMUX	1020-1021	tTimerDigito
Dsp1	1022	MaxCountTicks
Dsp2	1023	OFF (Offset Tabla Segment)
Dsp3	1024	GUIONES (Offset Tabla segment)
Dsp4	1025	
LEDS	1026	
Cont_Dig	1027	
Brillo	1028	
Variables para subrutinas de Conversión		
BCD	1029	
Cont_BCD	102A	
BCD1	102B	
BCD2	102C	
Tarea LCD		
IniDsp	102D-1031	tTimer2mS
Punt_LCD	1032-1033	tTimer260uS
ChardLCD	1034	tTimer40uS
Msg_L1	1035-1036	EOB
Msg_L2	1037-1038	Clear_LCD
EstPres_SendLCD	1039-103A	ADD_L1
EstPres_TareaLCD	103B-103C	ADD_L2
Tarea Leer PB1 tyTarea Leer PB2		
EstPres_LeerPB1	103D-103E	PortPB,MaskPB2 ($08)
EstPres_LeerPB2	103F-1040	PortPB,MaskPB2 ($01)
		tSupRebPB
		tShortP
		tLongP
Tarea Configurar		
Est_Pres_TConfig	1041-1042	LDConfig
ValorLong	1043	Lmin
LongOK	1044	Lmax
Tarea STOP		
		LDStop
Tarea Seleccionar		
Est_Pres_TSelec	1045-1046	LDSelect
Longitud	1047	tTimerCal
DeltaT	1048	tTimerError
Velocidad	1049	tTimerShott
		VelocMin
		VelocMax
		DeltaX_S
		DeltaX_R
		PortRele
		MaskRele
Tarea Brillo		
Est_Pres_TBrillo	104A-104B	tTimerBrillo
		MaskSCF
Tarea Leer DS		
Est_Pres_LeerDS	104C-104D	tTimerRebDS
Temp_DS	104E	
Valor_DS	104F	

VARIABLES	DIRECCIONES	VALORES
BANDERAS		
Banderas_1	1070	
ShortP1		Mask $01
LongP1		Mask $02
ShortP2		Mask $04
LongP2		Mask $08
ArrayOK		Mask $10
Banderas_2	1071	
RS		Mask $01
LCD_Ok		Mask $02
FinSendLCD		Mask $04
Second_Line		Mask $08
Generales		
LED_Testigo	1080	tTimerLDTst
		Carga_TC4
TABLAS		
Segment	1100	
Teclas	1110	
MENSAJES		
	1200	
TABLA DE TIMERS		
	1500	tTimer1mS
Timer1mS		tTimer10mS
Timer10mS		
Timer100mS		
Timer1S		
CounterTicks		tTimer100mS
Timer260uS		tTimer1S
Timer40uS		
Timer_RebPB1		
Timer_RebPB2		
Timer_RebTCL		
Timer_RebDS		
TimerDigito		
Timer2mS		
Timer_SHP1		
Timer_SHP2		
TimerCal		
TimerError		
TimerPant		
TimerFinPant		
TimerRociador		
TimerShot		
TimerBrillo		
Timer_LED_Testigo		
Timer_LP1		
Timer_LP2		

Cuadro 2: Estructuras de datos del Selector 623

2.3. Memoria de cálculo

2.3.1. Salida por comparación (OC)

Con el propósito de configurar la frecuencia de interrupción de la máquina de tiempos a 50 kHz y generar bases de tiempo adecuadas para el uso adecuado de la pantalla LCD, se configuró el canal 4 de salida por comparación para este propósito. El periodo de interrupción T_{oc} viene dado por:

$$T_{oc} = \frac{1}{50\,\text{kHz}} = 20\,\mu\text{s} \tag{1}$$

Para la Dragon 12+ con el Debug12, se cumple:

$$SysCLK = 48\,\text{MHz}$$
$$BusCLK = 24\,\text{MHz}$$

El periodo de interrupción por salida de comparación viene dado por:

$$T_{oc} = \frac{PRS \times T_{C4}}{BusCLK} \tag{2}$$

Note que con un divisor $PRS = 1$, se obtiene el valor de carga:

$$\begin{aligned} T_{C4} &= T_{oc} \times BusCLK \\ &= 480 \end{aligned}$$

Por tanto, cada vez que se genere la interrupción, se debe cargar el valor de `TCNT` a un acumulador y sumar el valor de carga indicado, debido a que `TCNT` no se reinicia con cada interrupción para la configuración dada.

2.3.2. Convertidor analógico a digital (ATD)

Con el propósito de leer el valor de tensión generado por el trimmer conectado al PAD7 de la Dragon12+, se configuró el convertidor ATD para realizar 4 conversiones de este canal, con 2 periodos de reloj para el muestreo, en 8 bits si signo, con una frecuencia de muestreo de 700 kHz. Esta frecuencia viene dada por:

$$f_s = \frac{BusCLK}{2(PRS + 1)} \tag{3}$$

Con un divisor $PRS = 16$, note que

$$f_s = 705.88\,\text{kHz} \tag{4}$$

Lo cual es lo suficientemente cercano a la frecuencia de muestreo solicitada para esta aplicación.

2.3.3. Subrutina Calcula

Se plantearon distintas ecuaciones cinemáticas para calcular los distintos parámetros que debe calcular la subrutina `Calcula`. La velocidad de la barra vendrá dada por:

$$Velocidad = \frac{DeltaX_S}{DeltaT1} \tag{5}$$

Donde $DeltaT1$ es el intervalo de tiempo entre la activación de $S1$ y $S2$. La longitud de la barra viene dada por:

$$Longitud = Velocidad \times DeltaT2 \tag{6}$$

Donde $DeltaT2$ es el tiempo entre la activación de $S2$ y $S2$ cuando llega el fin de la barra. $TimerPant$ es calculado de la siguiente manera:

$$TimerPant = \frac{DeltaX_R}{Velocidad} \tag{7}$$

De tal forma que se muestre en pantalla los mensajes cuando el principio de la barra alcance el rociador. $TimerFinPant$ es calculado de la siguiente manera:

$$TimerFinPant = TimerPant + \frac{Longitud}{Velocidad} \tag{8}$$

De tal forma que se para de mostrar el mensaje en pantalla cuando el fin de la barra pasa por el rociador. Por último, $TimerRociador$ se calcula de la siguiente manera:

$$TimerRociador = TimerPant + \frac{TimerFinPant - TimerPant}{2} \tag{9}$$

Esto permite que el rociador rocie la barra en su centro longitudinal.

2.4. Programa principal

En la Figura 2 se muestra el programa principal del Selector 623. Este se estructura de la siguiente manera:

- Configuración de hardware
- Inicialización de estructuras de datos
- Habilitación de interrupciones
- Inicialización de máquinas de estado
- Rutina de inicialización de LCD
- Despachador de tareas

Al ejecutar estas acciones en este orden, se configura todo lo necesario para que las tareas despachadas en el Despachador de Tareas ejecuten correctamente.

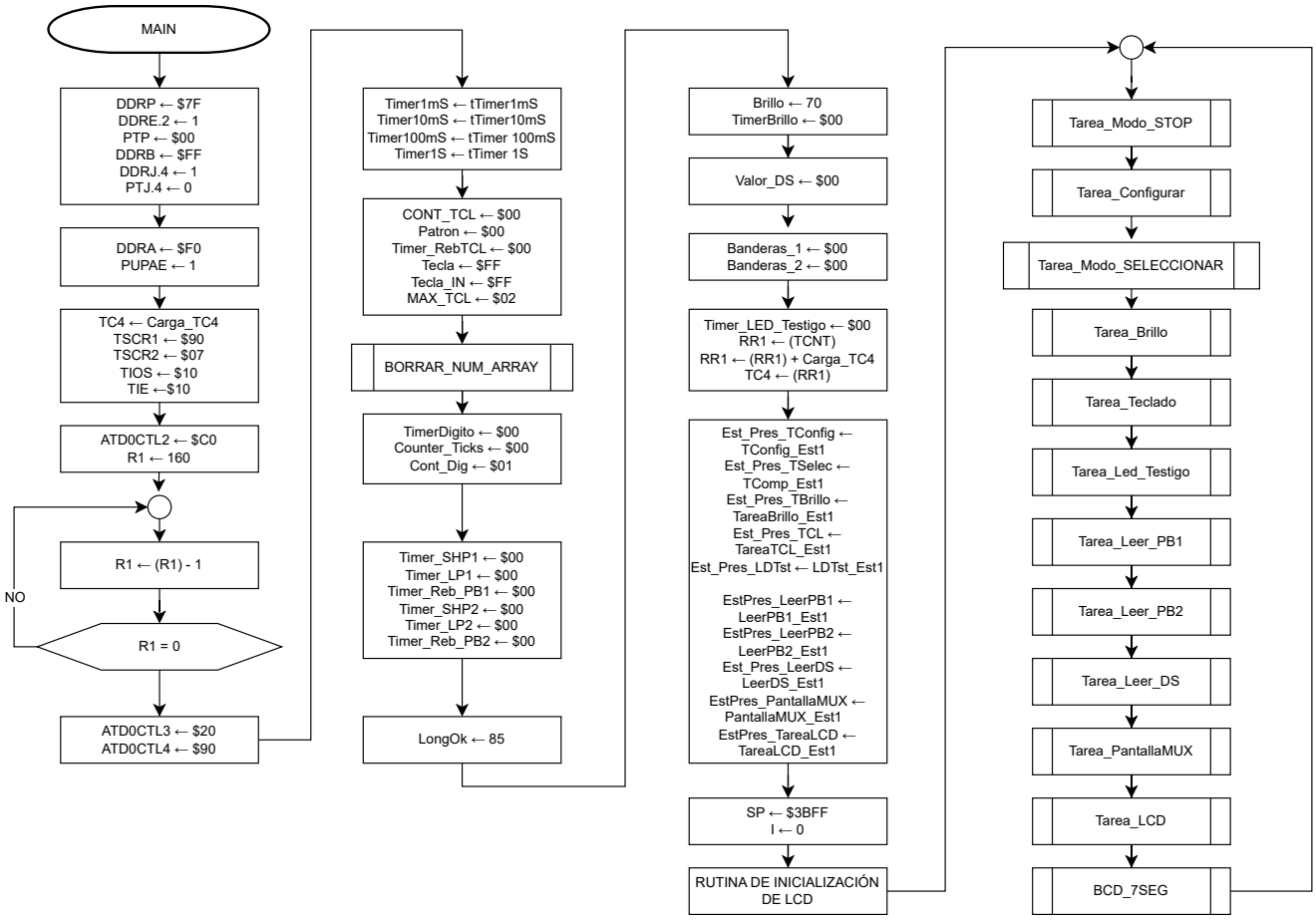

Figura 2: Programa principal del Selector 623

2.5. Rutina de inicialización de LCD

Esta rutina se ejecuta una sola vez después del POR en la inicialización de las estructuras de datos del Selector 623, y su propósito es configurar la pantalla LCD y las esturcturas de datos asociadas para su uso a través de los punteros `MSG_L1`, `MSG_L2`, y la bandera `LCD_Ok`. El diagrama de flujos asociado a esta rutina se muestra en la Figura 3.

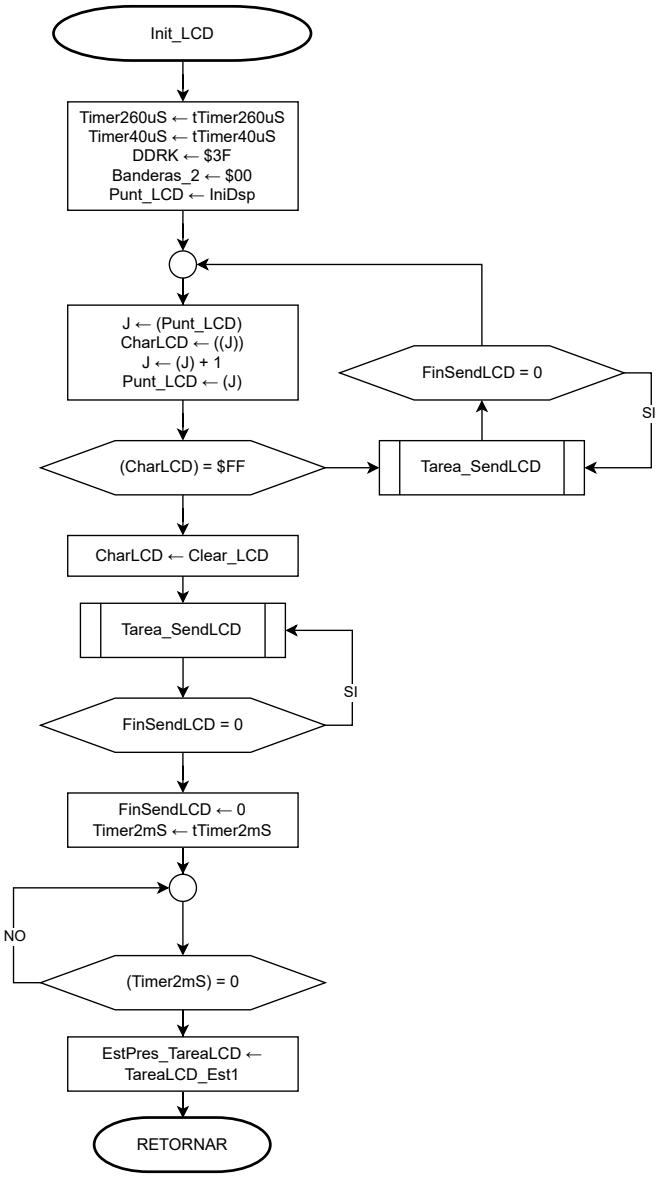

Figura 3: Rutina de inicializaciónd de LCD

2.6. Tareas

2.6.1. Tarea_Modo_STOP

Esta tarea evalúa el valor actual de los dipswitches almacenado en la variable `Valor_DS`, y si el usuario seleccionó el modo STOP del Selector 623 (`Valor_DS` = $C0), carga los punteros `MSG_L1` y `MSG_L2` con los mensajes:

- `MSG_BIENVENIDA_L1`: "SELECTOR 623"
- `MSG_BIENVENIDA_L2`: "MODO STOP"

Además, la tarea actualiza el valor de los LEDs en base al modo actual del Selector 623 a `LDStop`, y muestra los displays apagados. Esto es logrado cargando los valores $BB a `BCD1` y `BCD2`. Como `OFF` = $0B, note que $BB = 16 × `OFF` + `OFF` = 17 × `OFF`. Este detalle del diseño es utilizado ampliamente en el resto de tareas para mostarr los displays apagados sin deshabilitarlos, por lo cual solo se explicará para esta tarea, de tal forma que la explicación se extiende para cualquiera de las otras tareas en donde se utilice esto. Si el valor de `Valor_DS` no es el correspondiente al modo STOP del Selector 623, la tarea no ejecuta ninguna acción. El comportamiento descrito anteriormente se muestra en la Figura 4.

Parámetros de Entrada

- `Valor_DS`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- `LEDS`: Devuelto por medio de direccionamiento a memoria RAM.
- `MSG_L1`: Devuelto por medio de direccionamiento a memoria RAM.
- `MSG_L2`: Devuelto por medio de direccionamiento a memoria RAM.
- `Banderas_2`: Devuelto por medio de direccionamiento a memoria RAM.
- `BCD1`: Devuelto por medio de direccionamiento a memoria RAM.
- `BCD2`: Devuelto por medio de direccionamiento a memoria RAM.

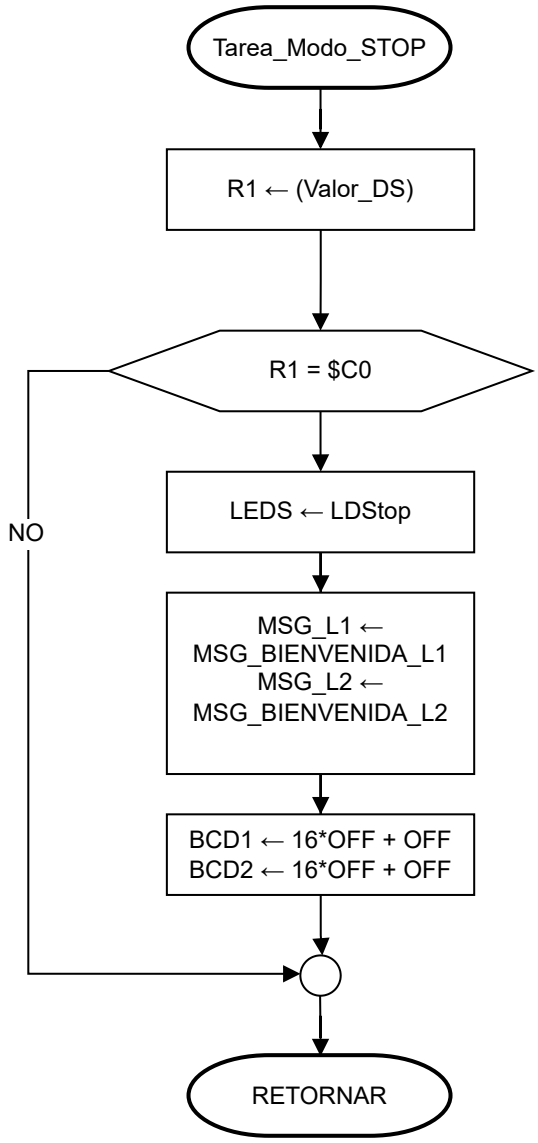

Figura 4: Diagrama de flujos de la Tarea_Modo_STOP

2.6.2. Tarea_Configurar

En esta tarea se evalúa el valor actual de los dipswitches almacenado en la variable `Valor_DS`, y si el usuario seleccionó el modo CONFIGURAR del Selector 623 (`Valor_DS` = $`80`), actualiza el valor de los LEDs en base al modo actual del Selector 623 a `LDConfig`. Además de esto, esta tarea implementa una máquina de estados denominada `TConfig` que permite al usuario digitar en el teclado matricial de la tarjeta Dragon 12+ un nuevo valor para la variable `LongOK` utilizada para discriminar las barras de aluminio de la siderúrgica que procesa el Selector 623 cuando se encuentra en el modo SELECCIONAR. Después de que el usuario digite un nuevo valor para `LongOK` y presione el botón de enter (# en el teclado matricial), el valor digitado es desplegado en la parte baja de los displays de 7 segmentos de la tarjeta Dragon 12+. Por tanto, si `Valor_DS` es el correspondiente al modo CONFIGURAR del Selector 623, la tarea salta a la subrutina correspondiente al estado actual de la máquina de estados, almacenado en la variable `Est_Pres_TConfig`. Si el valor de `Valor_DS` no es el correspondiente al modo CONFIGURAR del Selector 623, la tarea carga el estado inicial de la máquina de estados descrita anteriormente. Esto es con el propósito de que, la próxima vez que se seleccione este modo de operación del Selector 623, salte al primer estado de la máquina de estados. El comportamiento descrito anteriormente se muestra en la Figura 5.

Parámetros de Entrada

- `Valor_DS`: Accedido por medio de direccionamiento extendido.
- `Est_Pres_TConfig`: Accedido

Parámetros de Salida

- `LEDS`: Devuelto por medio de direccionamiento a memoria RAM.
- `Est_Pres_TConfig`: Devuelto por medio de direccionamiento a memoria RAM.

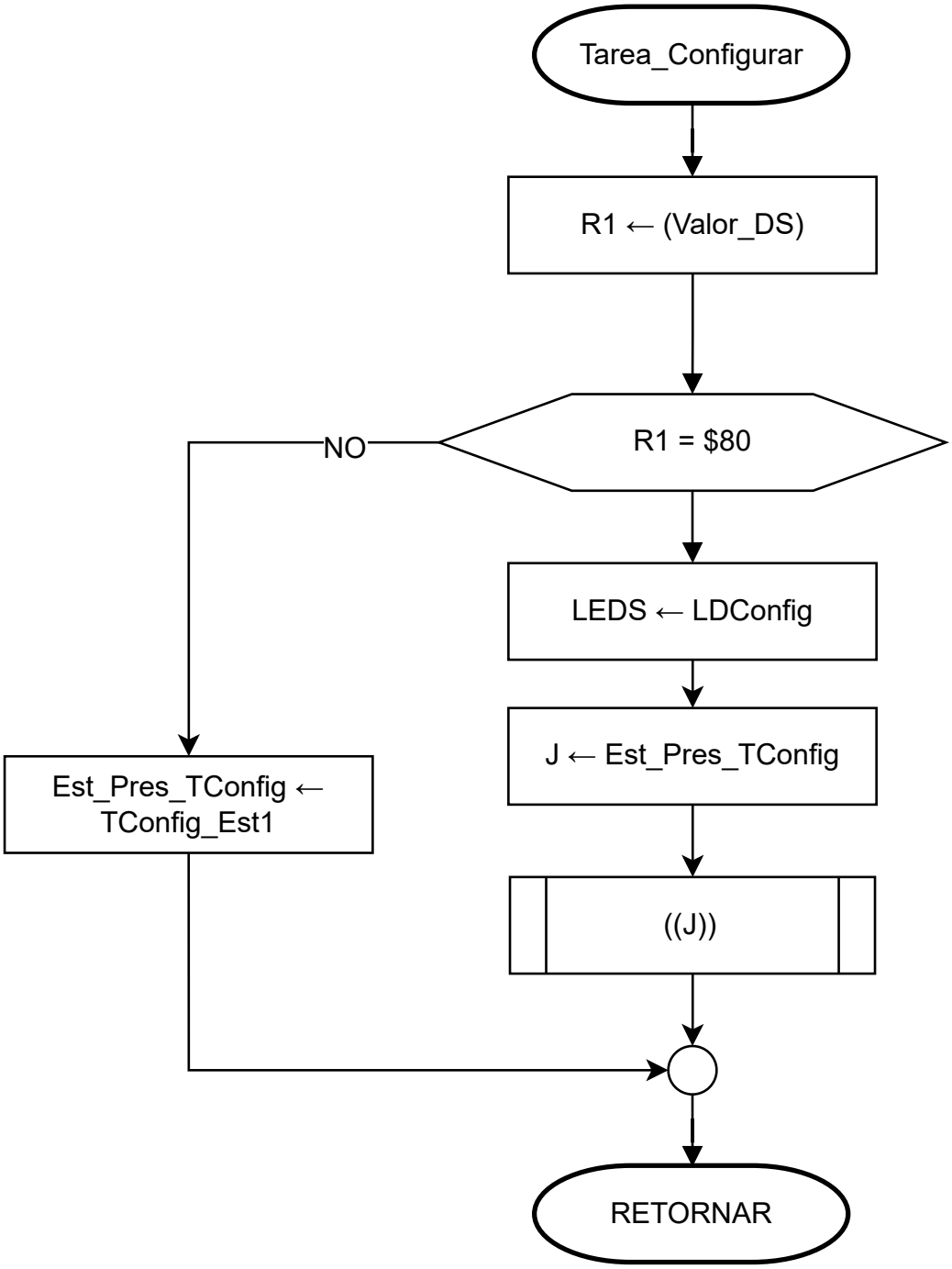

Figura 5: Diagrama de flujos de la Tarea_Configurar

2.6.2.1. TConfig_Est1

En este estado de la máquina de estados `TConfig` se cargan los punteros `MSG_L1` y `MSG_L2` con los mensajes:

- `MSG_CONFIGURAR_L1:` "MODO CONFIGURAR"
- `MSG_CONFIGURAR_L2:` "INGRESE LongOK"

Esto solicita al usuario a que ingrese un nuevo valor para `LongOK` en el teclado matricial. Además, este estado despliega el valor previamente almacenado en `LongOK` en la parte baja de los displays de 7 segmentos, mientras que la parte de alta de estos las apaga. Como se va a recibir un valor del teclado matricial, el estado hace uso de la subrutina `Borrar_NumArray` para borrar lo que se había recibido previamente a través del teclado matricial. Por último, el estado actualiza el próximo estado a `TConfig_Est2` para esperar un nuevo valor de `LongOK` del usuario. El comportamiento descrito anteriormente se muestra en la Figura 6.

