Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
IE0623-Microprocesadores
II ciclo 2023

Proyecto Final

RunMeter 623

Gabriel Baltodano Dormond C00906
Grupo 1

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

17 de diciembre del 2023

Índice

Índice de figuras

1. Resumen

En este proyecto se implementó la aplicación del sistema RunMeter 623, mediante la la tarjeta de training dragon 12, el desarrollo de esta aplicación se realizó a lo largo de todo el curso con el fin de entrelazar todos los conocimientos aprendidos y desarrollados en un proyecto final. El Runmeter 623, es un sistema para el despliegue de información en un velódromo. Este cuenta con una pantalla LCD, un display de 7 segmentos y dos sensores fotoreflectivos que en este caso se simulan mediante dos botones de la tarjeta. Adicionalmente, el sistema cuenta con dos interruptores para definir el modo de operación de la aplicación, leds para indicar en cual modo de operación se encuentra y un teclado para poder ingresar la cantidad de vueltas que se desean realizar, además se cuenta con un potenciómetro que permite al operario variar el brillo de la pantalla LCD. La aplicación cuenta con tres modos, el modo libre que solo despliega un mensaje en pantalla, el modo configuración que se encarga de recibir el valor de vueltas a realizar promedio del teclado matricial y el modo competencia. La funcionalidad del modo competencia de la aplicación parte de que el cada vez que el ciclista recorre la distancia entre los dos sensores, el programa deberá calcular la velocidad promedio de este y las vueltas que lleva para poder desplegarlas en pantalla cuando se esté en el modo competencia. Adicionalmente, se calculan los tiempos que le tardan al ciclista en llegar a la meta para que este pueda observar el progreso y velocidad que lleva. Además se encuentra el tiempo que le tarda en llegar a la pantalla para que cuando este pase se apaga la información desplegada en la pantalla de 7 segmentos y el sistema vuelve a calcular las variables deseadas para la próxima vuelta que este realice, el sistema debe ser capaz de determinar si la velocidad es mayor o menor a la permitida enviando un mensaje de alerta por medio de la LCD y si se terminan las vueltas este debe enviar un mensaje de fin de competencia en donde el ciclista puede ver las vueltas realizadas y la velocidad promedio de la última vuelta realizada. De este modo, se logró implementar una solución para realizar la aplicación con el hardware a disposición en la Dragon 12, con un programa en ensamblador para poder controlar las funcionalidades del hardware y obtener el funcionamiento adecuado de la aplicación.

2. Diseño de la aplicación

En esta sección se mostrara el diseño de cada tarea para la aplicación. Las estructuras de datos utilizadas para cada tarea se muestra en la Tabla 1.

Tabla 1: Estructuras de Datos

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	1006	
Tarea_PantallaMUX		
Est_Pres_PantallaMUX	1020-1021	tTimerDigito
Dsp1	1022	MaxCountTicks
Dsp2	1023	OFF (Offset Tabla Segment)
Dsp3	1024	GUIONES (Offser 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,Mask $8
EstPres_LeerPB2	103F-1040	PortPB,Mask $1
		tSupRebPB
		tShortP
		tLongP
Tarea Configurar		
Est_Pres_TConfig	1041-1042	LDConfig
ValorVueltas	1043	MinNumVueltas
NumVueltas	1044	MaxNumVueltas
Tarea Libre		
		LDLibre
Tarea Competencia		
Est_Pres_TComp	1045-1046	LDComp
Vueltas	1047	tTimerVel
DeltaT	1048	tTimerError
Veloc	1049	VelocMIn
		VelocMax
Tarea Brillo		
Esr_Pres_TBrillo	104A-104B	tTimerBrillo
		MaskSCF