Parámetros de Entrada

- `Valor_DS`: Accedido por medio de direccionamiento extendido.
- `Est_Pres_TConfig`: Accedido por medio de direccionamiento extendido.
- `LongOK`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- `Est_Pres_TConfig`: Devuelto por medio de direccionamiento a memoria RAM.
- `MSG_L1`: Devuelto por medio de direccionamiento a memoria RAM.
- `MSG_L2`: Devuelto por medio de direccionamiento a memoria RAM.
- `Banderas_2`: Devuelto por medio de direccionamiento a memoria RAM.
- `BCD1`: Devuelto por medio de direccionamiento a memoria RAM.
- `BCD2`: Devuelto por medio de direccionamiento a memoria RAM.

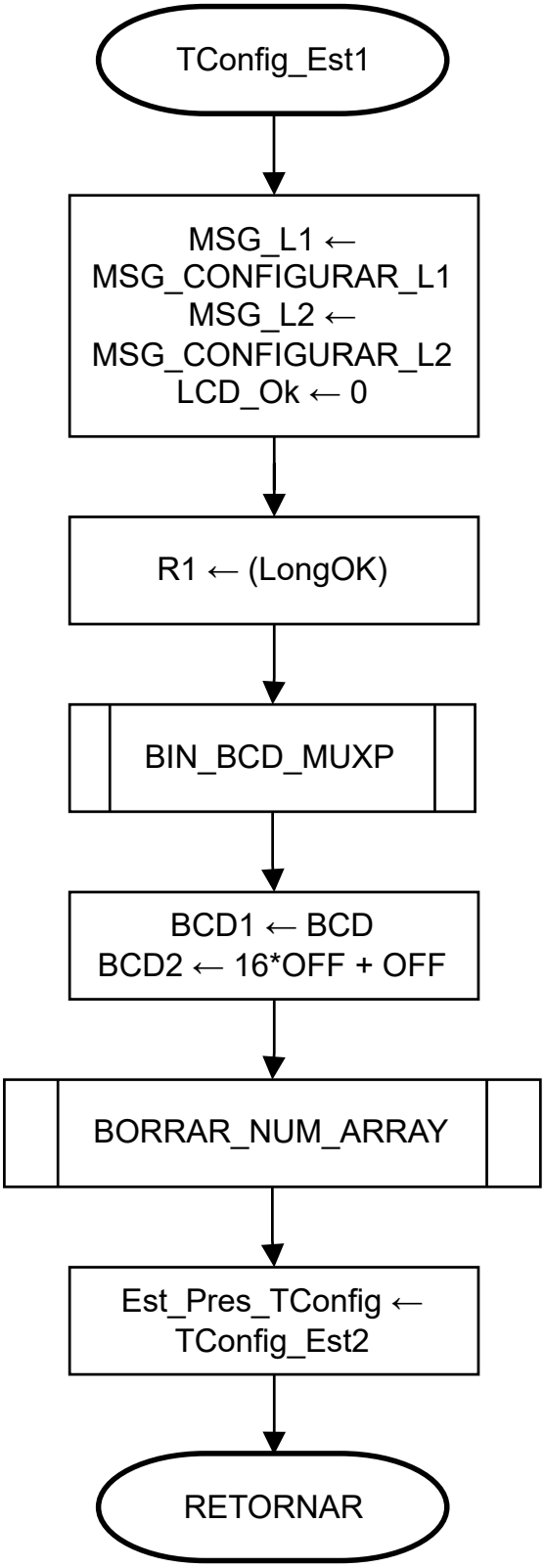

Figura 6: Diagrama de flujos de TConfig_Est1

2.6.2.2. TConfig_Est2

En este estado de la máquina de estados `TConfig`, se espera a que el usuario forme una secuencia de teclas válida. Es decir, que digite un valor para `LongOK` en el teclado matricial, y finalice con digitar la tecla Enter (#). Se hace uso de la subrutina `BCD_BIN` para convertir el valor de dos dígitos digitado en el teclado matricial para convertirlo a un valor binario. Este valor es guardado temporalmente en la variable `ValorLong`, y se verifica si el valor digitado es un valor válido de longitud. La condición a satisfacer para que una longitud digitada sea válida se muestra en (10).

$$\texttt{Lmin} < \texttt{ValorLong} < \texttt{Lmax} \tag{10}$$

Sin embargo, como el valor digitado solo puede ser de dos dígitos (`MAX_TCL` $= 2$), no es necesario realizar la comparación con `Lmax`, ya que tiene valor 99, siendo ese el máximo valor que se puede formar con dos dígitos en BCD, implicando que no se puede digitar un valor mayor a este desde el teclado matricial de la Dragon 12+. En caso de que la longitud digitada haya sido inválida, se borra Num_Array por medio de la subrutina `Borrar_NumArray`. Por lo contrario, se convierte nuevamente el valor a BCD haciendo uso de la subrutina `BIN_BCD_MUXP`, y se despliega en la parte baja de los displays de 7 segmentos, mientras que la parte alta de los mismos se apaga. Por último, se actualiza el valor de `LongOK` por medio del valor digitado en el teclado matricial, siempre y cuando este haya sido válido. El comportamiento descrito anteriormente se muestra en la Figura 7.

Parámetros de Entrada

- `Banderas_1`: Accedido por medio de direccionamiento extendido.
- `LongOK`: Accedido por medio de direccionamiento extendido.
- `ValorLong`: Accedido por medio de direccionamiento extendido.
- `Num_Array`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- `Est_Pres_TConfig`: Devuelto por medio de direccionamiento a memoria RAM.
- `MSG_L1`: Devuelto por medio de direccionamiento a memoria RAM.
- `MSG_L2`: Devuelto por medio de direccionamiento a memoria RAM.
- `BCD1`: Devuelto por medio de direccionamiento a memoria RAM.
- `BCD2`: Devuelto por medio de direccionamiento a memoria RAM.

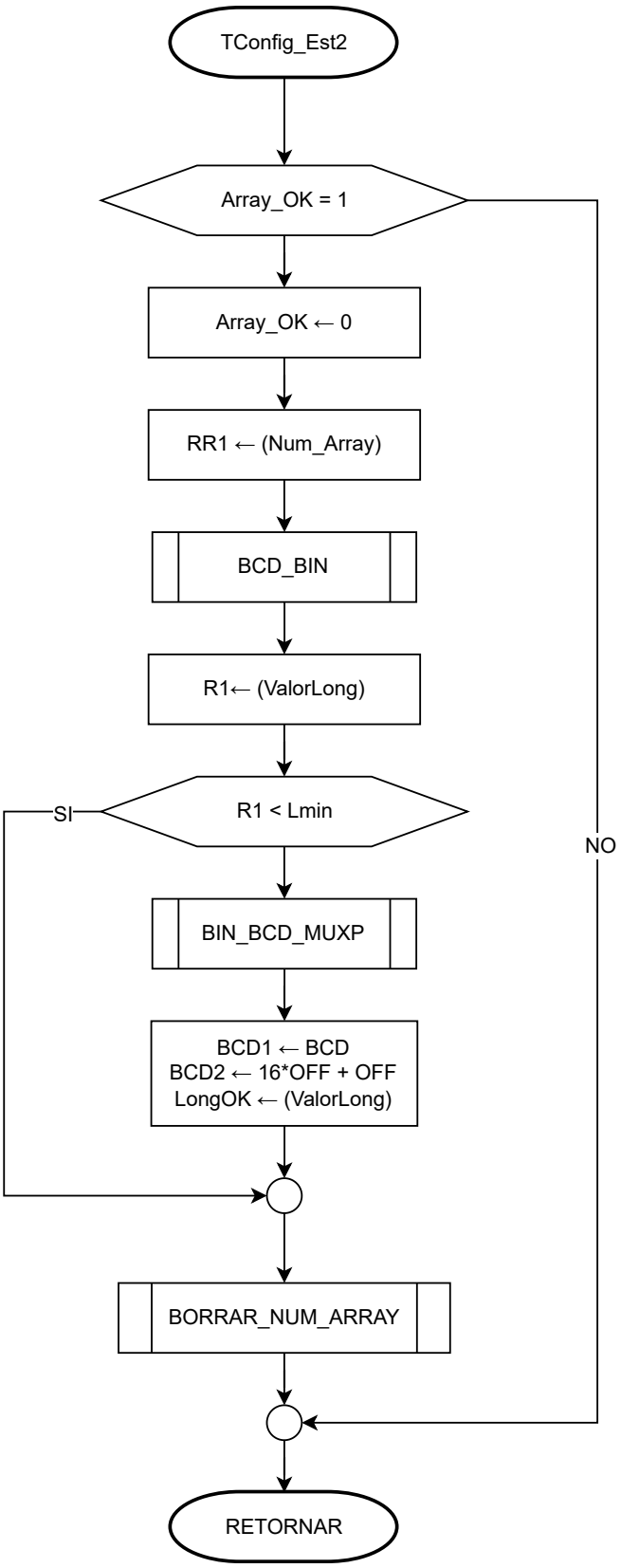

Figura 7: Diagrama de flujos de TConfig_Est2

2.6.3. Tarea_Modo_SELECCIONAR

Esta tarea evalúa el valor actual de los dip-switches almacenado en la variable `Valor_DS`, y si el usuario seleccionó el modo SELECCIONAR del Selector 623 (`Valor_DS` = $00), actualiza el patrón desplegado en los LEDS de la tarjeta Dragon12+ correspondiete a este modo de operación, correspondiente a `LDSelect`. Además de esto, esta tarea implementa la máquina de estados `TSelec`, la cual es encargada de leer los sensores S1 y S2, evaluar si los parámetros cinemáticos de las barras que transcurren la siderúrgica son válidos, y rocía el centro longitudinal de dichas barras. Además, despliega mensajes al usuario para informar el estado actual del proceso de selección de barras. Para ello, la tarea salta a la subrutina correspondiente al estado actual de la máquina de estados, almacenado en ela variable `Est_Pres_TSelec`. Si el valor de `Valor_DS` no es el correspondiente al modo SELECCIONAR del Selector 623, la tarea carga el estado inicial de la máquina de estados descrita anteriormente. El comportamiento descrito anteriormente se muestra en la Figura 8.

Parámetros de Entrada

- `Valor_DS`: Accedido por medio de direccionamiento extendido.
- `Est_Pres_TSelec`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- `LEDS`: Devuelto por medio de direccionamiento a memoria RAM.
- `Est_Pres_TSelec`: Devuelto por medio de direccionamiento a memoria RAM.

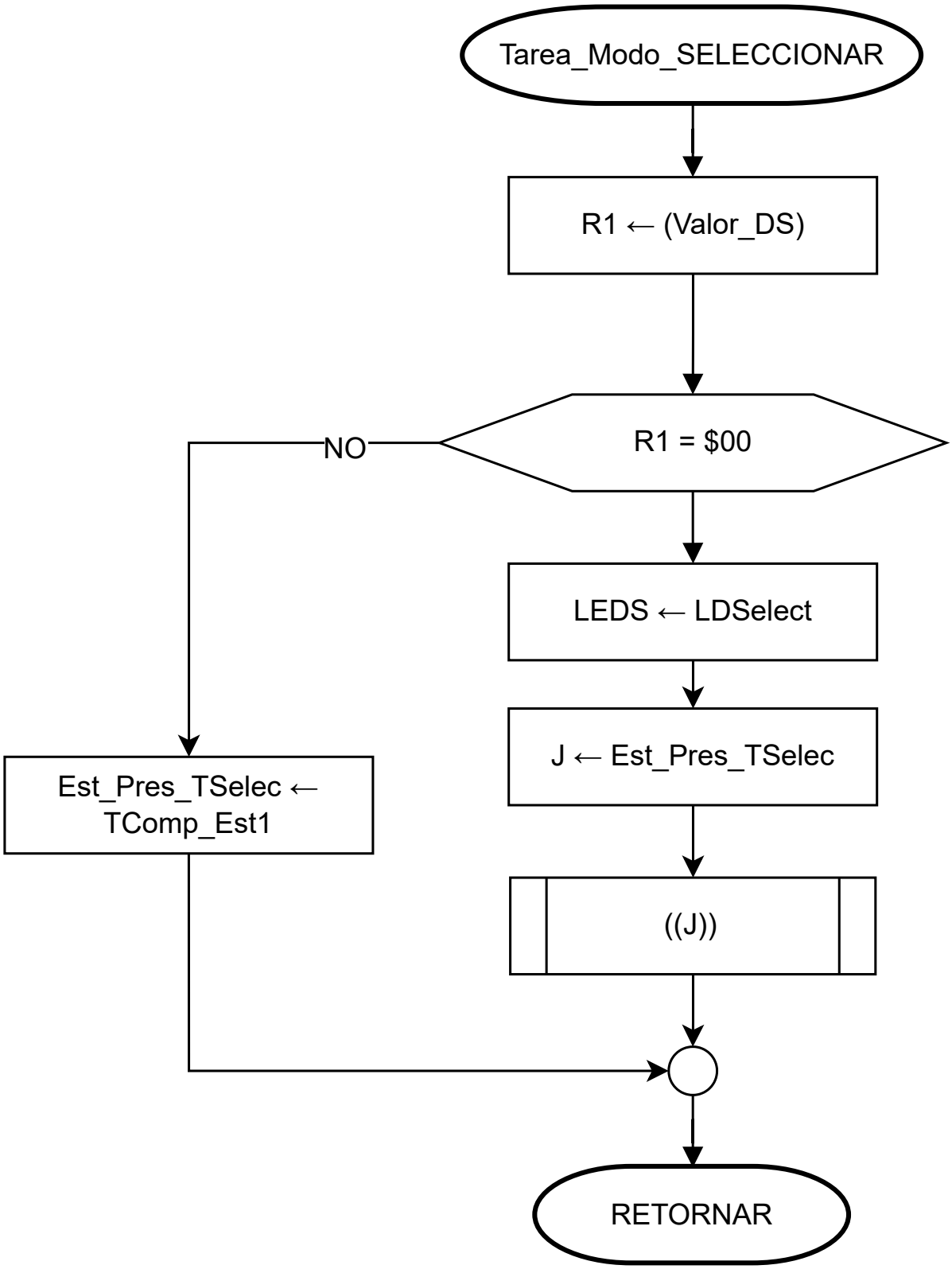

Figura 8: Diagrama de flujos de Tarea_Modo_SELECCIONAR

2.6.3.1. TComp_Est1

En este estado la máquina de estados `TSelec` se cargan los punteros `MSG_L1` y `MSG_L2` con los mensajes:

- `MSG_SELECCIONAR_L1:` "MODO SELECCIONAR"
- `MSG_CONFIGURAR_L2:` "ESPERANDO S1..."

Este mensaje le indica al usuario que se está esperando la activación del sensor S1 del Selector 623. Además de esto, se apagan los displays de 7 segmentos y se actualiza el estado para que el próximo estado sea `TComp_Est2` para esperar la activación del sensor S1. Esto se hace por medio de la variable de estado `Est_Pres_TSelec`. El comportamiento descrito anteriormente se muestra en la Figura 9.

Parámetros de Entrada

- `Banderas_2`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- `Est_Pres_TSelec`: Devuelto por medio de direccionamiento a memoria RAM.
- `MSG_L1`: Devuelto por medio de direccionamiento a memoria RAM.
- `MSG_L2`: Devuelto por medio de direccionamiento a memoria RAM.
- `BCD1`: Devuelto por medio de direccionamiento a memoria RAM.
- `BCD2`: Devuelto por medio de direccionamiento a memoria RAM.

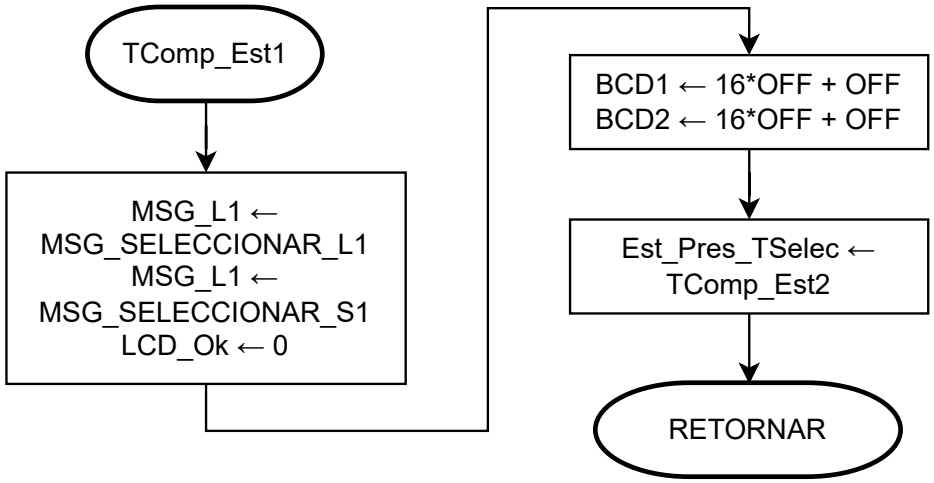

Figura 9: Diagrama de flujos de TComp_Est1

2.6.3.2. TComp_Est2

En este estado de la máquina de estados `TSelec` se espera a la activación del sensor S1. Una vez se activa el sensor S1, se cargan los punteros `MSG_L1` y `MSG_L2` con los mensajes:

- `MSG_SELECCIONAR_L1`: "MODO SELECCIONAR"
- `MSG_SELECCIONAR_S2`: "ESPERANDO S2..."

Este mensaje le indica al usuario que se está esperando la activación del sensor S2, con el propósito de calcular el tiempo entre la activación de los sensores y calcular la velocidad de la barra, la cual será almacenada en la variable `Velocidad`. Para ello, se carga el timer `TimerCal` con `tTimerCal` = 100, el cual corresponde a un timer base 100mS.

- **Nota:** Esto implica que los intervalos temporales entre la activación de los sensores no puede sobrepasar los 10 s, ya que por lo contrario, se obtendrían resultados incorrectos.

Se actualiza el próximo estado para que sea `TComp_Est3` para esperar la activación del sensor S2. Esto se hace por medio de la variable de estado `Est_Pres_TSelec`. El comportamiento descrito anteriormente se muestra en la Figura 10.

Parámetros de Entrada

- `Banderas_1`: Accedido por medio de direccionamiento extendido.
- `Banderas_2`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- `Est_Pres_TSelec`: Devuelto por medio de direccionamiento a memoria RAM.
- `MSG_L1`: Devuelto por medio de direccionamiento a memoria RAM.
- `MSG_L2`: Devuelto por medio de direccionamiento a memoria RAM.
- `TimerCal`: Devuelto por medio de direccionamiento a memoria RAM.

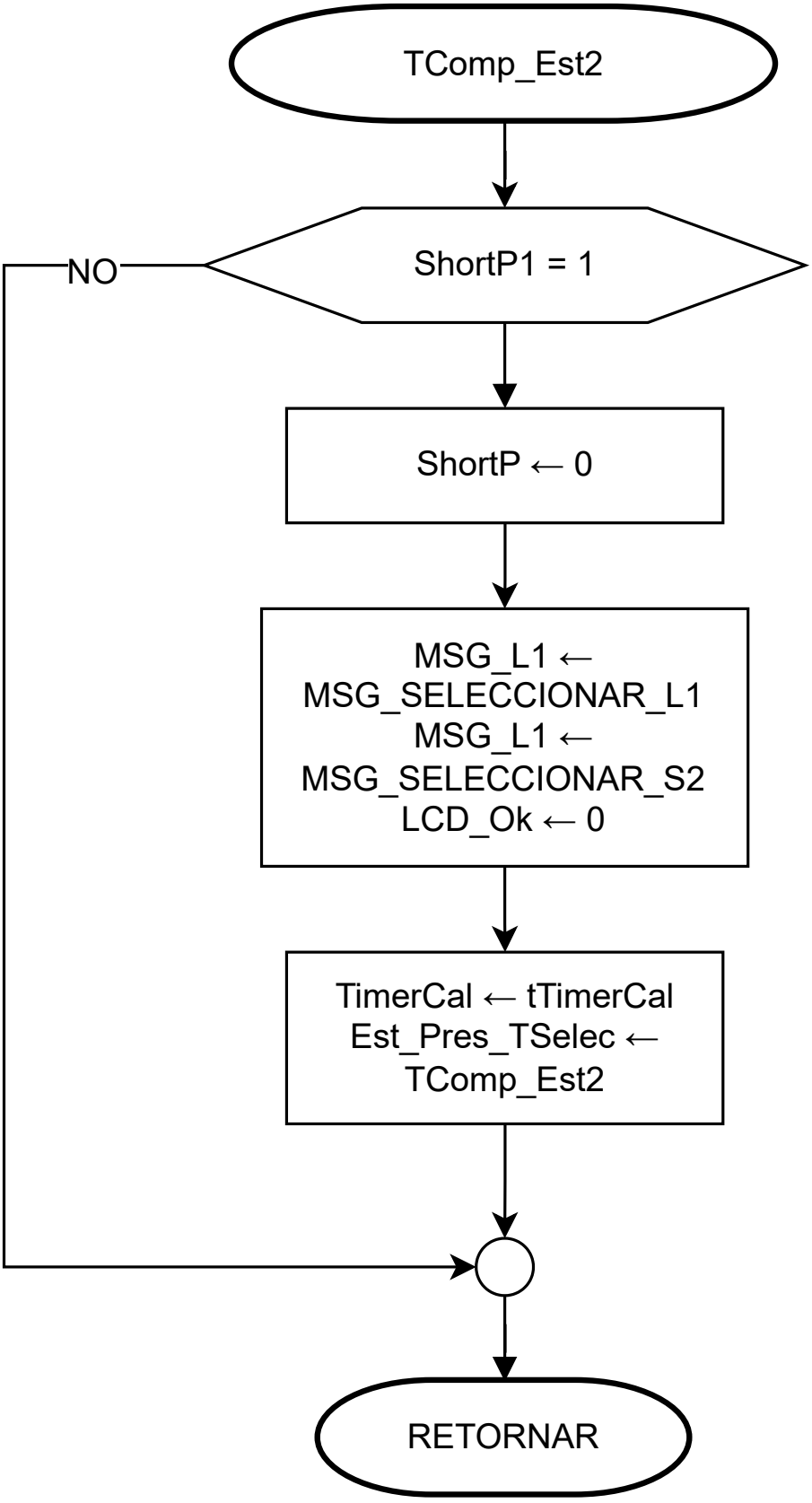

Figura 10: Diagrama de flujos de TComp_Est2

2.6.3.3. TComp_Est3

En este estado de la máquina de estados `TSelec` se espera a la activación del sensor S2. Una vez se activa el sensor S2, se cargan los punteros `MSG_L1` y `MSG_L2` con los mensajes:

- `MSG_SELECCIONAR_L1`: "MODO SELECCIONAR"
- `MSG_SELECCIONAR_BARRA`: "ESPERA FIN BARRA"

Este mensaje le indica al usuario que se está esperando nuevamente a la activación del sensor S2. Esto es debido a que, la próxima vez que el sensor S2 genere un pulso corto, se detectará el fin de la barra, y por tanto, se podrá calcular la longitud de la misma a partir de los intervalos temporales entre activación de sensores obtenidos, y la velocidad calculada. Una vez que el sensor S2 se active, se calcula el intervalo temporal entre la activación de lo sensores S1 y S2 por medio de la ecuación (11).

$$\texttt{DeltaT} = \texttt{tTimerCal} - \texttt{TimerCal} \tag{11}$$

Se actualiza el próximo estado para que sea `TComp_Est3` para esperar nuevamente la activación del sensor S2 para detectar el fin de la barra de aluminio. Esto se hace por medio de la variable de estado `Est_Pres_TSelec`. El comportamiento descrito anteriormente se muestra en la Figura 11.

Parámetros de Entrada

- `Banderas_1`: Accedido por medio de direccionamiento extendido.
- `Banderas_2`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- `Est_Pres_TSelec`: Devuelto por medio de direccionamiento a memoria RAM.
- `MSG_L1`: Devuelto por medio de direccionamiento a memoria RAM.
- `MSG_L2`: Devuelto por medio de direccionamiento a memoria RAM.
- `TimerCal`: Devuelto por medio de direccionamiento a memoria RAM.

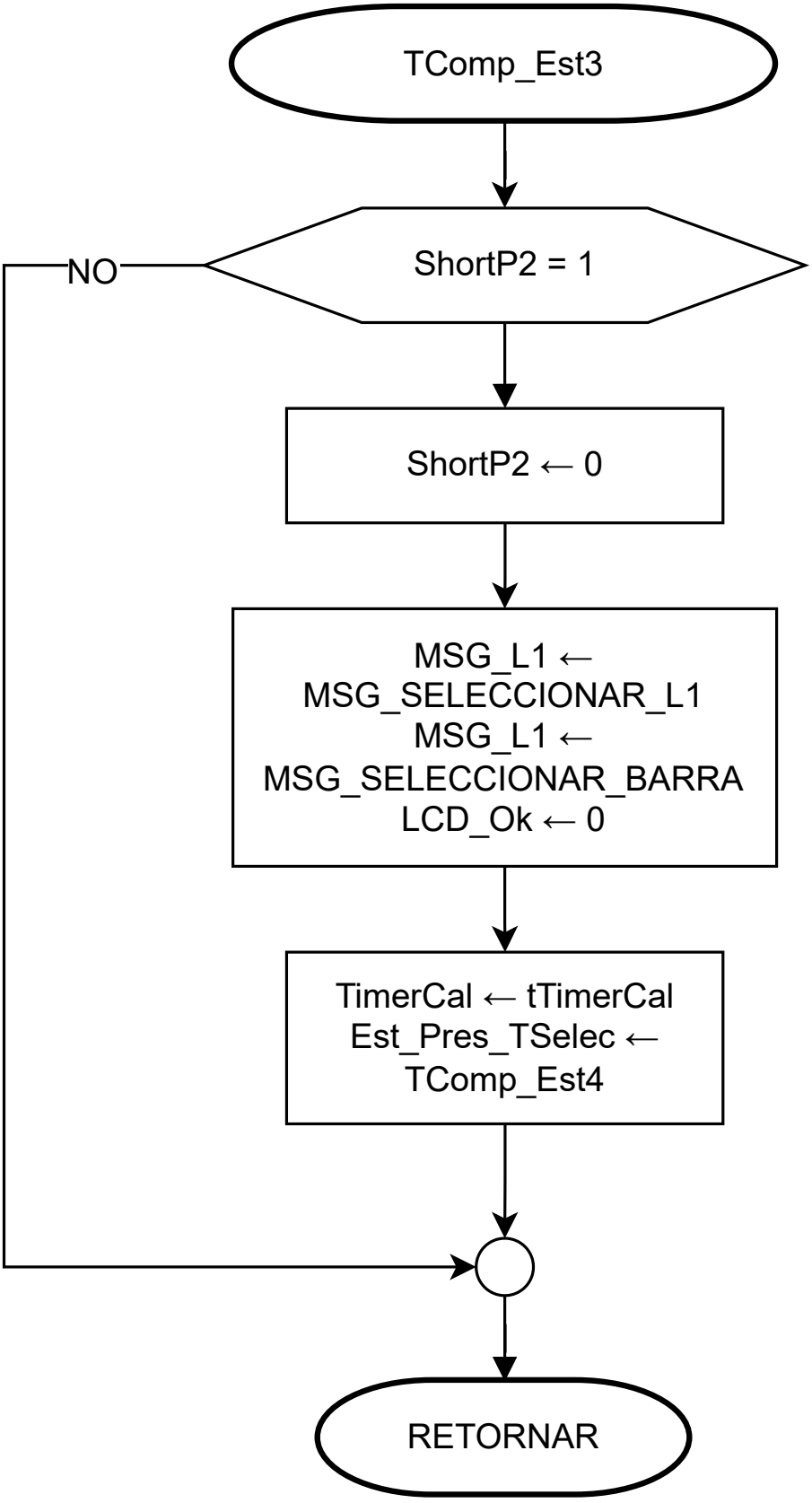

Figura 11: Diagrama de flujos de TComp_Est3

2.6.3.4. TComp_Est4

En este estado de la máquina de estados `TSelec` se espera a la segunda activación del sensor S2. Una vez esta activación sucede, se realiza el cálculo de las variables correspondientes por medio de la subrutina `Calcula`, y se verifica si la longitud y velocidad calcualadas son válidas. Si estos parámetros son válidos, se despliega un mensaje para indicar que se está calculando las variables. Por lo contrario, se despliegan mensajes indicando alguno de los dos parámetros que fueron inválidos, se borran todos los parámetros calculados, se despliegan guiones en los displays, y se pasa al estado `TComp_Est6` para retener el mensaje de error de acuerdo `TimerError`. Si los parámetros fueron válidos, se pasa al estado `TComp_Est5` para esperar el despliegue de los datos calculados en los displays de 7 segmentos, de acuerdo a `TimerPant`. Se actualiza el próximo estado por medio de la variable de estado `Est_Pres_TSelec`. El comportamiento descrito anteriormente se muestra en la Figura 12.

Parámetros de Entrada

- `Banderas_1`: Accedido por medio de direccionamiento extendido.
- `Banderas_2`: Accedido por medio de direccionamiento extendido.
- `TimerRociador`: Devuelto por medio de direccionamiento a memoria RAM.
- `TimerPant`: Devuelto por medio de direccionamiento a memoria RAM.
- `TimerFinPant`: Devuelto por medio de direccionamiento a memoria RAM.
- `Velocidad`: Devuelto por medio de direccionamiento a memoria RAM.
- `Longitud`: Devuelto por medio de direccionamiento a memoria RAM.

Parámetros de Salida

- `Est_Pres_TSelec`: Devuelto por medio de direccionamiento a memoria RAM.
- `MSG_L1`: Devuelto por medio de direccionamiento a memoria RAM.
- `MSG_L2`: Devuelto por medio de direccionamiento a memoria RAM.
- `DeltaT`: Devuelto por medio de direccionamiento a memoria RAM.
- `TimerRociador`: Devuelto por medio de direccionamiento a memoria RAM.
- `TimerPant`: Devuelto por medio de direccionamiento a memoria RAM.
- `TimerFinPant`: Devuelto por medio de direccionamiento a memoria RAM.
- `Velocidad`: Devuelto por medio de direccionamiento a memoria RAM.
- `Longitud`: Devuelto por medio de direccionamiento a memoria RAM.

- `BCD1`: Devuelto por medio de direccionamiento a memoria RAM.
- `BCD2`: Devuelto por medio de direccionamiento a memoria RAM.
- `TimerError`: Devuelto por medio de direccionamiento a memoria RAM.

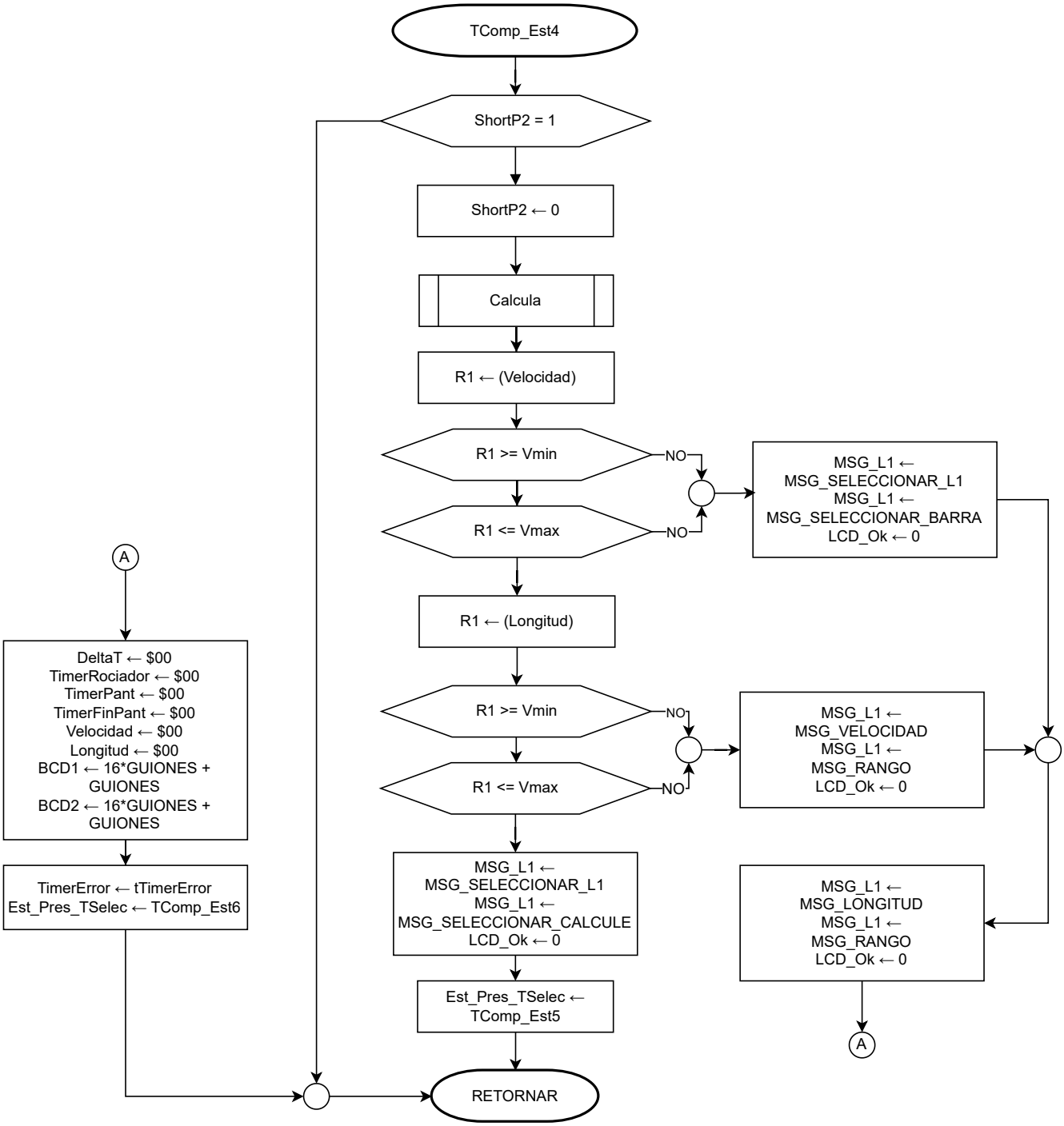

Figura 12: Diagrama de flujos de TComp_Est4

2.6.3.5. TComp_Est5

En este estado se espera que llegue el momento de desplegar los parámetros cinemáticos de la barra obtenidos de la subrutina `Calcula` en los displays, de acuerdo a `TimerPant`. Una vez finaliza este timer, se despliegan los mensajes:

- `MSG_SELECCIONAR_L1`: "MODO SELECCIONAR"
- `MSG_SELECCIONAR_VL`: "VELOC LONG"

En la parte baja de los displays se muestra el valor de `Longitud`, y en la parte baja `Velocidad`. Para ello, se hace uso de la subrutina `BIN_BCD_MUXP` para obtener los valores en BCD de cada uno de estos valores binarios necesarios para desplegar los valores en los displays. Además, se actualiza el próximo estado para que sea `TComp_Est7` y esperar a que `TimerRociador` termnie y activar el rociador por medio del microrele. Se actualiza el próximo estado por medio de la variable de estado `Est_Pres_TSelec`. El comportamiento descrito anteriormente se muestra en la Figura 13.

Parámetros de Entrada

- `Banderas_2`: Accedido por medio de direccionamiento extendido.
- `TimerPant`: Devuelto por medio de direccionamiento a memoria RAM.
- `Velocidad`: Devuelto por medio de direccionamiento a memoria RAM.
- `Longitud`: Devuelto por medio de direccionamiento a memoria RAM.

Parámetros de Salida

- `Est_Pres_TSelec`: Devuelto por medio de direccionamiento a memoria RAM.
- `MSG_L1`: Devuelto por medio de direccionamiento a memoria RAM.
- `MSG_L2`: Devuelto por medio de direccionamiento a memoria RAM.
- `BCD1`: Devuelto por medio de direccionamiento a memoria RAM.
- `BCD2`: Devuelto por medio de direccionamiento a memoria RAM.

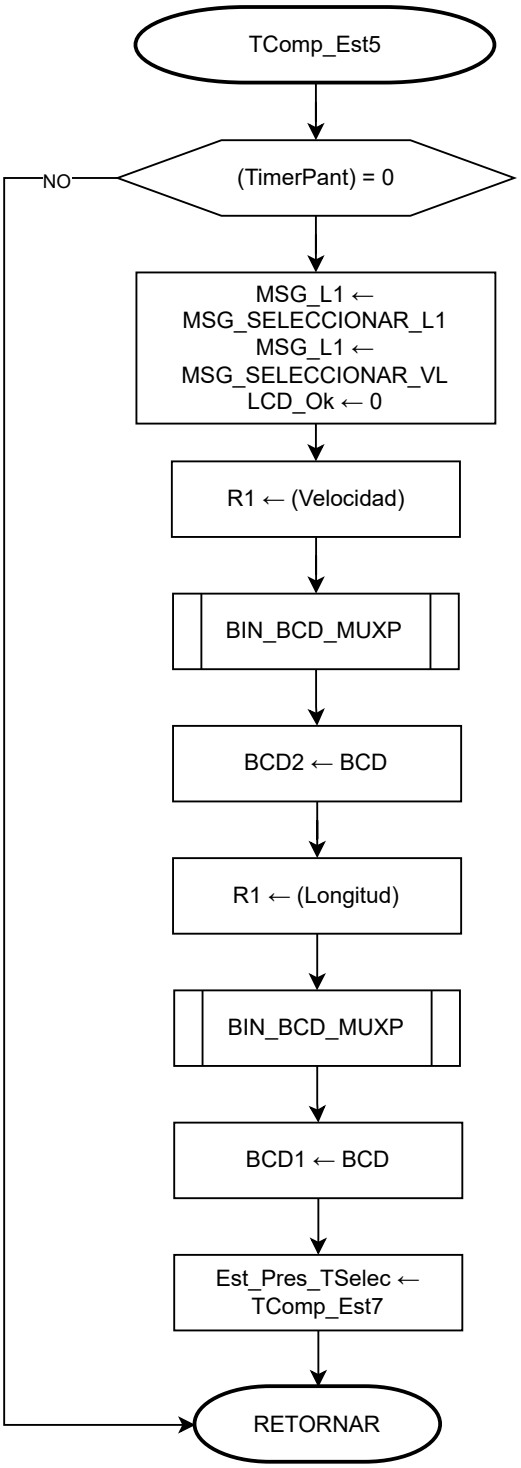

Figura 13: Diagrama de flujos de TComp_Est5

2.6.3.6. TComp_Est6

Este estado de la máquina de estados `TSelec` retiene un mensaje de error en la LCD y en los displays (guiones). Para ello, se espera a que `TimerError` llegue a cero, el cual fue previamente cargado con `tTimerError` en caso de que se hubiese tenido un parámetro inválido en el estado `TComp_Est4`. Una vez `TimerError` finaliza, se retorna al estado `TComp_Est1` para reiniciar la máquina de estados, y se inicie el proceso de selección de una nueva barra. Se actualiza el próximo estado por medio de la variable de estado `Est_Pres_TSelec`. El comportamiento descrito anteriormente se muestra en la Figura 14.

Parámetros de Entrada

- `TimerError`: Devuelto por medio de direccionamiento a memoria RAM.

Parámetros de Salida

- `Est_Pres_TSelec`: Devuelto por medio de direccionamiento a memoria RAM.

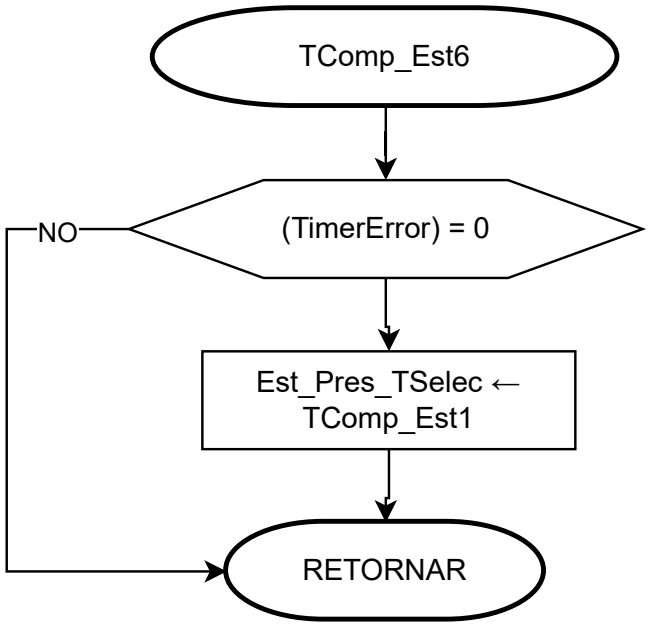

Figura 14: Diagrama de flujos de TComp_Est6

2.6.3.7. TComp_Est7

Este estado de la máquina de estados `TSelec` espera a que finalice `TimerRociador` para activar el rociador, lo cual es logrado al activar el micrroele conectado a PE2 en la tarjeta Dragon12+. Una vez finaliza `TimerRociador`, se habilita el rociador, se carga `TimerShot` con `tTimerShot` para deshabilitar el rociador en el siguiente estado, y se pasa al estado `TComp_Est8` para esperar a que `TimerShot` finalice. Se actualiza el próximo estado por medio de la variable de estado `Est_Pres_TSelec`. El comportamiento descrito anteriormente se muestra en la Figura 15.

Parámetros de Entrada

- `TimerRociador`: Devuelto por medio de direccionamiento a memoria RAM.

Parámetros de Salida

- `Est_Pres_TSelec`: Devuelto por medio de direccionamiento a memoria RAM.
- `PortRele`: Accedido por medio de direccionamiento extendido.
- `TimerShot`: Devuelto por medio de direccionamiento a memoria RAM.

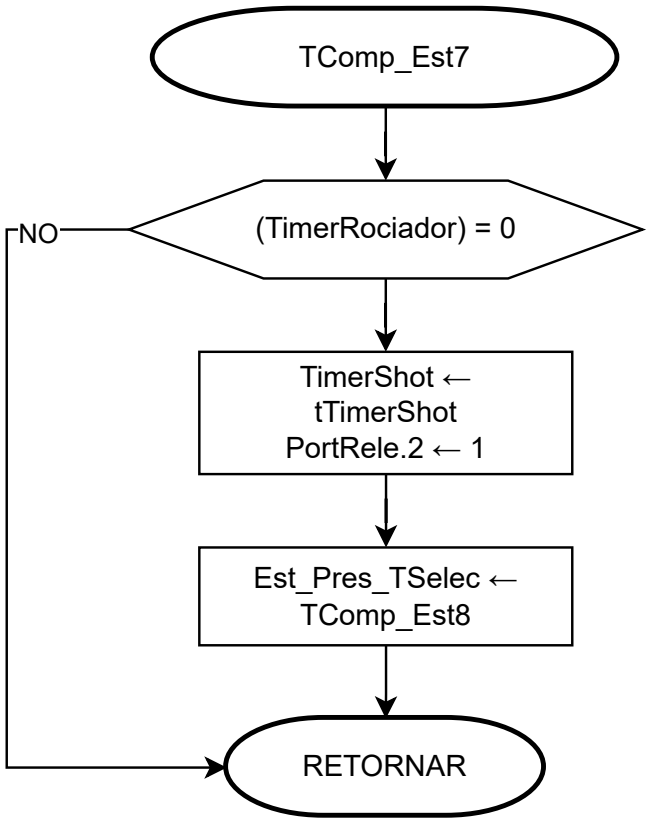

Figura 15: Diagrama de flujos de TComp_Est7

2.6.3.8. TComp_Est8

En este estado de la máquina de estados `TSelec` se espera a que `TimerShot` finalice para deshabilitar el rociador. Una vez finaliza `TimerShot`, se deshabilita el rociador, y se espera a que `TimerFinPant` finalice para parar de desplegar los mensajes en la LCD y los displays. Una vez que `TimerFinPant` finalice, se carga a `Est_Pres_TSelec` con `TComp_Est1` para reiniciar el funcionamiento de la máquina de estados, haciendola arrancar en el estado inicial para procesar otra barra. Se actualiza el próximo estado por medio de la variable de estado `Est_Pres_TSelec`. El comportamiento descrito anteriormente se muestra en la Figura 16.

Parámetros de Entrada

- `TimerShot`: Devuelto por medio de direccionamiento a memoria RAM.
- `TimerFinPant`: Devuelto por medio de direccionamiento a memoria RAM.

Parámetros de Salida

- `Est_Pres_TSelec`: Devuelto por medio de direccionamiento a memoria RAM.
- `PortRele`: Accedido por medio de direccionamiento extendido.

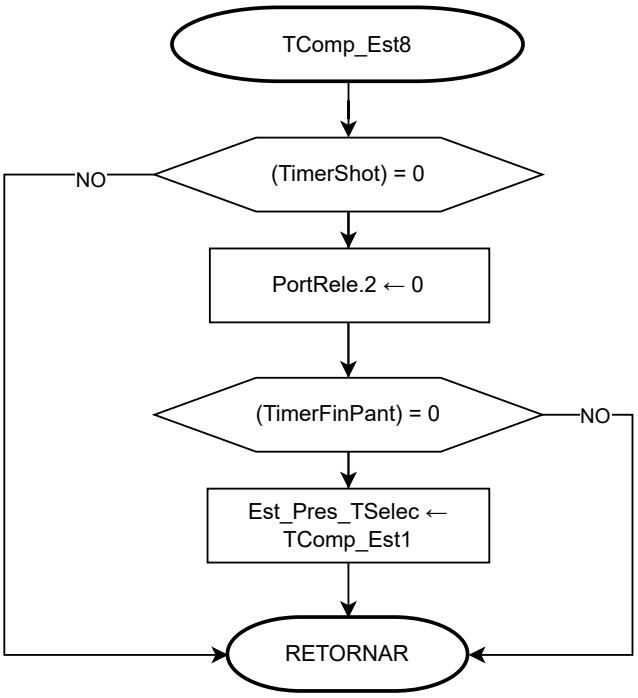

Figura 16: Diagrama de flujos de TComp_Est8

2.6.4. Tarea_Brillo

Esta tarea implementa la máquina de estados `TareaBrillo`, la cual, en base a la lectura de una conversión analógico/digital del trimmer conectado al PAD7 de la tarjeta Dragon12+, actualiza el valor de la variable `Brillo` en una escala líneal de 0 a 100. Esto se realiza con el propósito de cambiar el nivel de brillo de los displays multiplexados en `PORTB` y los LEDS. La conversión analógico/digital es configurada como se indica en la memoria de cálculo correspondiente. Para ello, se utiliza la variable de estado `Est_Pres_TBrillo`, la cual contiene el próximo estado al cual saltará la máquina de estados en `Tarea_Brillo`. El comportamiento descrito anteriormente se muestra en la Figura 17.

Parámetros de Entrada

- `Est_Pres_TBrillo`: Devuelto por medio de direccionamiento a memoria RAM.

Parámetros de Salida

- Esta tarea no tiene parámetros de salida.

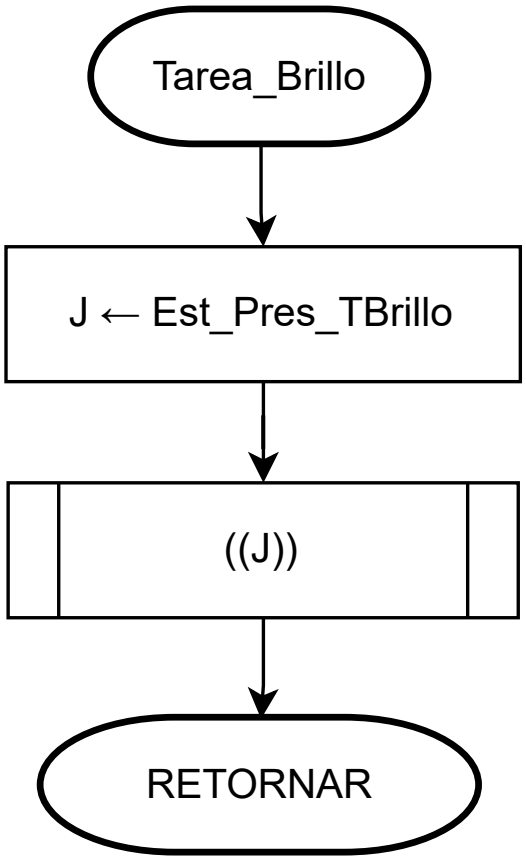

Figura 17: Diagrama de flujos de Tarea_Brillo

2.6.4.1. TareaBrillo_Est1

En este estado se carga `TimerBrillo` con `tTimerBrillo` para generar un ciclo de conversión cada 400 ms. Además, se carga el próximo estado `TareaBrillo_Est2` para esperar a que el timer recien cargado finalice e iniciar el ciclo de conversión Se actualiza el próximo estado por medio de la variable de estado `Est_Pres_TBrillo`. El comportamiento descrito anteriormente se muestra en la Figura 18.

Parámetros de Entrada

- Esta tarea no tiene parámetros de entrada.

Parámetros de Salida

- `Est_Pres_TBrillo`: Devuelto por medio de direccionamiento a memoria RAM.

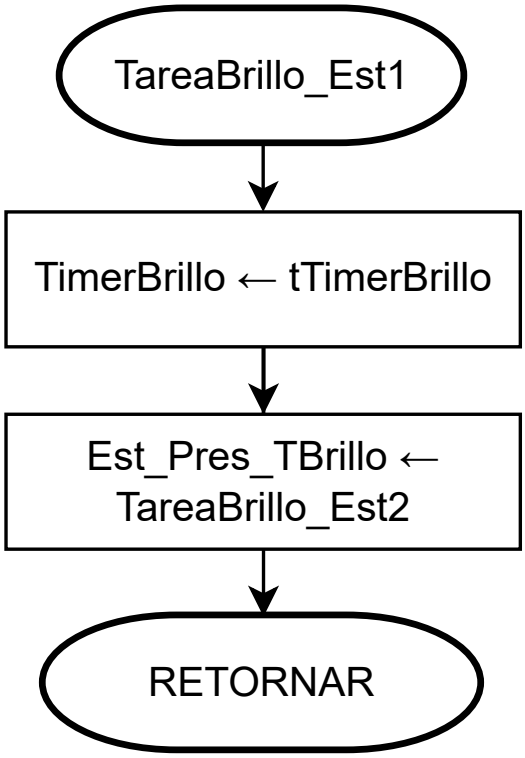

Figura 18: Diagrama de flujos de TareaBrillo_Est1

2.6.4.2. TareaBrillo_Est2

En este estado se espera a que finalice `TimerBrillo` para generar un ciclo de conversión ADC. Una vez el ciclo de conversión haya sido iniciado, se carga el próximo estado `TareaBrillo_Est3` para esperar a que finalice el ciclo de conversión, y calcular los parámetros necesarios en base a la conversión. Se actualiza el próximo estado por medio de la variable de estado `Est_Pres_TBrillo`. El comportamiento descrito anteriormente se muestra en la Figura 19.

Parámetros de Entrada

- `TimerBrillo`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- `Est_Pres_TBrillo`: Devuelto por medio de direccionamiento a memoria RAM.
- `ATD0CTL5`: Devuelto por medio de direccionamiento a memoria RAM.

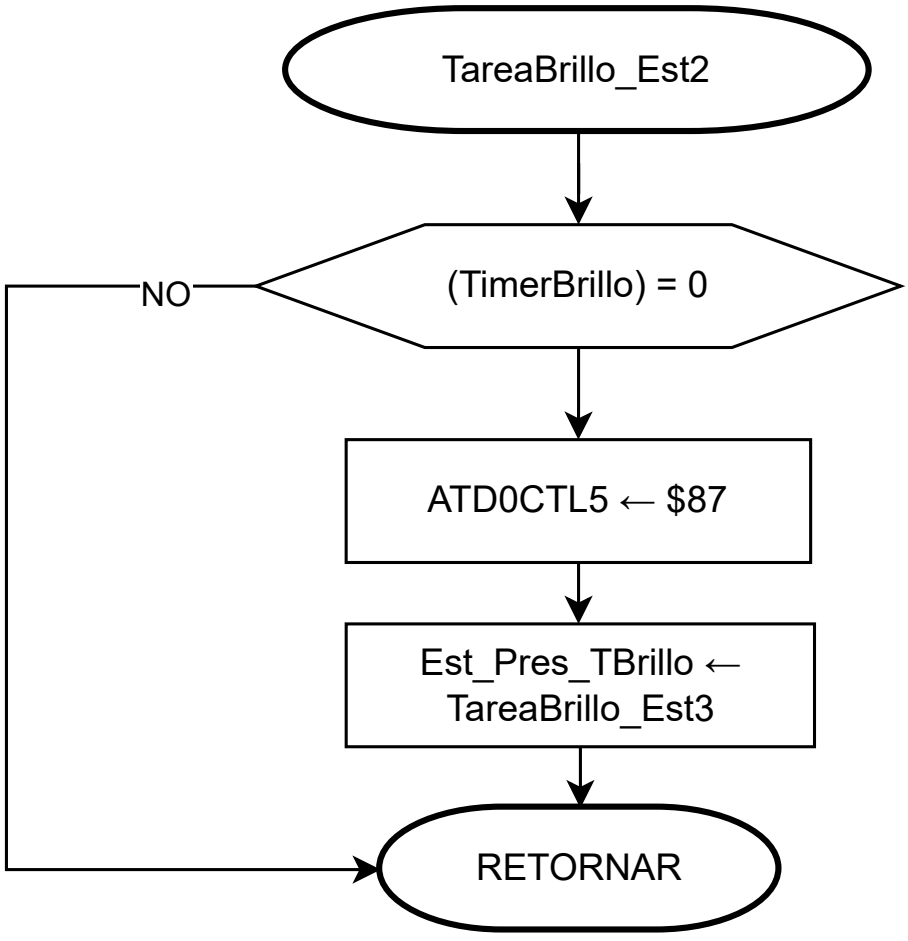

Figura 19: Diagrama de flujos de TareaBrillo_Est2

2.6.4.3. TareaBrillo_Est3

En este estado se espera a que finalice el ciclo deconversión iniciado en el estado `TareaBrillo_Est2`. Se leen las 4 conversiones realizadas en el ciclo y se promedian. Seguido a esto, se convierte el valor obtenido del ciclo de conversión que se encuentra en el rango de 0-255 a una escala de 0-100. Para ello, se utilizó la instrucción `FDIV`, por lo que se tuvo que corregir posteriormente el resultado, primero multiplicándolo por 100 y después dividiéndolo por 2^{16} debido a que los resultados de esta instrucción quedan referidos a este valor. Finalmente, se almacena el valor calculado en la variable `Brillo`, y se pasa al estado `TareaBrillo_Est1` para iniciar un nuevo ciclo de conversión. Se actualiza el próximo estado por medio de la variable de estado `Est_Pres_TBrillo`. El comportamiento descrito anteriormente se muestra en la Figura 20.

Parámetros de Entrada

- `ATD0STAT0`: Accedido por medio de direccionamiento extendido.
- `ADR00H`: Accedido por medio de direccionamiento extendido.
- `ADR01H`: Accedido por medio de direccionamiento extendido.
- `ADR02H`: Accedido por medio de direccionamiento extendido.
- `ADR03H`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- `Est_Pres_TBrillo`: Devuelto por medio de direccionamiento a memoria RAM.
- `Brillo`: Devuelto por medio de direccionamiento a memoria RAM.

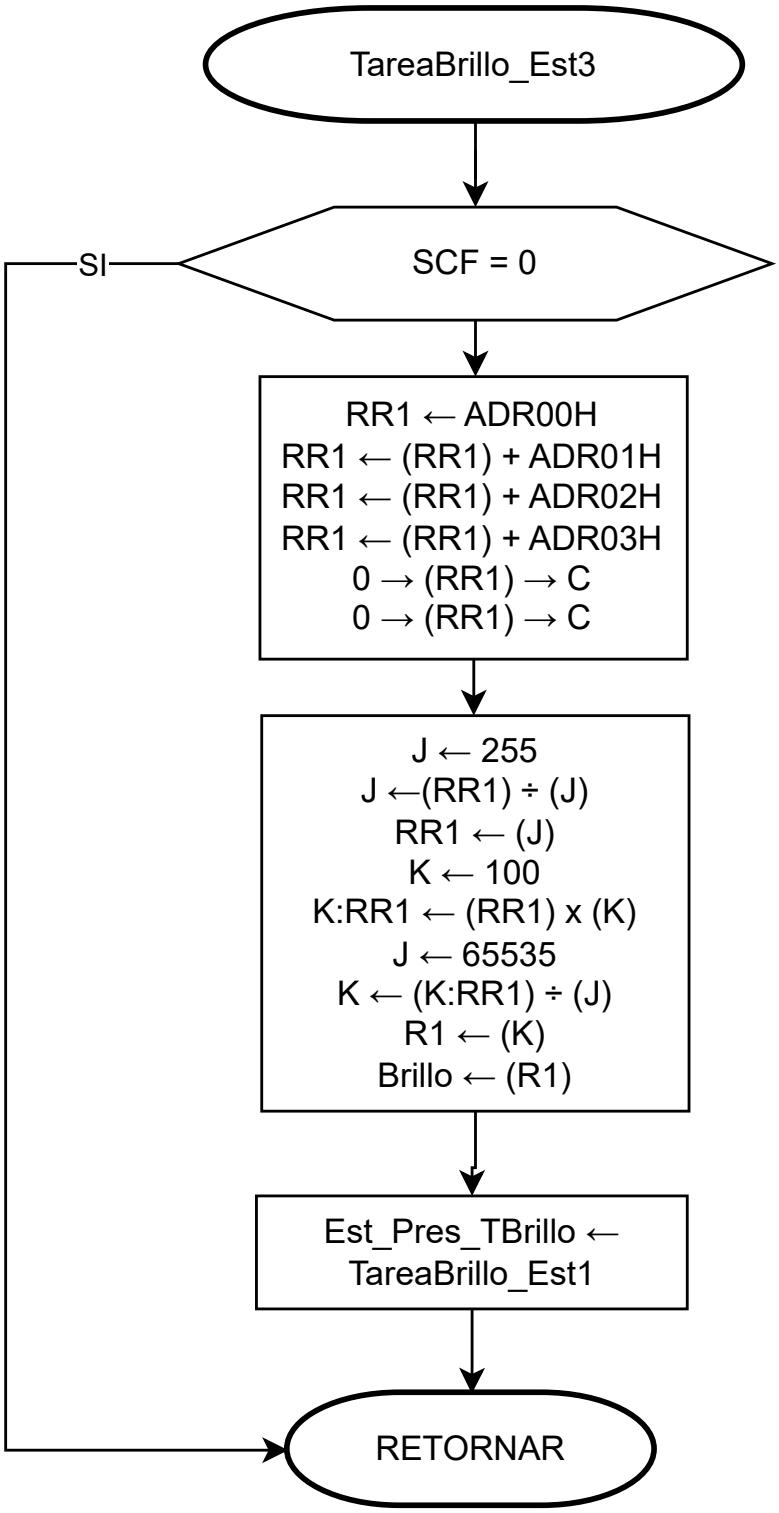

Figura 20: Diagrama de flujos de TareaBrillo_Est3

2.6.5. Tarea_Teclado

Esta tarea implementa una máquina de estados que suprime los rebotes generados en los botones del teclado matricial de la Dragon12+, y también detecta las teclas de borrado y Enter. Se actualiza el próximo estado por medio de la variable de estado `Est_Pres_TCDL`. El comportamiento descrito anteriormente se muestra en la Figura 21.

Parámetros de Entrada

- `Est_Pres_TCL`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- Esta tarea no tiene parámetros de salida

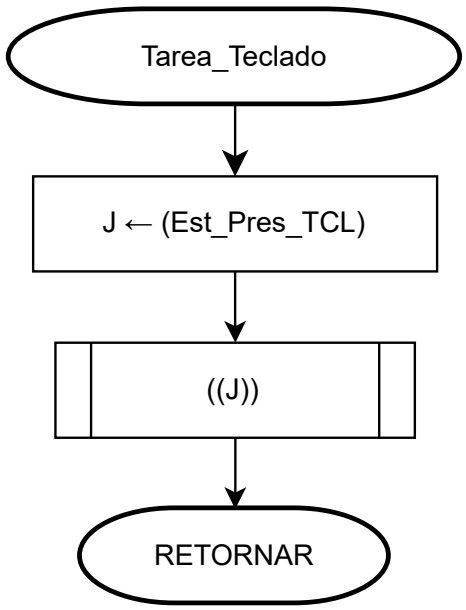

Figura 21: Diagrama de flujos de Tarea_Teclado

2.6.5.1. TareaTCL_Est1

Este estado inicia un timer para suprimir los rebotes del teclado, en caso de que se haya digitado una tecla. Se actualiza el próximo estado por medio de la variable de estado `Est_Pres_TCL`. El comportamiento descrito anteriormente se muestra en la Figura 22.

Parámetros de Entrada

- `Tecla`: Accedido por medio de direccionamiento extendido.Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- `Est_Pres_TCL`: Devuelto por medio de direccionamiento a memoria RAM.
- `Tecla_IN`: Devuelto por medio de direccionamiento a memoria RAM.
- `TimerReb_TCL`: Devuelto por medio de direccionamiento a memoria RAM.

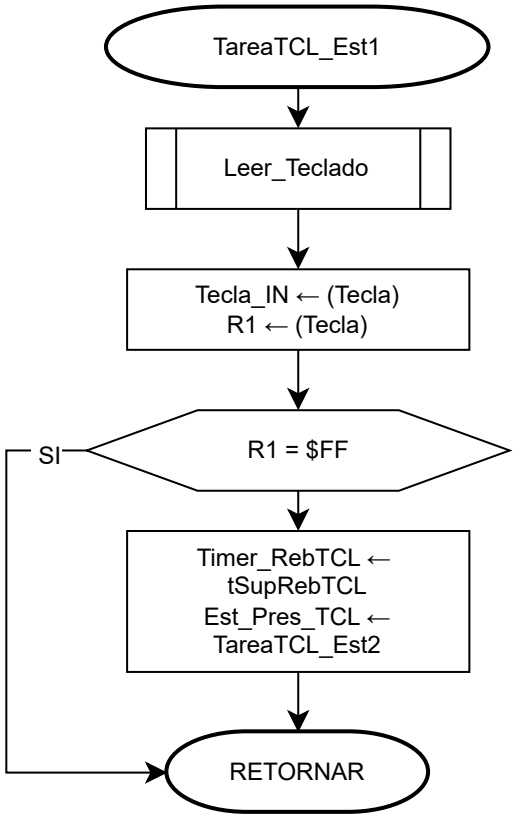

Figura 22: Diagrama de flujos de TareaTCL_Est1

2.6.5.2. TareaTCL_Est2

En este estado, se lee nuevamente el teclado para asegurarse de que la tecla presionada fue válida. En caso de que haya sido inválida, se devuelve al estado 1. Se actualiza el próximo estado por medio de la variable de estado `Est_Pres_TCL`. El comportamiento descrito anteriormente se muestra en la Figura 23.

Parámetros de Entrada

- `Tecla`: Accedido por medio de direccionamiento extendido.Accedido por medio de direccionamiento extendido.
- `Tecla_IN`: Accedido por medio de direccionamiento extendido.Accedido por medio de direccionamiento extendido.
- `TimerReb_TCL`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- `Est_Pres_TCL`: Devuelto por medio de direccionamiento a memoria RAM.

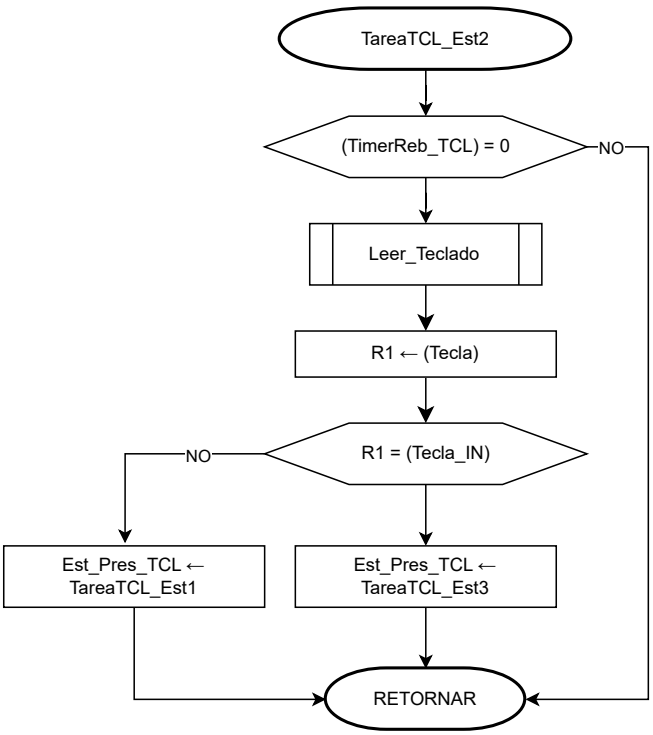

Figura 23: Diagrama de flujos de TareaTCL_Est2

2.6.5.3. TareaTCL_Est3

En este estado, se espera a que la tecla pare de ser retenida para que su valor sea almacenado en `NumArray`, acción la cual realizará el estado 4 de la máquina de estados. Se actualiza el próximo estado por medio de la variable de estado `Est_Pres_TCL`. El comportamiento descrito anteriormente se muestra en la Figura 24.

Parámetros de Entrada

- `Tecla`: Accedido por medio de direccionamiento extendido.Accedido por medio de direccionamiento extendido.
- `Tecla_IN`: Accedido por medio de direccionamiento extendido.Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- `Est_Pres_TCL`: Devuelto por medio de direccionamiento a memoria RAM.

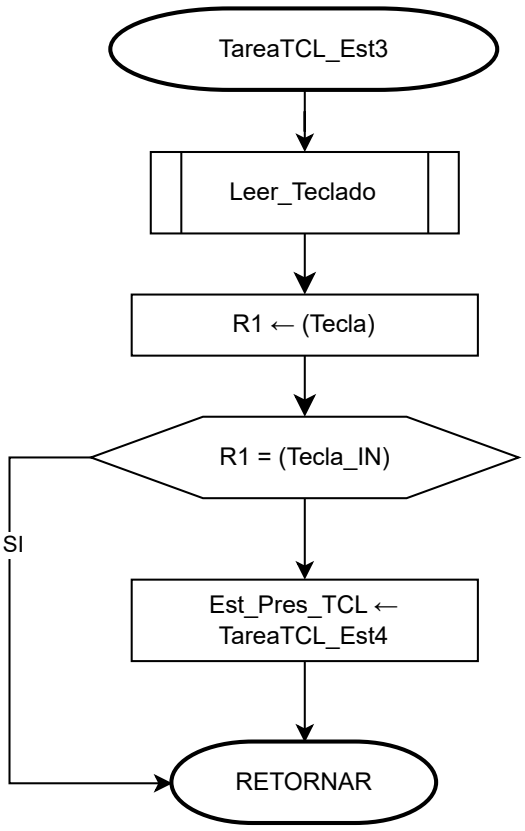

Figura 24: Diagrama de flujos de TareaTCL_Est3

2.6.5.4. TareaTCL_Est4

Este estado almacena la tecla leída en `NumArray`. En caso de que se haya presionado la tecla borrar o la tecla enter, se borra la tecla, o se confirma el ingreso de una secuencia de teclas válidas. Se actualiza el próximo estado por medio de la variable de estado `Est_Pres_TCL`. El comportamiento descrito anteriormente se muestra en la Figura 25.

Parámetros de Entrada

- `Tecla_IN`: Accedido por medio de direccionamiento extendido.
- `Cont_TCL`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- `Est_Pres_TCL`: Devuelto por medio de direccionamiento a memoria RAM.
- `Tecla_IN`: Devuelto por medio de direccionamiento a memoria RAM.
- `Cont_TCL`: Devuelto por medio de direccionamiento a memoria RAM.

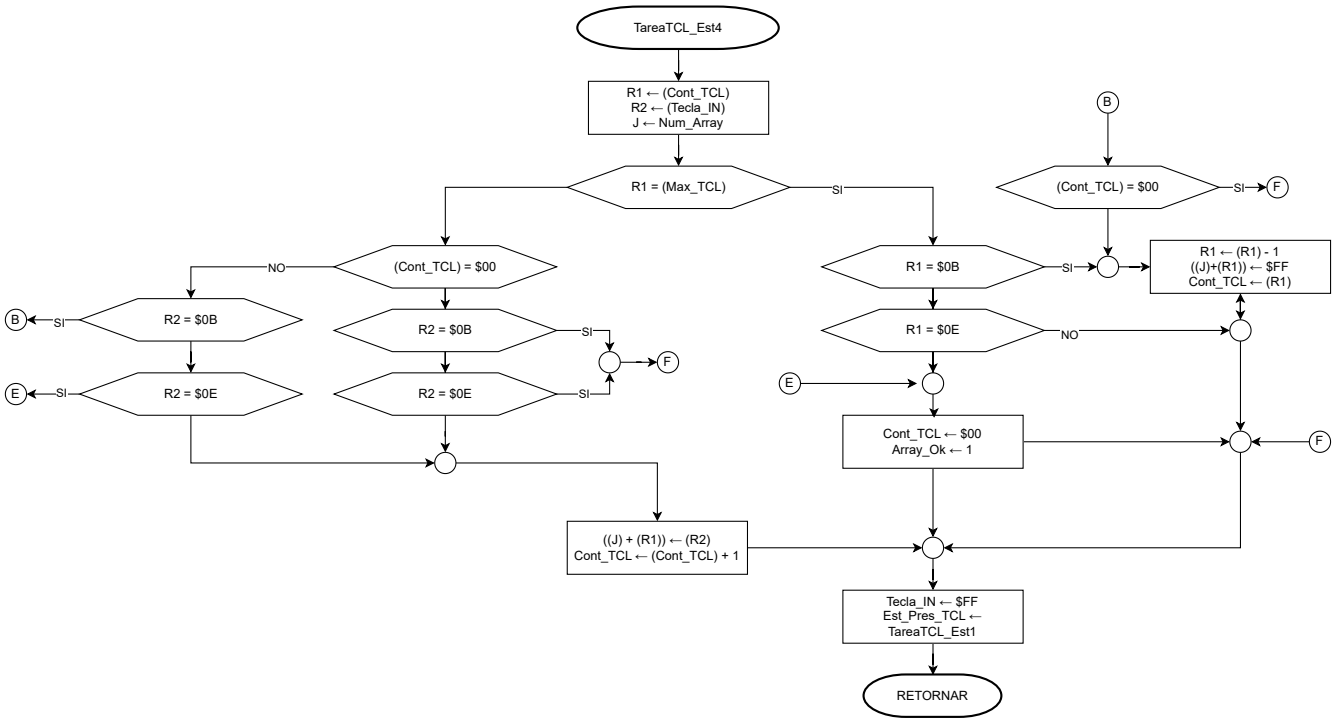

Figura 25: Diagrama de flujos de TareaTCL_Est4

2.6.6. Tarea_Led_Testigo

Esta tarea implementa la máquina de estados `LDTst`, la cual cicla el LED RGB en la tarjeta Dragon12+ entre los 3 colores que esta posee. Esto se realiza con el propósito de asegurar el correcto funcionamiento de la Máquina de Tiempos, de la cual dependen todas las máquinas de estado implementadas en este proyecto. Por tanto, si el LED RGB no está ciclando entre los colores rojo, azul, y verde, significa que la Máquina de Tiempos no está funcionando adecuadamente. Para implementar esta máquina de estados, se hace uso de la variable de estado `Est_Pres_LDTst`, la cual contiene el próximo estado al cual saltará la máquina de estados en `Tarea_Led_Testigo`. El comportamiento descrito anteriormente se muestra en la Figura 26.

Parámetros de Entrada

- Esta tarea no posee parámetros de entrada.

Parámetros de Salida

- `Est_Pres_LDTst`: Devuelto por medio de direccionamiento a memoria RAM.

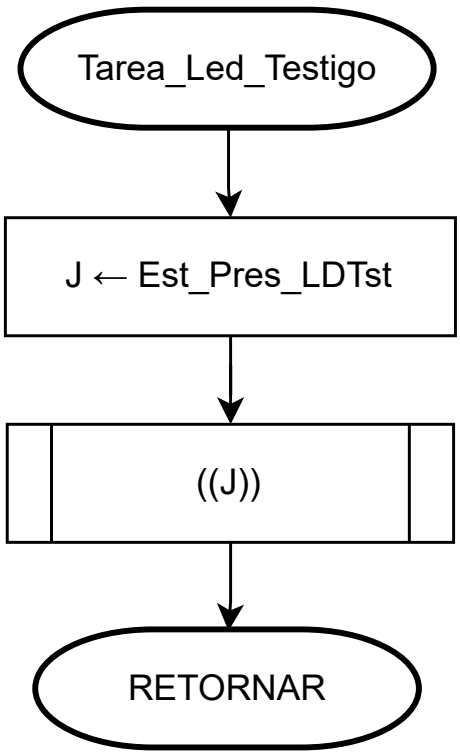

Figura 26: Diagrama de flujos de Tarea_Led_Testigo

2.6.6.1. LDTst_Est1

En este estado se espera a que finalice `Timer_LED_Testigo`. Cuando este finaliza, se apaga el LED verde y se enciende el LED azul. Se carga el estado `LDTst_Est2` para esperar nuevamente a `Timer_LED_Testigo` y después encender el LED rojo. Se actualiza el próximo estado por medio de la variable de estado `Est_Pres_LDTst`. El comportamiento descrito anteriormente se muestra en la Figura 27.

Parámetros de Entrada

- `Timer_LED_Testigo`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- `Est_Pres_LDTst`: Devuelto por medio de direccionamiento a memoria RAM.
- `PTP`: Devuelto por medio de direccionamiento a memoria RAM.
- `Timer_LED_Testigo`: Accedido por medio de direccionamiento extendido.

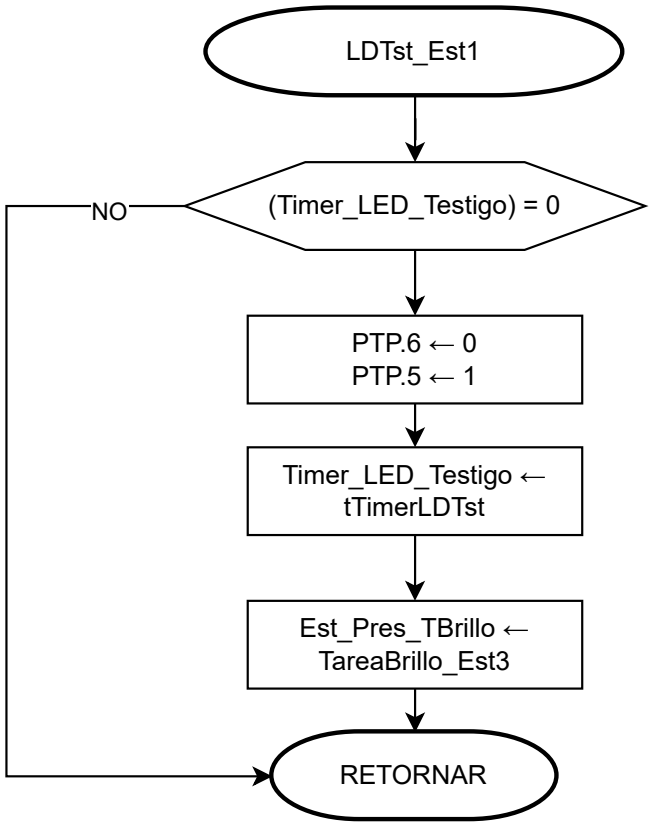

Figura 27: Diagrama de flujos de LDTst_Est1

2.6.6.2. LDTst_Est2

En este estado se espera a que finalice `Timer_LED_Testigo`. Cuando este finaliza, se apaga el LED azul y se enciende el LED rojo. Se carga el estado `LDTst_Est3` para esperar nuevamente a `Timer_LED_Testigo` y después encender el LED verde. Se actualiza el próximo estado por medio de la variable de estado `Est_Pres_LDTst`. El comportamiento descrito anteriormente se muestra en la Figura 28.

Parámetros de Entrada

- `Timer_LED_Testigo`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- `Est_Pres_LDTst`: Devuelto por medio de direccionamiento a memoria RAM.
- `PTP`: Devuelto por medio de direccionamiento a memoria RAM.
- `Timer_LED_Testigo`: Accedido por medio de direccionamiento extendido.

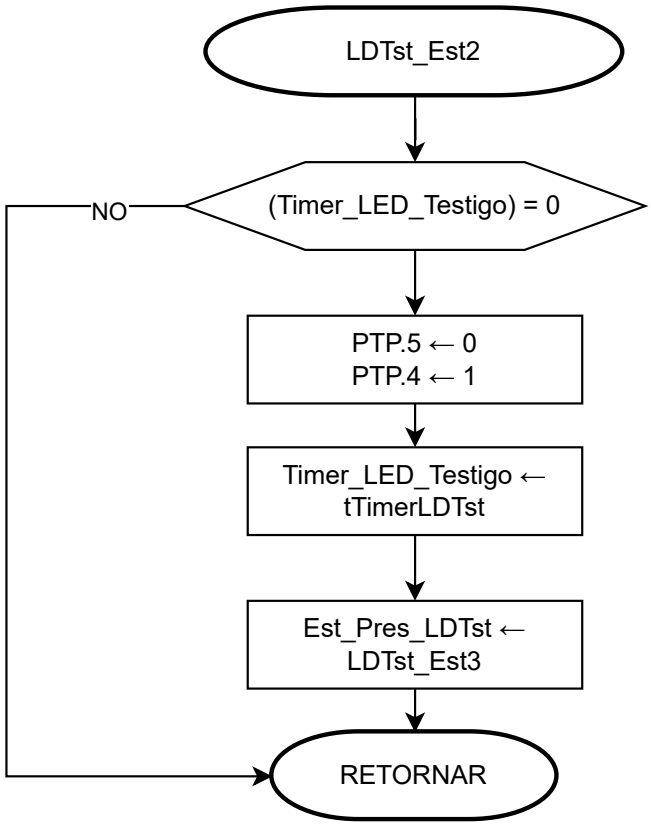

Figura 28: Diagrama de flujos de LDTst_Est2

2.6.6.3. LDTst_Est3

En este estado se espera a que finalice `Timer_LED_Testigo`. Cuando este finaliza, se apaga el LED rojo y se enciende el LED verde. Se carga el estado `LDTst_Est1` para esperar nuevamente a `Timer_LED_Testigo` y después encender el LED azul. Se actualiza el próximo estado por medio de la variable de estado `Est_Pres_LDTst`. El comportamiento descrito anteriormente se muestra en la Figura 29.

Parámetros de Entrada

- `Timer_LED_Testigo`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- `Est_Pres_LDTst`: Devuelto por medio de direccionamiento a memoria RAM.
- `PTP`: Devuelto por medio de direccionamiento a memoria RAM.
- `Timer_LED_Testigo`: Accedido por medio de direccionamiento extendido.

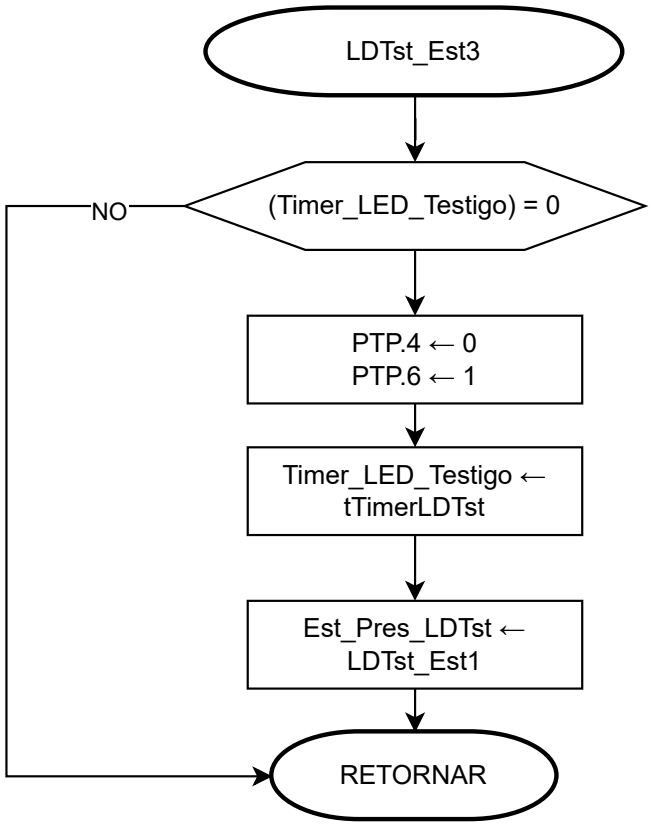

Figura 29: Diagrama de flujos de LDTst_Est3

2.6.7. Tarea_Leer_PB1/2

Estas tareas implementan máquinas de estado que leen los botones pulsadores ubicados en los pads PH3 (S1) y PH0 (S2) y suprimen los rebotes de los mismos. Debido a que estas máquinas de estado son idénticas, con la excepción de que el sufijo de las variables que cada una usa cambia de 1 a 2, solo se mostrarán los diagramas de flujos correspondientes a solo una de ellas. Para implementar esta máquina de estados, se hace uso de la variable de estado `EstPres_LeerPB1/2`, la cual contiene el próximo estado al cual saltará la máquina de estados en `Tarea_Leer_PB1/2`. El comportamiento descrito anteriormente se muestra en la Figura 30.

Parámetros de Entrada

- `EstPres_LeerPB1/2`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- Esta tarea no tiene parámetros de salida.

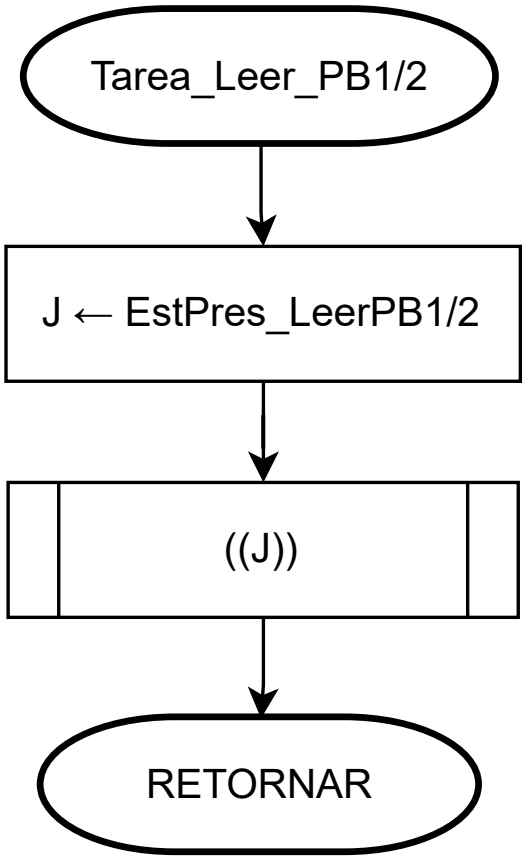

Figura 30: Diagrama de flujos de Tarea_Leer_PB1/2

2.6.7.1. LeerPB1/2_Est1

En este estado, se espera a que el botón pulsador sea presionado. Una vez es presionado, se cargan los timers `Timer_Reb_PB1/2`, `Timer_SHP1/2`, y `Timer_LP1/2` para suprimir rebotes, detectar una pulsación corta, y una pulsación larga. En esta aplicación no se hace uso de la función de pulsación larga, este detalle de diseño es un remanente del diseño realizado de esta tarea en evaluaciones previas del curso. Una vez cargados los timers, se salta al estado `LeerPB1/2_Est2` para esperar a que finalice el timer de supersión de rebotes. Se actualiza el próximo estado por medio de la variable de estado `EstPres_LeerPB1/2`. El comportamiento descrito anteriormente se muestra en la Figura 31.

Parámetros de Entrada

- `Banderas_1`: Accedido por medio de direccionamiento extendido.
- `PortPB`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- `EstPres_LeerPB1/2`: Devuelto por medio de direccionamiento a memoria RAM.
- `Timer_Reb_PB1/2`: Devuelto por medio de direccionamiento a memoria RAM.
- `Timer_SHP1/2`: Devuelto por medio de direccionamiento a memoria RAM.
- `Timer_LP1/2`: Devuelto por medio de direccionamiento a memoria RAM.

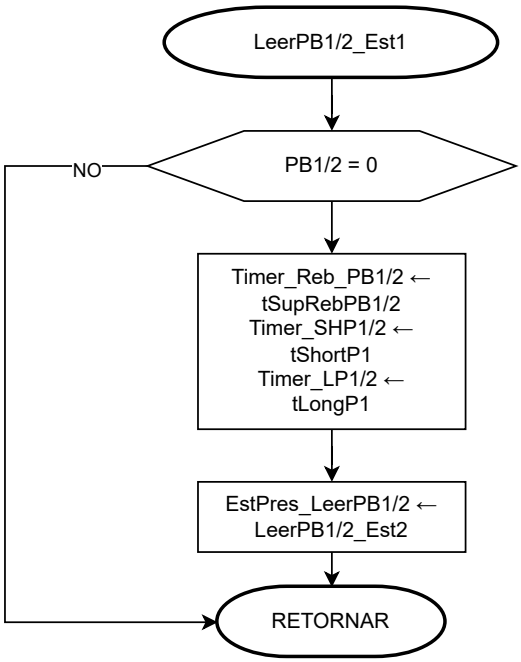

Figura 31: Diagrama de flujos de LeerPB1/2_Est1

2.6.7.2. LeerPB1/2_Est2

En este estado, se espera que termine el timer `Timer_Reb_PB1/2` para suprimir los rebotes del botón. En caso de que haya sido una lectura válida (se lee 0 en el PB en el estado 1 y en el estado 2), se salta al estado `LeerPB1/2_Est3` para esperar a que finalice el timer pulsación corta. Por lo contrario, se devuelve al estado `LeerPB1/2_Est1`, descartando la lectura ya que fue inválida. Se actualiza el próximo estado por medio de la variable de estado `EstPres_LeerPB1/2`. El comportamiento descrito anteriormente se muestra en la Figura 32.

Parámetros de Entrada

- `Banderas_1`: Accedido por medio de direccionamiento extendido.
- `Timer_Reb_PB1/2`: Accedido por medio de direccionamiento extendido.
- `PortPB`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- `Banderas_1`: Devuelto por medio de direccionamiento a memoria RAM.
- `EstPres_LeerPB1/2`: Devuelto por medio de direccionamiento a memoria RAM.

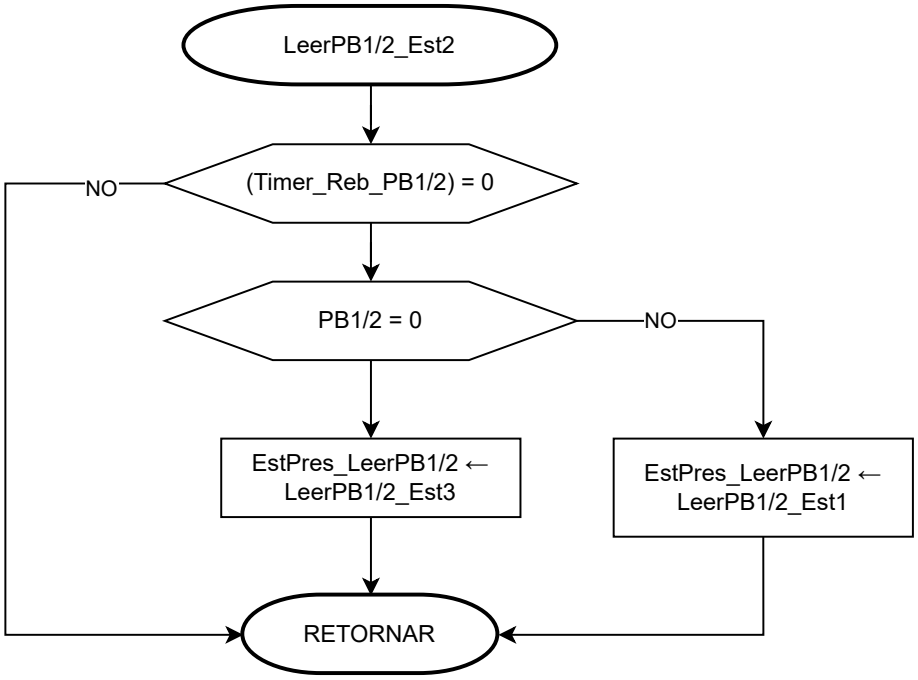

Figura 32: Diagrama de flujos de LeerPB1/2_Est2

2.6.7.3. LeerPB1/2_Est3

En este estado, se espera que termine el timer `Timer_SHP1/2` para detectar una pulsación corta. En caso de que el botón sigue siendo presionado (se lee 0 en el PB en el estado 2 y en el estado 3), se salta al estado `LeerPB1/2_Est4` para esperar a que finalice el timer pulsación larga, ya que es probable que la pulsación sea larga. Por lo contrario, se devuelve al estado `LeerPB1/2_Est1`, levantando la bandera `ShortP1/2`, ya que se trata de una pulsación corta. Se actualiza el próximo estado por medio de la variable de estado `EstPres_LeerPB1/2`. El comportamiento descrito anteriormente se muestra en la Figura 33.

Parámetros de Entrada

- `Banderas_1`: Accedido por medio de direccionamiento extendido.
- `Timer_SHP1/2`: Accedido por medio de direccionamiento extendido.
- `PortPB`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- `Banderas_1`: Devuelto por medio de direccionamiento a memoria RAM.
- `EstPres_LeerPB1/2`: Devuelto por medio de direccionamiento a memoria RAM.

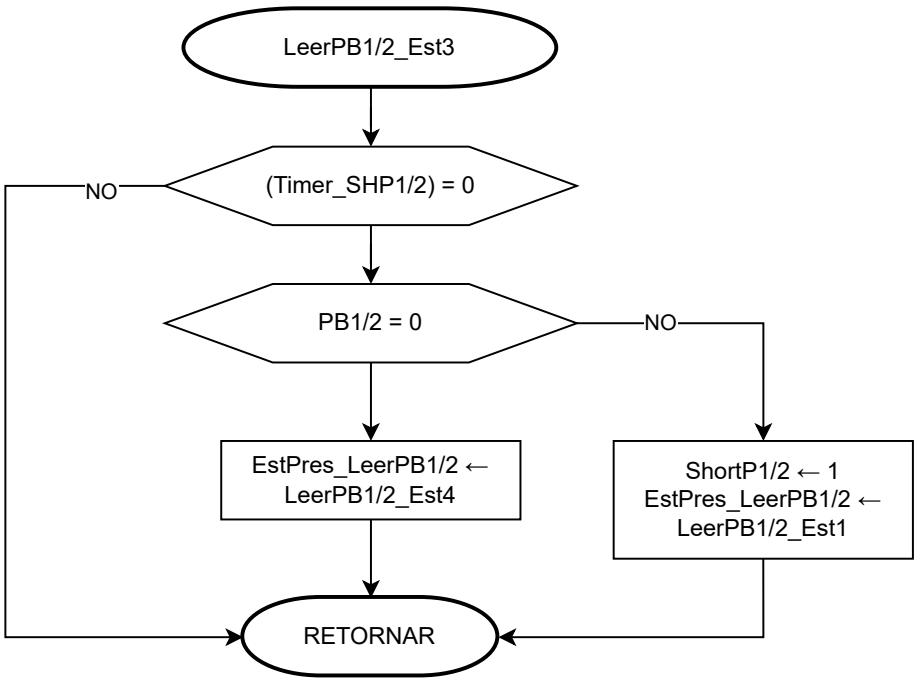

Figura 33: Diagrama de flujos de LeerPB1/2_Est3

2.6.7.4. LeerPB1/2_Est4

En este estado se espera a que el timer de pulsación larga termine. Si se sigue presionando el botón una vez que este timer termine, se determina que la pulsación fue larga. Por lo contrario, si aun no ha terminado el timer, se verifica si el botón sigue siendo pulsado. Si ya no sigue siendo pulsado, esto implica que la pulsación fue corta, por lo contrario, no se hace nada. Cuando se determina el tipo de pulsación, se pasa al estado 1 para leer nuevamente otra pulsación del botón. Se actualiza el próximo estado por medio de la variable de estado `EstPres_LeerPB1/2`. El comportamiento descrito anteriormente se muestra en la Figura 34.

Parámetros de Entrada

- `Banderas_1`: Accedido por medio de direccionamiento extendido.
- `PortPB`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- `Banderas_1`: Devuelto por medio de direccionamiento a memoria RAM.
- `EstPres_LeerPB1/2`: Devuelto por medio de direccionamiento a memoria RAM.

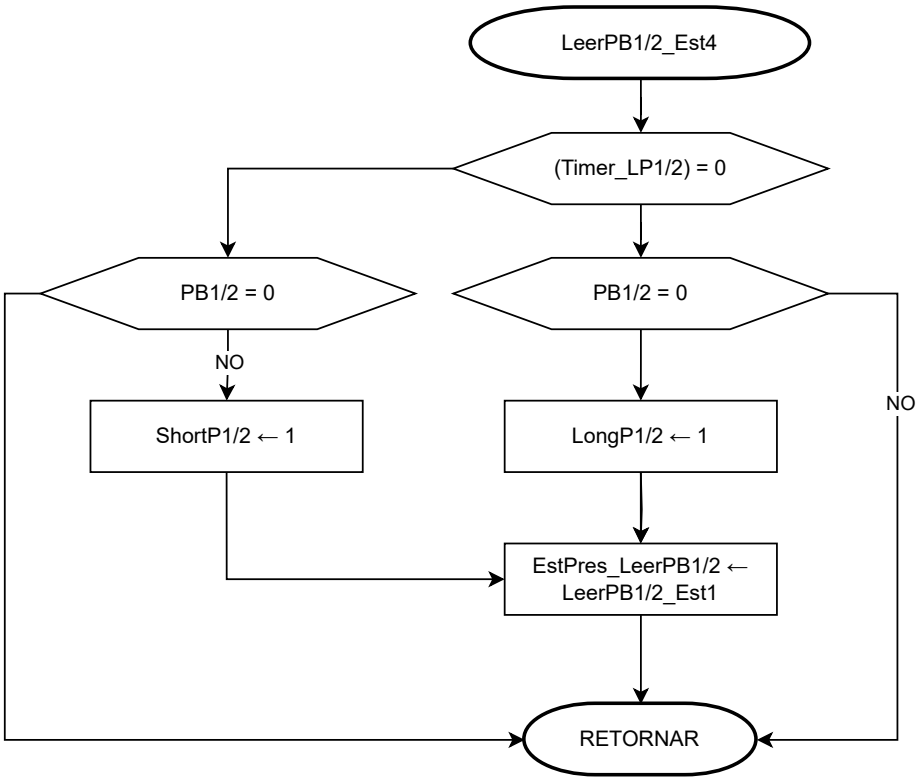

Figura 34: Diagrama de flujos de LeerPB1/2_Est4

2.6.8. Tarea_Leer_DS

Esta tarea implementa una máquina de estados que lee los dipswitches en la tarjeta Dragon 12+, y suprime los rebotes de la misma. Por medio de estos dipswitches se puede seleccionar uno de los 3 modos de operación del Selector 623. Para implementar esta máquina de estados, se hace uso de la variable de estado `Est_Pres_LeerDS`, la cual contiene el próximo estado al cual saltará la máquina de estados en `Tarea_Leer_DS`. El comportamiento descrito anteriormente se muestra en la Figura 35.

Parámetros de Entrada

- `Est_Pres_LeerDS`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- Esta tarea no tiene parámetros de salida.

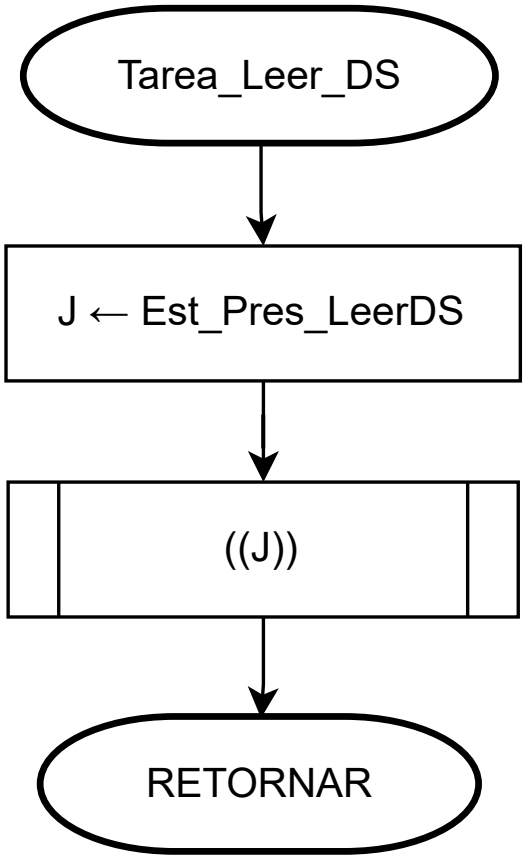

Figura 35: Diagrama de flujos de Tarea_Leer_DS

2.6.8.1. LeerDS_Est1

En este estado, se lee por primera vez los dipswitches y se guardan temporalmente en `Temp_DS`. Se carga `Timer_Reb_DS` para suprimir los rebotes del mismo, y se pasa al estado 2 para esperar a que este timer termine. Se actualiza el próximo estado por medio de la variable de estado `Est_Pres_LeerDS`. El comportamiento descrito anteriormente se muestra en la Figura 36.

Parámetros de Entrada

- `PortPB`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- `Temp_DS`: Devuelto por medio de direccionamiento a memoria RAM.
- `Timer_Reb_DS`: Devuelto por medio de direccionamiento a memoria RAM.
- `Est_Pres_LeerDS`: Devuelto por medio de direccionamiento a memoria RAM.

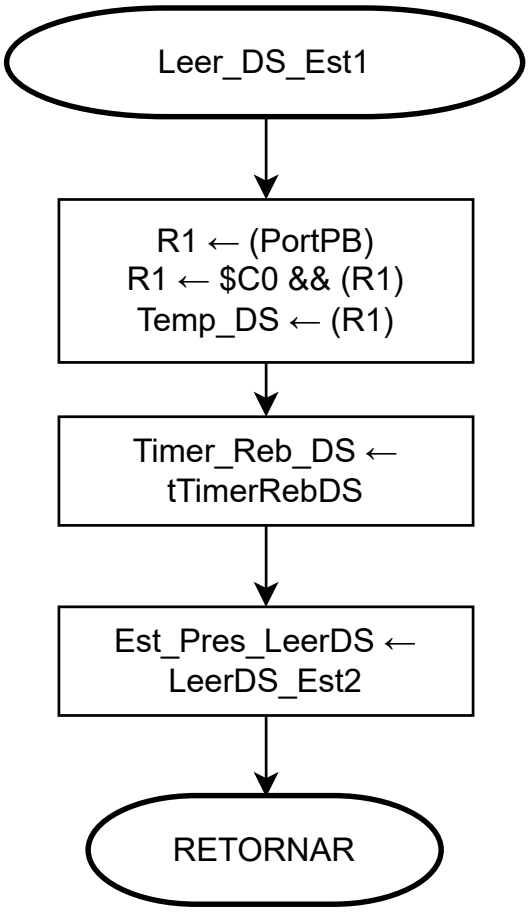

Figura 36: Diagrama de flujos de LeerDS_Est1

2.6.8.2. LeerDS_Est2

En este estado, se lee por segunda vez los dipswitches, después de que el timer de rebotes finalice. Si el estado de los dipswitches sigue siendo el que se leyó en el estado anterior, se guarda su valor en `Valor_DS`. Por lo contrario, se descarta la lectura y se regresa al estado 1. Se actualiza el próximo estado por medio de la variable de estado `Est_Pres_LeerDS`. El comportamiento descrito anteriormente se muestra en la Figura 37.

Parámetros de Entrada

- `PortPB`: Accedido por medio de direccionamiento extendido.
- `Timer_Reb_DS`: Accedido por medio de direccionamiento extendido.
- `Temp_DS`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- `Valor_DS`: Devuelto por medio de direccionamiento a memoria RAM.
- `Est_Pres_LeerDS`: Devuelto por medio de direccionamiento a memoria RAM.

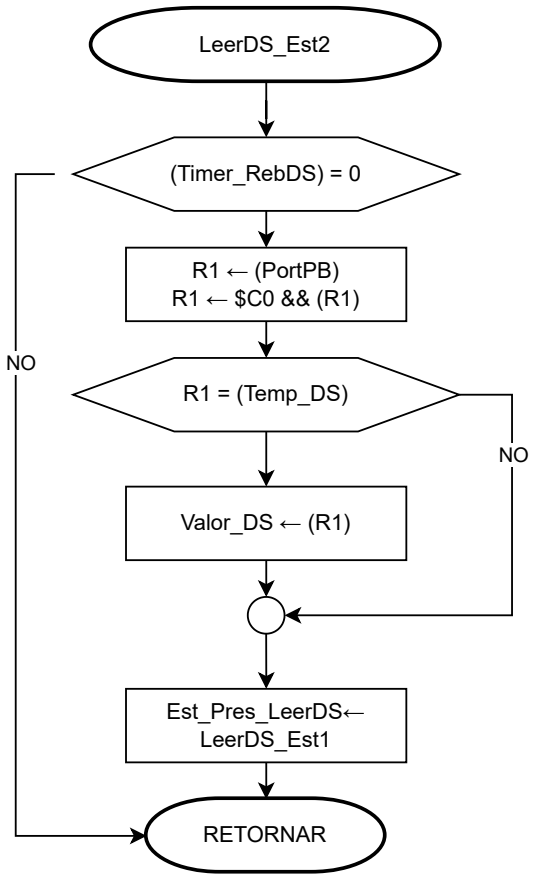

Figura 37: Diagrama de flujos de LeerDS_Est2

2.6.9. Tarea_PantallaMUX

Esta tarea implementa una máquina de estados denominada `PantallaMUX`, la cual está encargada de multiplexar los LEDS y los displays de 7 segmentos conectados a PORTB. La máquina de estados salta al próximo estado por medio de la variable de estado `Est_Pres_PantallaMUX`. El comportamiento descrito anteriormente se muestra en la Figura 38.

Parámetros de Entrada

- `Est_Pres_PantallaMUX`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- Esta tarea no tiene parámetros de salida.

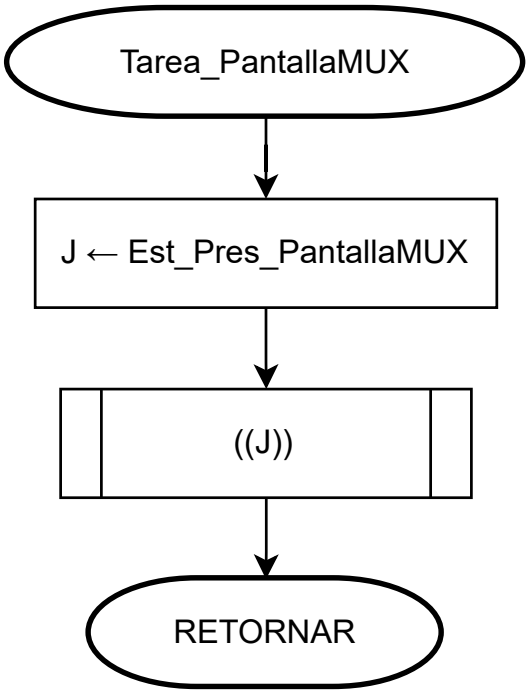

Figura 38: Diagrama de flujos de Tarea_PantallaMUX

2.6.9.1. PantallaMUX_Est1

En este estado, se cicla el valor desplegado en `PORTB` en base al contador `Cont_Dig`, el cual define que display o LEDS se van a habilitar durante un tiempo `tTimerDigito`. Se salta al estado 2 para definir el brillo del valor desplegado. Se actualiza el próximo estado por medio de la variable de estado `Est_Pres_PantallaMUX`. El comportamiento descrito anteriormente se muestra en la Figura 39.

Parámetros de Entrada

- `TimerDigito`: Accedido por medio de direccionamiento extendido.
- `DISP1`: Accedido por medio de direccionamiento extendido.
- `DISP2`: Accedido por medio de direccionamiento extendido.
- `DISP3`: Accedido por medio de direccionamiento extendido.
- `DISP4`: Accedido por medio de direccionamiento extendido.
- `PORTB`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- `TimerDigito`: Devuelto por medio de direccionamiento a memoria RAM.
- `DIG1`: Devuelto por medio de direccionamiento a memoria RAM.
- `DIG2`: Devuelto por medio de direccionamiento a memoria RAM.
- `DIG3`: Devuelto por medio de direccionamiento a memoria RAM.
- `DIG4`: Devuelto por medio de direccionamiento a memoria RAM.
- `PORTB`: Devuelto por medio de direccionamiento a memoria RAM.
- `Cont_Dig`: Devuelto por medio de direccionamiento a memoria RAM.
- `PTJ`: Devuelto por medio de direccionamiento a memoria RAM.
- `Est_Pres_PantallaMUX`: Devuelto por medio de direccionamiento a memoria RAM.

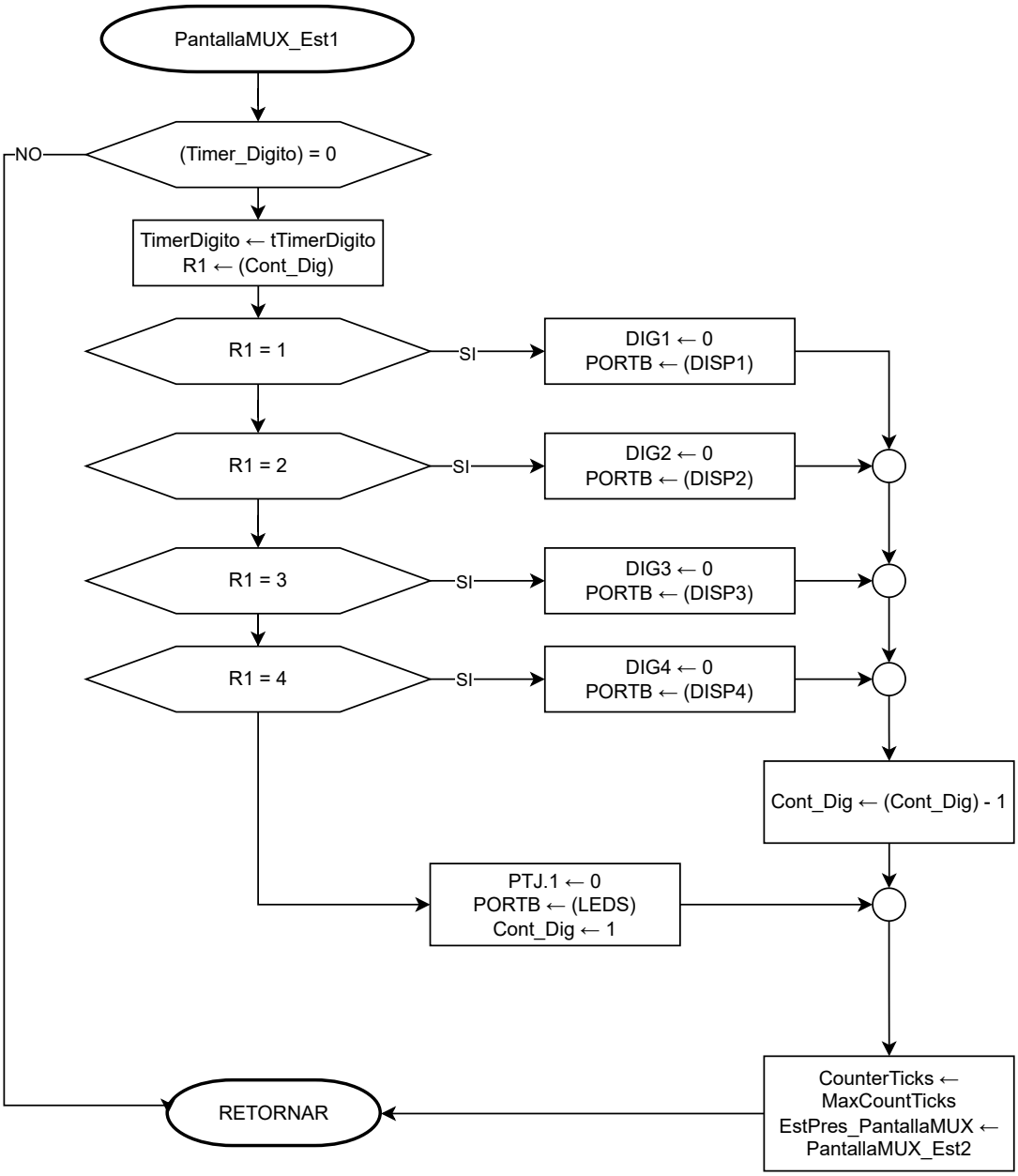

Figura 39: Diagrama de flujos de PantallaMUX_Est1

2.6.9.2. PantallaMUX_Est2

En este estado se controla el momento en el cual se deshabilita el dígito siendo desplegado actualmente. Esto es realizado en base al valor de la variable `Brillo`, lo cual permite controlar el nivel de luminosidad del valor siendo desplegado actualmente. Se actualiza el próximo estado por medio de la variable de estado `Est_Pres_PantallaMUX`. El comportamiento descrito anteriormente se muestra en la Figura 40.

Parámetros de Entrada

- `CounterTicks`: Accedido por medio de direccionamiento extendido.
- `Brillo`: Accedido por medio de direccionamiento extendido.
- `PORTB`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- `PTP`: Devuelto por medio de direccionamiento a memoria RAM.
- `PTJ`: Devuelto por medio de direccionamiento a memoria RAM.
- `Est_Pres_PantallaMUX`: Devuelto por medio de direccionamiento a memoria RAM.

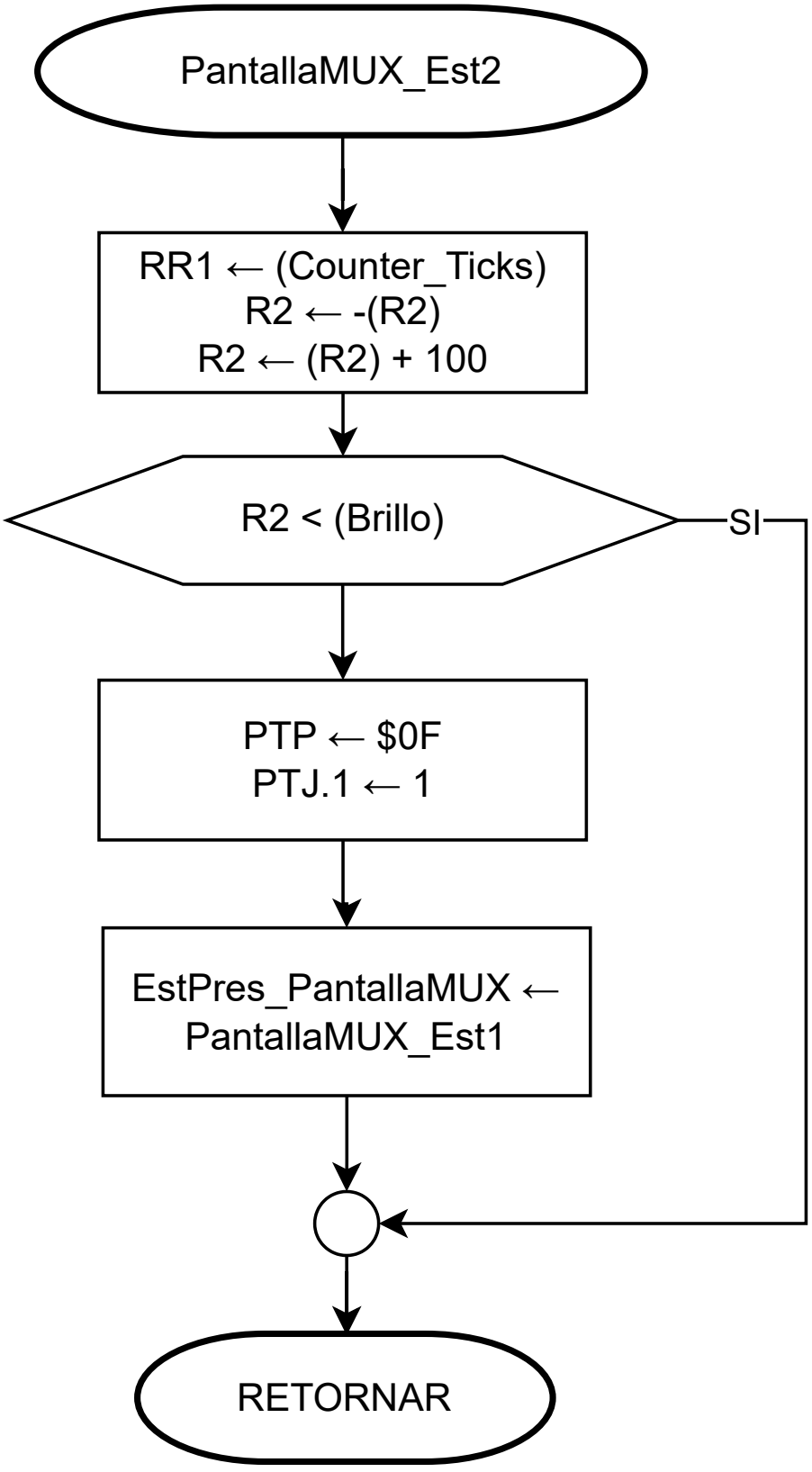

Figura 40: Diagrama de flujos de PantallaMUX_Est2

2.6.10. Tarea_LCD

Esta tarea implementa la máquina de estados `TareaLCD`, la cual implementa el protocolo estroboscópico para desplegar mensajes en la pantalla LCD en la tarjeta Dragon 12+. Debido al largo tiempo de ejecución de esta tarea en comparación al resto de tareas en el despachador de tareas, se ejecuta únicamente si `LCD_Ok` = 0; es decir, si se solicitó desplegar un mensaje en la pantalla LCD. Se actualiza el próximo estado por medio de la variable de estado `EstPres_TareaLCD`. El comportamiento descrito anteriormente se muestra en la Figura 41.

Parámetros de Entrada

- `EstPres_TareaLCD`: Accedido por medio de direccionamiento extendido.
- `Banderas_2`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- Esta tarea no tiene parámetros de salida

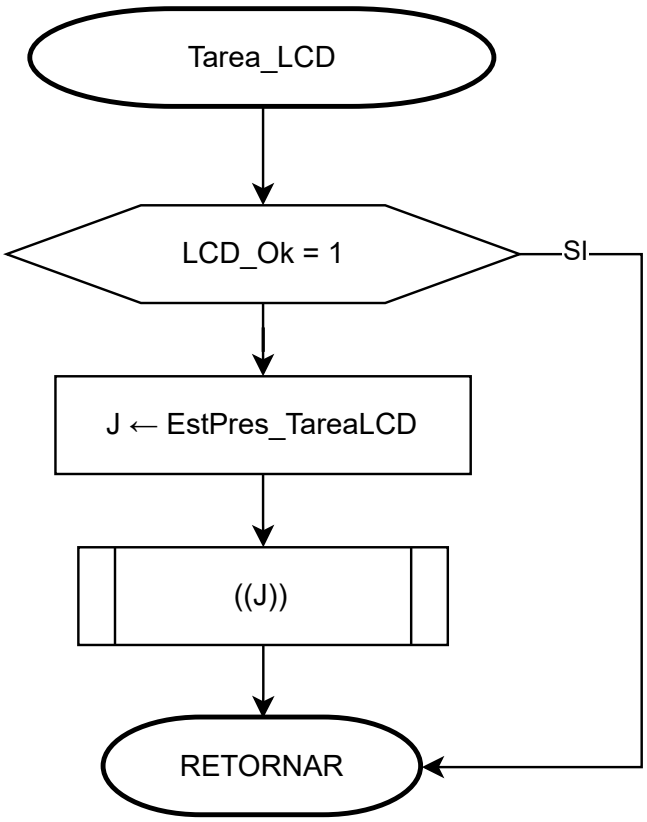

Figura 41: Diagrama de flujos de Tarea_LCD

2.6.10.1. TareaLCD_Est1

En este estado se pregunta si es hora de desplegar la primera o segunda línea en la pantalla LCD, y llama a la subrutina `Tarea_SendLCD` la cual se encarga de enviar el mensaje a la pantalla. Se pasa al estado 2 para enviar cada uno de los caracteres del mensaje. Se actualiza el próximo estado por medio de la variable de estado `EstPres_TareaLCD`. El comportamiento descrito anteriormente se muestra en la Figura 42.

Parámetros de Entrada

- `MSG_L1`: Accedido por medio de direccionamiento extendido.
- `MSG_L2`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- `EstPres_TareaLCD`: Devuelto por medio de direccionamiento a memoria RAM.
- `CharLCD`: Devuelto por medio de direccionamiento a memoria RAM.
- `Punt_LCD`: Devuelto por medio de direccionamiento a memoria RAM.
- `Banderas_2`: Devuelto por medio de direccionamiento a memoria RAM.

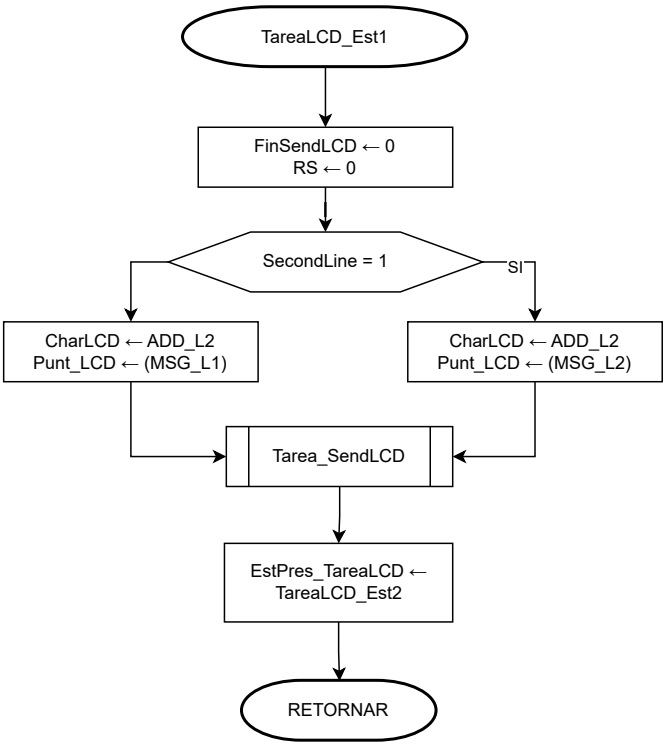

Figura 42: Diagrama de flujos de TareaLCD_Est1

2.6.10.2. TareaLCD_Est2

En este estado se recorre el arreglo de caracteres que conforman el mensaje que se va enviar a la pantalla LCD y se llama a `Tarea_SendLCD` para implementar el protocolo estroboscópico. Una vez es recorrido todo el arreglo, se pasa al estado 1 para mandar la segunda línea del mensaje, si fuese necesario. Se actualiza el próximo estado por medio de la variable de estado `EstPres_TareaLCD`. El comportamiento descrito anteriormente se muestra en la Figura 43.

Parámetros de Entrada

- `Banderas_2`: Accedido por medio de direccionamiento extendido.
- `Punt_LCD`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- `EstPres_TareaLCD`: Devuelto por medio de direccionamiento a memoria RAM.
- `Punt_LCD`: Devuelto por medio de direccionamiento a memoria RAM.

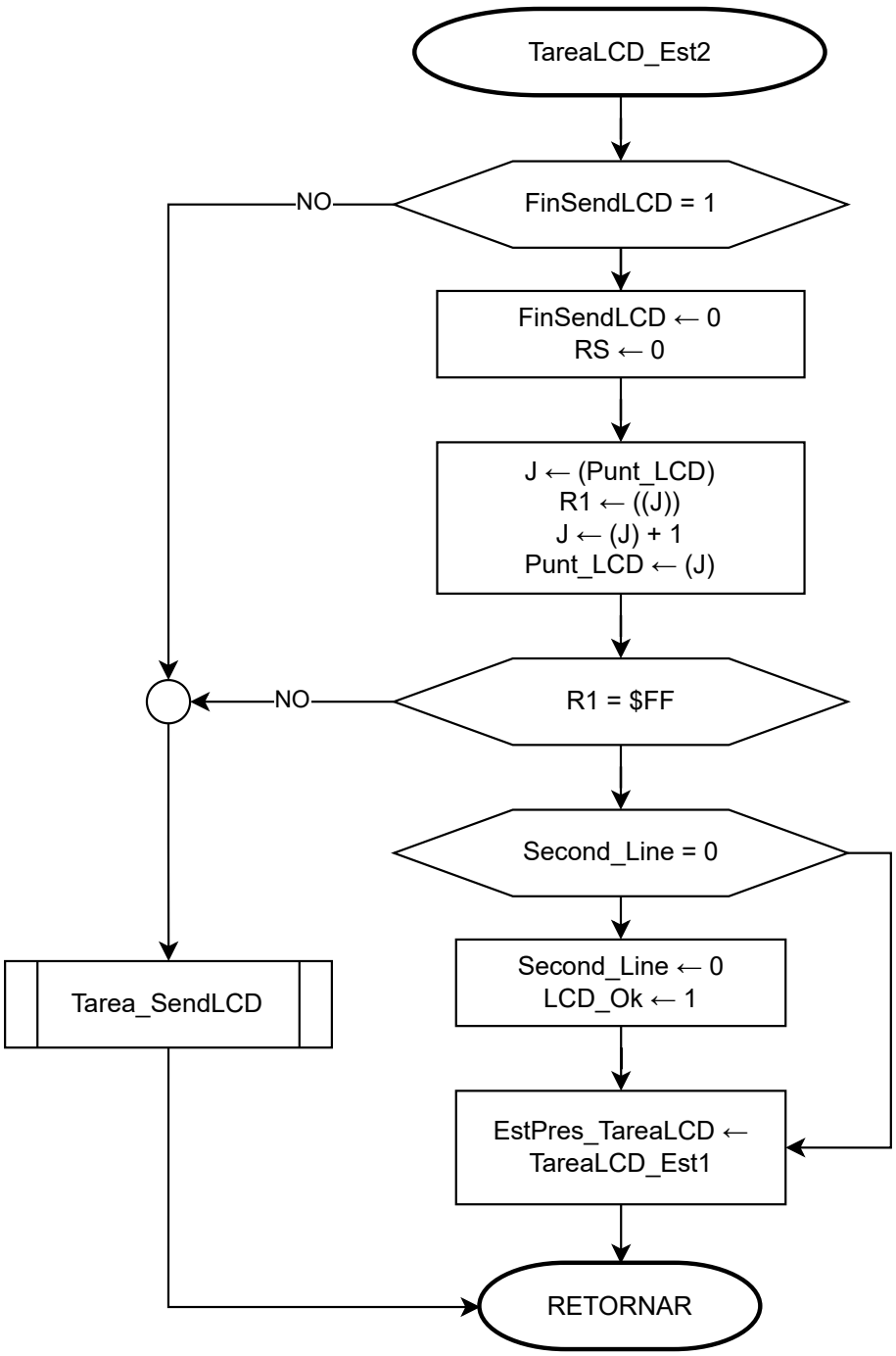

Figura 43: Diagrama de flujos de TareaLCD_Est2

2.6.11. Send_LCD

Esta tarea es llamada por `TareaLCD` para implementar el protocolo estroboscópico para desplegar mensajes en la pantalla LCD. Se actualiza el próximo estado por medio de la variable de estado `EstPres_TareaSendLCD`. El comportamiento descrito anteriormente se muestra en la Figura 44.

Parámetros de Entrada

- `EstPres_SendLCD`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- Esta tarea no tiene parámetros de entrada.

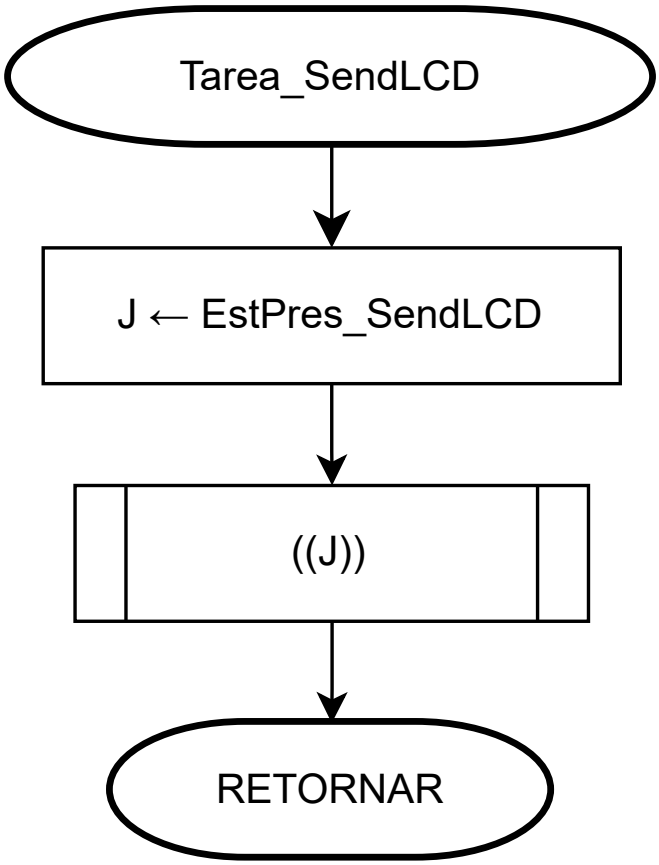

Figura 44: Diagrama de flujos de TareaSendLCD

2.6.11.1. SendLCD_Est1

En este estado se envía un caracter a la pantalla LCD, primero enviando el nibble superior. Además de esto, se carga `Timer260uS` debido a que la transferencia de datos entre el MCU y la pantalla LCD no es inmediata, y ocupa cierto tiempo de retardo para que la acción se ejecuta adecuadamente. Se actualiza el próximo estado por medio de la variable de estado `EstPres_SendLCD`. El comportamiento descrito anteriormente se muestra en la Figura 45.

Parámetros de Entrada

- `CharLCD`: Accedido por medio de direccionamiento extendido.
- `Bandras_2`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- `PORTK`: Devuelto por medio de direccionamiento a memoria RAM.
- `EstPres_SendLCD`: Devuelto por medio de direccionamiento a memoria RAM.
- `Timer260uS`: Devuelto por medio de direccionamiento a memoria RAM.

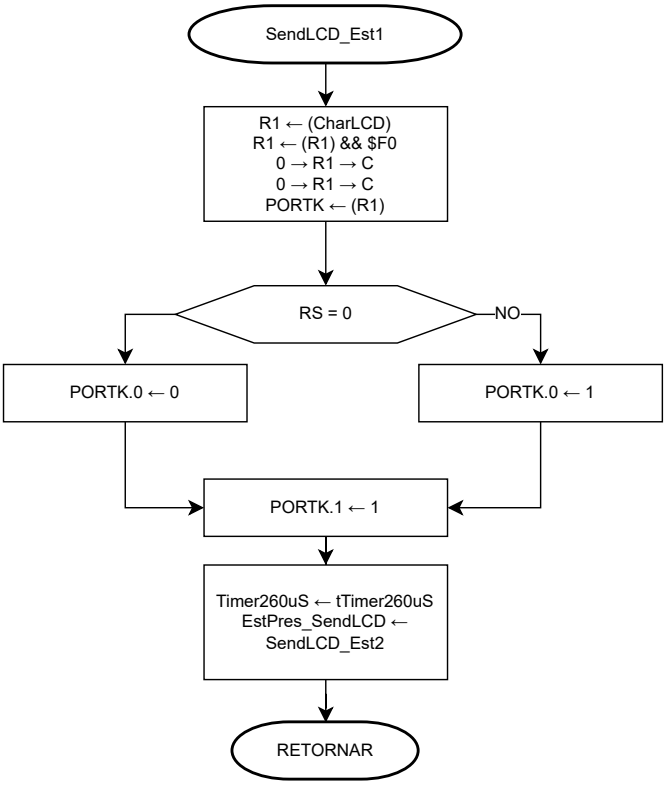

Figura 45: Diagrama de flujos de SendLCD_Est1

2.6.11.2. SendLCD_Est2

En este estado se envía un caracter a la pantalla LCD, ahora enviando el nibble inferior. Además de esto, se carga `Timer260uS` debido a que la transferencia de datos entre el MCU y la pantalla LCD no es inmediata, y ocupa cierto tiempo de retardo para que la acción se ejecuta adecuadamente. Se actualiza el próximo estado por medio de la variable de estado `EstPres_SendLCD`. El comportamiento descrito anteriormente se muestra en la Figura 46.

Parámetros de Entrada

- `CharLCD`: Accedido por medio de direccionamiento extendido.
- `Bandras_2`: Accedido por medio de direccionamiento extendido.
- `Timer260uS`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- `PORTK`: Devuelto por medio de direccionamiento a memoria RAM.
- `EstPres_SendLCD`: Devuelto por medio de direccionamiento a memoria RAM.
- `Timer260uS`: Devuelto por medio de direccionamiento a memoria RAM.

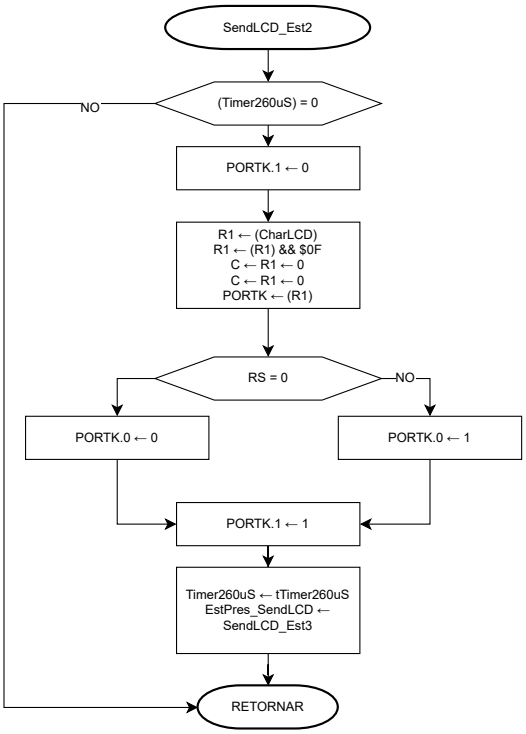

Figura 46: Diagrama de flujos de SendLCD_Est2

2.6.11.3. SendLCD_Est3

En este estado se espera a que finalice `Timer260uS` por el último caracter enviado a la pantalla LCD. Una vez finaliza este timer, se carga `Timer40uS` como secuencia final del protocolo estroboscópico que implementa esta máquina de estados. Se actualiza el próximo estado por medio de la variable de estado `EstPres_SendLCD`. El comportamiento descrito anteriormente se muestra en la Figura 47.

Parámetros de Entrada

- `Timer260uS`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- `PORTK`: Devuelto por medio de direccionamiento a memoria RAM.
- `EstPres_SendLCD`: Devuelto por medio de direccionamiento a memoria RAM.
- `Timer260uS`: Devuelto por medio de direccionamiento a memoria RAM.
- `Timer40uS`: Devuelto por medio de direccionamiento a memoria RAM.

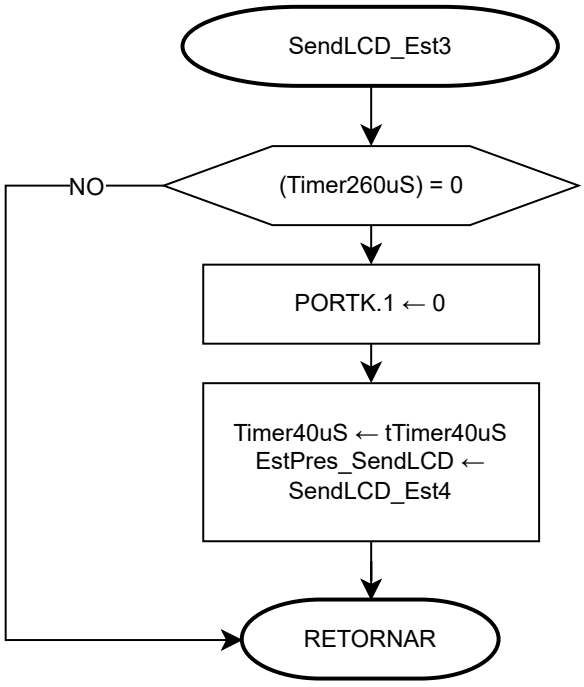

Figura 47: Diagrama de flujos de SendLCD_Est3

2.6.11.4. SendLCD_Est4

En este estado se espera a que finalice `Timer40uS`. Una vez finaliza este timer, se ha concluido con el protocolo estroboscópico para el envío de un mensaje a la pantalla LCD. En estado, se carga el estado inicial `SendLCD_Est1` para realizar nuevamente el envío de un mensaje a la pantalla LCD, si fuese necesario. Se actualiza el próximo estado por medio de la variable de estado `EstPres_SendLCD`. El comportamiento descrito anteriormente se muestra en la Figura 48.

Parámetros de Entrada

- `Timer40uS`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- `Banderas_2`: Devuelto por medio de direccionamiento a memoria RAM.
- `EstPres_SendLCD`: Devuelto por medio de direccionamiento a memoria RAM.

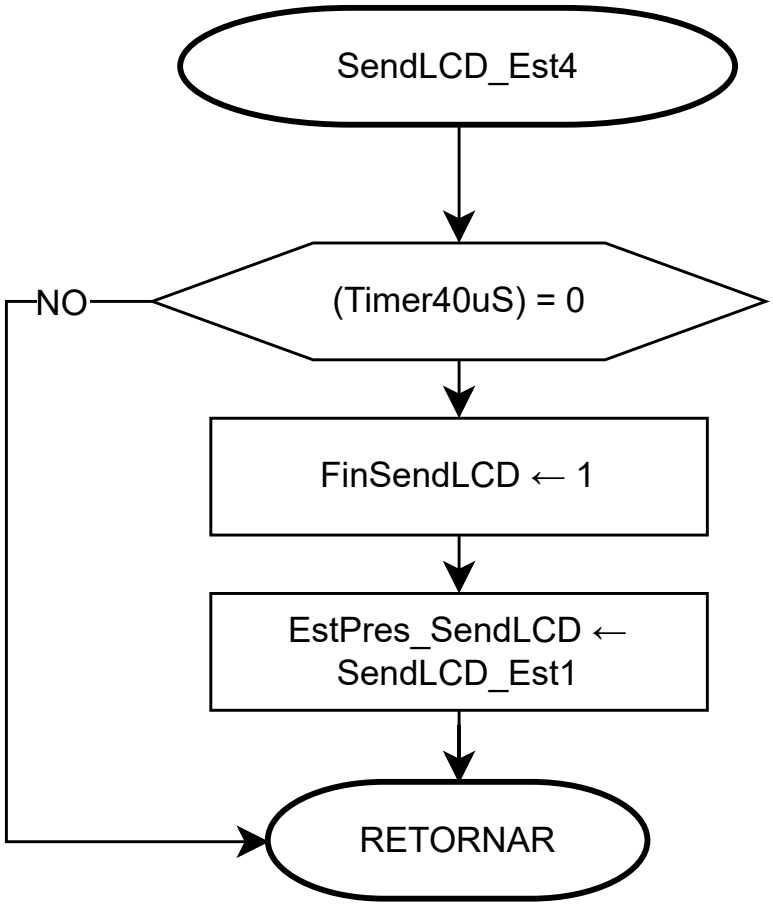

Figura 48: Diagrama de flujos de SendLCD_Est4

2.7. Subrutinas

2.7.1. Calcula

Esta subrutina calcula distintos parámetros cinemáticos necesarios para procesar una barra de aluminio de la siderúrgica. Estos se denotan a continuación:

Parámetros de Entrada

- `DeltaT`: Accedido por medio de direccionamiento extendido.
- `TimerCal`: Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- `Velocidad`: La velocidad a la que viaja la barra, calculada a partir del primer intervalo de tiempo `DeltaT`. Devuelto a través de direccionamiento a memoria RAM.
- `Longitud`: La longitud de la barra, calcualada a partir del segundo intervalo de tiempo `DeltaT`. Devuelto por medio de direccionamiento a memoria RAM.
- `TimerPant`: Timer para iniciar a desplegar un mensaje en la pantalla. Devuelto por medio de direccionamiento a memoria RAM.
- `TimerFinPant`: Timer para indicar el fin de despliegue de un mensaje en la pantalla. Devuelto por medio de direccionamiento a memoria RAM.
- `TimerRociador`: Timer para indicar la activación del microrrele, el cual habilita el rociador. Devuelto por medio de direccionamiento a memoria RAM.

El comportamiento descrito anteriormente se muestra en la Figura 49.

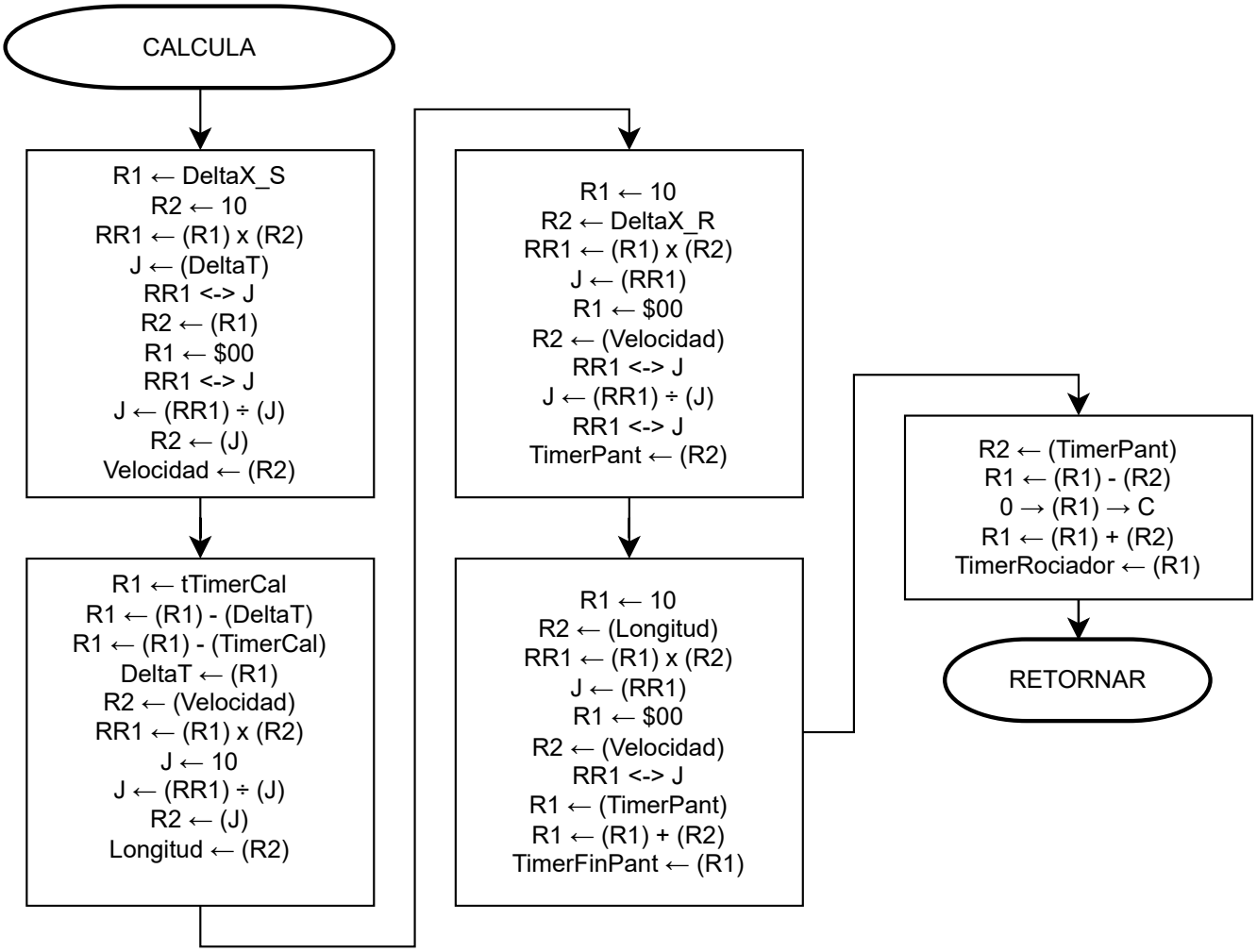

Figura 49: Diagrama de flujos de Calcula

2.7.2. BIN_BCD_MUXP

Esta subrutina convierte un valor en binario a BCD. A pesar de que fue primero realizada para la máquina de estados `PantallaMux`, es utilizada en otras subrutinas. El comportamiento descrito anteriormente se muestra en la Figura 50.

Parámetros de Entrada

- `R1`: Valor binario a convertir. Accedido por medio del acumuluador A.

Parámetros de Salida

- `BCD`: Valor BCD convertido. Devuelto por medio de direccionamiento a memoria RAM.

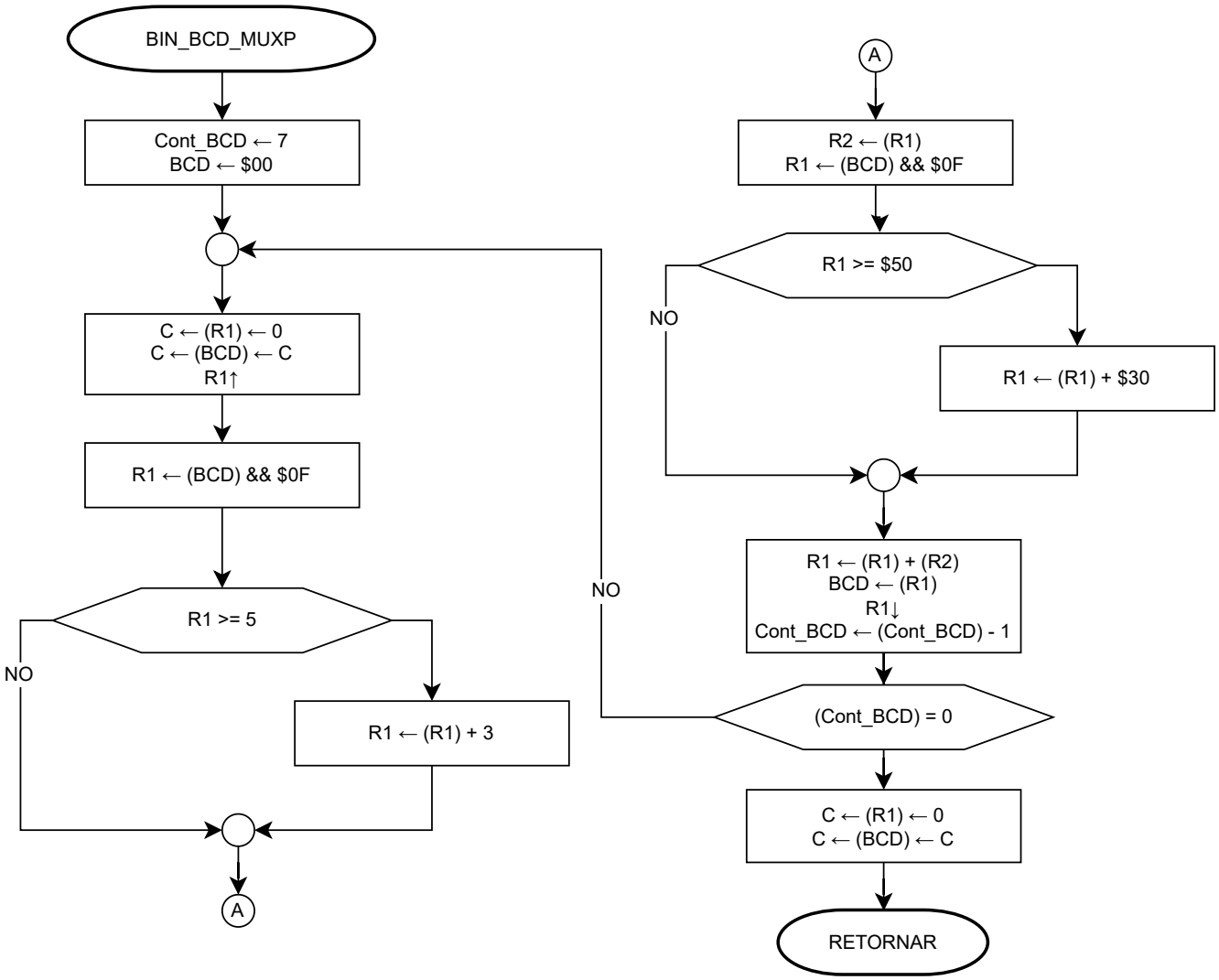

Figura 50: Diagrama de flujos de BIN_BCD_MUXP

2.7.3. BCD_7SEG

Esta subrutina convierte un valor BCD a 7 segmentos para ser desplegado en los displays de la tarjeta Dragon12+. El comportamiento descrito anteriormente se muestra en la Figura 51.

Parámetros de Entrada

- `BCD2`: Valor en BCD correspondiente a la parte alta del valor a convertir a 7 segmentos. Accedido por medio de direccionamiento extendido.
- `BCD1`: Valor en BCD correspondiente a la parte baja del valor a convertir a 7 segmentos. Accedido por medio de direccionamiento extendido.

Parámetros de Salida

- `DSP1`: Cuarto dígito del valor en 7 segmentos. Devuelto por medio de direccionamiento a memoria RAM.
- `DSP2`: Tercer dígito del valor en 7 segmentos. Devuelto por medio de direccionamiento a memoria RAM.
- `DSP3`: Segundo dígito del valor en 7 segmentos. Devuelto por medio de direccionamiento a memoria RAM.
- `DSP4`: Primer dígito del valor en 7 segmentos. Devuelto por medio de direccionamiento a memoria RAM.

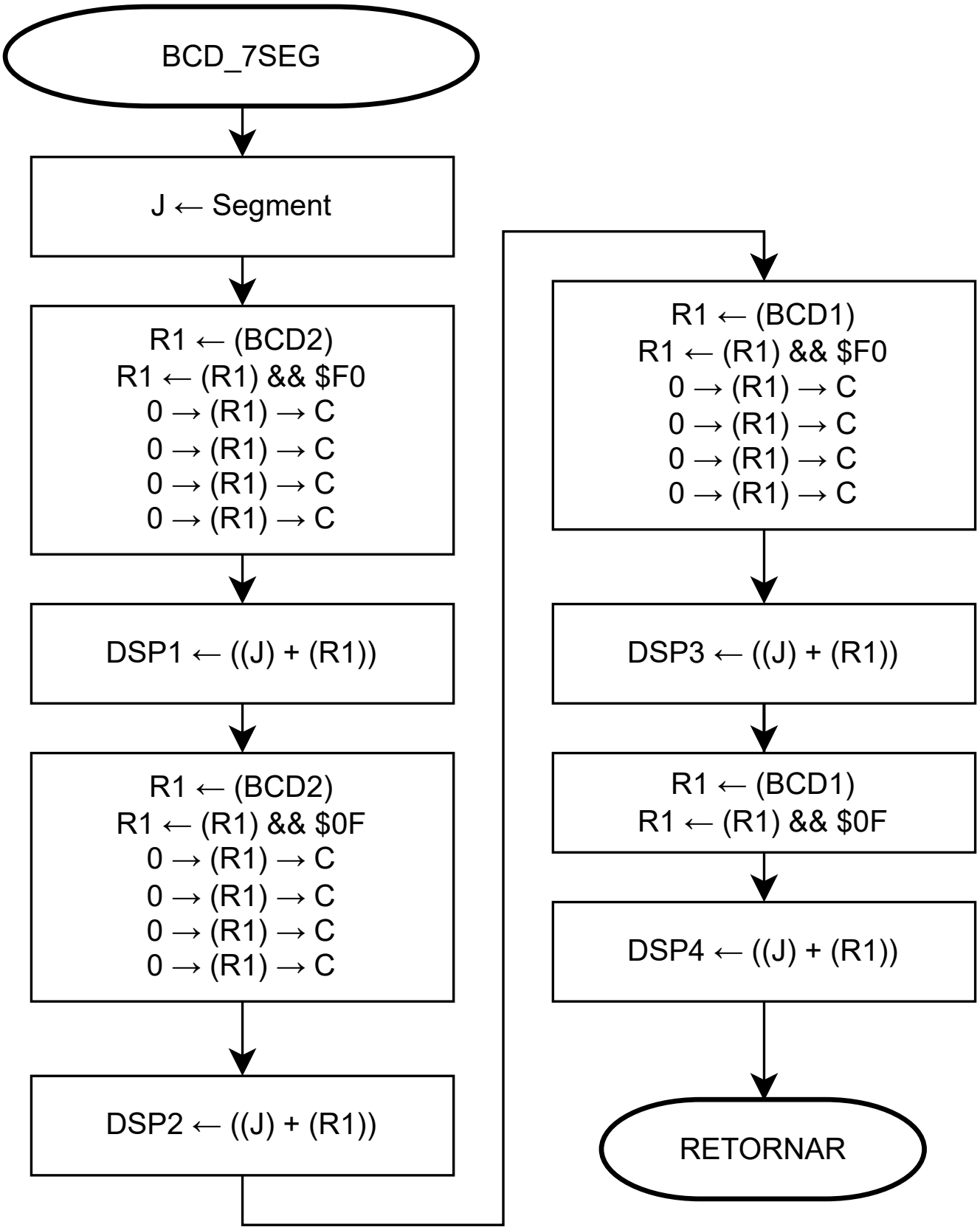

Figura 51: Diagrama de flujos de BCD_7SEG

2.7.4. Borrar_NumArray

Esta subrutina borra una secuencia de teclas válida digitada en el teclado matricial de la tarjeta Dragon 12+ y almacenado en el arreglo `NumArray`. De esta manera, una vez se borra el `NumArray`, su contenido corresponde a $FF en todas sus posiciones. El comportamiento descrito anteriormente se muestra en la Figura 52.

Parámetros de Entrada

- `NumArray`: Arreglo con secuencia de teclas válida. Accedido por medio de direccionamiento extendido.
- `Max_TCL`: Cantidad máxima de teclas que se pueden almacenar en `NumArray`.

Parámetros de Salida

- `NumArray` borrado ($FF desde la posición 0 hasta la posición `Max_TCL` − 1).

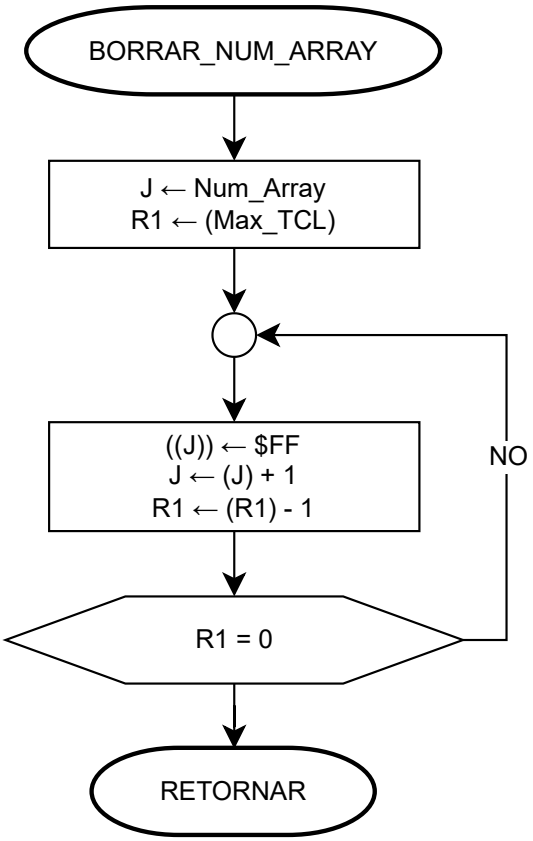

Figura 52: Diagrama de flujos de Borrar_NumArray

2.7.5. BCD_BIN

Esta subrutina convierte un valor en BCD y lo convierte a binario por medio de multiplicación de décadas. El comportamiento descrito anteriormente se muestra en la Figura 53.

Parámetros de Entrada

- `R2`: Valor BCD a convertir. Accedido a través del acumulador B.

Parámetros de Salida

- `ValorLong`: Valor en binario convertido. Devuelto por medio de direccionamiento a memoria RAM.

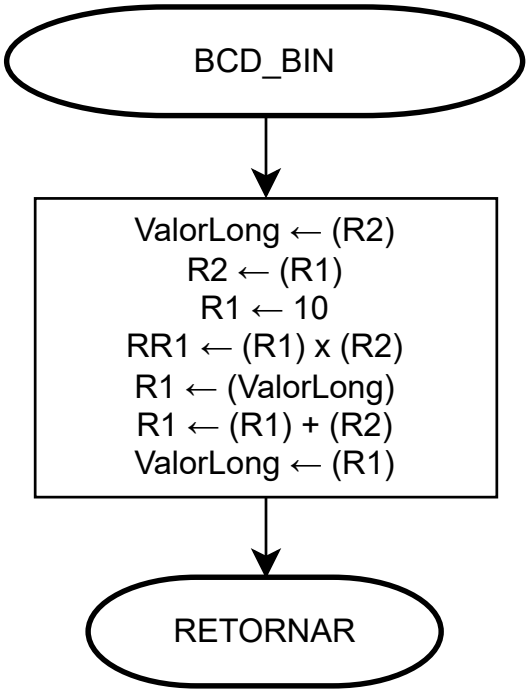

Figura 53: Diagrama de flujos de BCD_BIN

2.7.6. Máquina de Tiempos

La máquina de tiempos se compone de 3 subrutinas:

- `Maquina_Tiempos`: Esta subrutina de atención de interrupción por salida de comparación en el canal 4 (OC4) se encarga de verificar cuales timers deben de decrementarse, en base a los timers BaseT.
- `Decre_Timers`: Decrementar los timers definidos por el programador. Recibe la dirección de la tabla de timers a decrementar a través del índice X.
- `Decre_Timers_BaseT`: Decrementa los timers que generar las bases de tiempo de 1mS, 10mS, 100mS, y 1S. Recibe la dirección de la tabla de timers a decrementar a través del índice X.

La ISR `Maquina_Tiempos` se ejecuta a una frecuencia de interrupción de 50kHz, con el propósito de generar bases de tiempo lo suficientemente pequeñas para el uso adecuado de la pantalla LCD. Los diagramas de flujo asociados a la máquina de tiempos se muestran en la Figura 54.

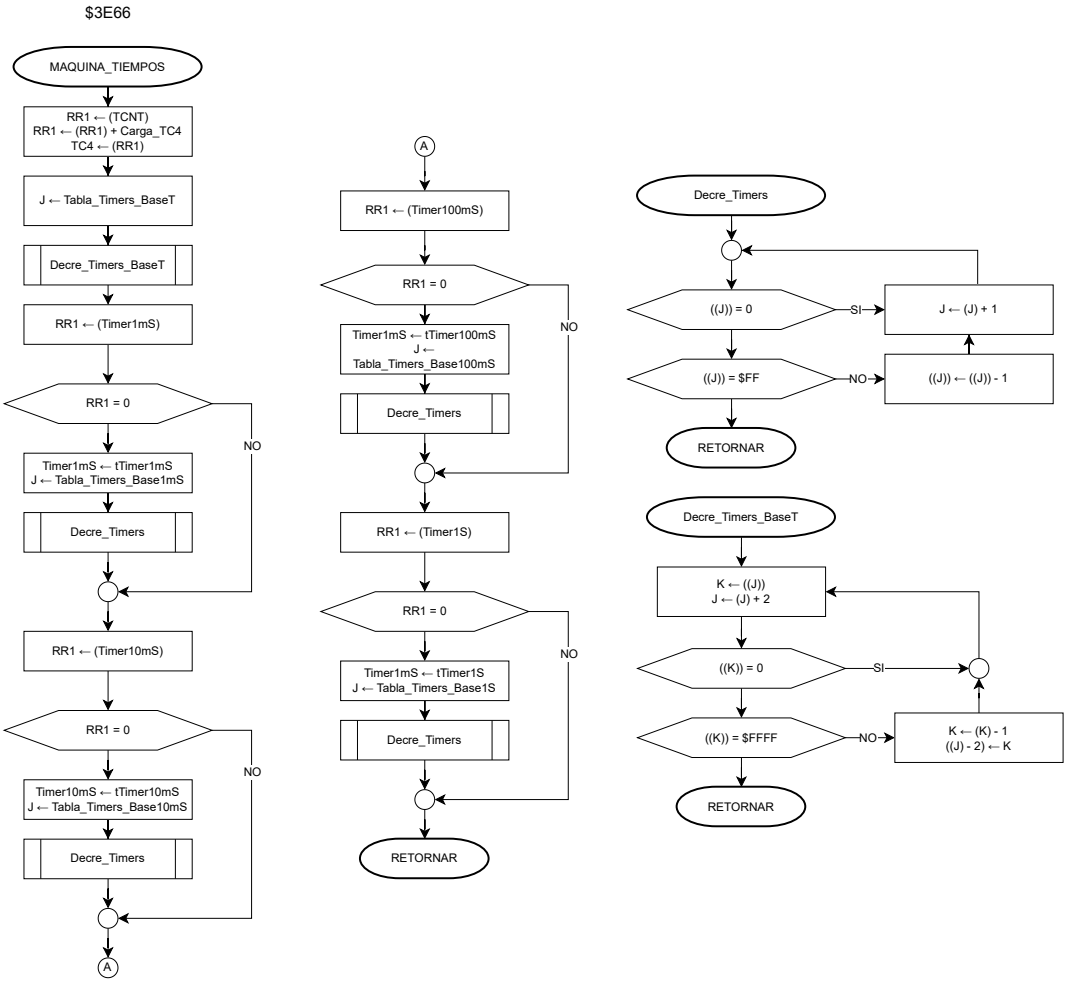

Figura 54: Diagrama de flujos de Maquina_Tiempos, Decre_Timers, y Decre_Timers_BaseT

3. Conclusiones y recomendaciones

3.1. Conclusiones

- Se diseñó el Selector 623 en la tarjeta Dragon12+, cumpliendo con todos los requisitos funcionales del mismo.
- Se integró de forma adecuada las subrutinas diseñadas a lo largo de las evaluaciones y laboratorios del curso, para diseñar de forma exitosa el Selector 623.

3.2. Recomendaciones

- Para comprobar el correcto funcionamiento del Selector 623, se recomineda utilizar un cronómetro e ir apuntando los tiempos en los que suceden cada uno de los eventos al procesar una barra en el modo Seleccionar. De esta forma, se asegura que la lógica del programa que calcula los timers correspondientes a la máquina de estados asociada al modo Seleccionar es correcta.
- Se recomienda hacer uso de herramientas de control de versiones como `git` para recuperar la versión funcional del programa que se está diseñando en caso de que este llegase a parar de funcionar. Esto es particularmente importante considerando la forma en que el Selector 623 fue programado, ya que se tienden a cometer errores con frecuencia al programar en lenguaje ensamblador.
- Para aquellos usuarios que hacen uso de un computador con un sistema operativo de la familia GNU/Linux, con el propósito de acelerar el proceso de desarrollo, se recomienda hacer uso de un archivo `Makefile` para ensamblar y cargar el progama a la tarjeta Dragon12+, ya que se puede probar el programa con la ejecución de un solo comando.
- En la misma línea de la recomendación anterior, también se recomienda utilizar el programa `minicom` para accesar a la consola de la tarjeta Dragon12+. Este programa sirve como alternativa a AsmIDE que, en caso de migrar las herramientas de desarrollo de AsmIDE a otra plataforma, es necesario un emulador de terminal, y `minicom` resulta ser una buena opción debido a que tiene soporte de los protocolos XOn y XOff, permitiendo la carga de objetos .s19 a la tarjeta Dragon12+.
- Para la depuración del programa, los comandos `md`, `t`, `pc`, `br`, `nobr`, `g`, `rd`, resultan ser de gran utilidad a la hora de tratar con un programa con cientos de líneas de código, como lo es el diseño del Selector 623, en donde muchas cosas pueden fallar al realizar ligeros cambios al código. En particular, `t` resulta útil para depurar el funcionamiento de la subrutina `Calcula`, en donde se tenía que verificar que los distintos parámetros que la subrutina genera fuesen correctos. El límite máximo de dos breakpoints en cualquier momento hace a `br` una opción poco eficiente para depurar cada uno de estos resultados, debido a que hay que buscar la dirección en donde se guarde el resultado en memoria principal en el archivo listado y actualizar los breakpoints. Al usar `t`, se puede correr el programa paso por paso, o varios

pasos con la ejecución de un solo comando, lo cual permite encontrar errores en el código de manera rápida.

Referencias

[1] Delgado, Geovanny. *Material del curso IE623*. Escuela de Ingeniería Eléctrica, Universidad de Costa Rica, 2021.