VARIABLES	DIRECCIONES	VALORES
BANDERAS		
Banderas_1	1070	
ShortP1	Mask $01	
LongP1	Mask $02	
ShortP2	Mask $04	
LognP2	Mask $08	
Array_OK	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_TC5
TABLAS		
Segment	1100	
Teclas	1110	
MENSAJES		
	1200	
TABLA DE TIMERS		
	1500	tTimer1mS
Timer1mS		tTimer10mS
CounterTicks		tTimer100mS
Timer260uS		tTimer1S
Timer40uS		
Timer10mS		
Timer_RebPB1		
Timer_RebPB2		
Timer_RebTCL		
TimerDigito		
Timer2mS		
Timer100mS		
Timer_SHP1		
Timer_SHP2		
Timer1S		
TimerVel		
TimerError		
TimerPant		
TimerFinPant		
TimerBrillo		
Timer_LP1		
Timer_LP2		
Timer_LED_Testigo		

2.1. Configuración de Hardware y Programa Principal

2.1.1. Memoria de Cálculo para Configuración de Hardware

Para que el periodo de interrupción sea cada $20\mu s$ se un prescalador de 16 y se obtuvo que el valor de TC4 es:

$$TC4 = \frac{20\mu s * 24MHz}{16} = 30 \tag{1}$$

Para la frecuencia de operación del convertidor analógico digital a 600kHz se deberá señeccionar el prescalador de la siguiente manera, al despejar PRS:

$$f_s = \frac{Bus_{clk}}{2 * (PRs + 1)} \tag{2}$$

$$PRS = \frac{24MHz}{2 * 600KHz} - 1 = 19 \tag{3}$$

2.2. Memoria de Calculo para Subrutina Calcula

A continuación se presentan el calculo de los valores que se utilizaron en la subrutina calcula para poder determinar los distintos valores de las variables que se tienen que calcular.

Para poder determinar el valor de la variable deltaT, se sabe que deltaT sería la diferencia de tiempo que hay entre la detección del primer sensor y el segundo. De este modo, como los timers son decrementales y tTimerVel=100mS entonces se tiene que:

$$\mathbf{DeltaT} = 100 - TimerVel \tag{4}$$

Ahora para encontrar la velocidad promedio del ciclista en Km/h se tendría que:

$$v = \frac{d}{t} \tag{5}$$

Donde d=55m y t=TimerVeloc entonces se tendría al pasarlo Km/h

$$v = \frac{55m}{TimerVel} * \frac{1Km}{1000m} * \frac{3600s}{1h} * \frac{1}{100mS} = \frac{1980}{TimerVel} \tag{6}$$

Entonces si **FactorConv** = 1980, veloc sería igual a:

$$\mathbf{Veloc} = \frac{\mathbf{FactorConv}}{TimerVel} \tag{7}$$

Ahora para encontrr TimerPant, se sabe que la distancia entre S2 y Meta es de 200m entonces:

$$TimerPant = \frac{200}{veloc} \tag{8}$$

Como TimerPant debe estar en base de 100mS se realiza el siguiente procedimiento para obtener el **FactorConv2**, con velov en m/s:

$$TimerPant = \frac{200m}{veloc \div 3{,}6} * 10 = \frac{7200}{Veloc} = \frac{\mathbf{FactorConv2}}{Veloc} \tag{9}$$

Por ultimo para determinar el tiempo de TimerFinPant, se realiza el mismo procedimiento que la anterior solo que d = 300m, entonces se tiene que:

$$TimerFinPant = \frac{300m}{veloc \div 3{,}6} * 10 = \frac{10800}{Veloc} = \frac{\mathbf{FactorConv3}}{Veloc} \tag{10}$$

2.3. Diagrama Programa Principal

En el Programa principal primero se tiene la configuración de hardware, luego inicialización de las variables de cada tarea o subrutina, posterior a esto se inicializa la pantalla LCD y se pasa al despachador de tareas. El diseño de la rutina que se encarga de inicializar la pantalla se muestra en la Figura [1] y el resto del programa principal en la Figura [2].

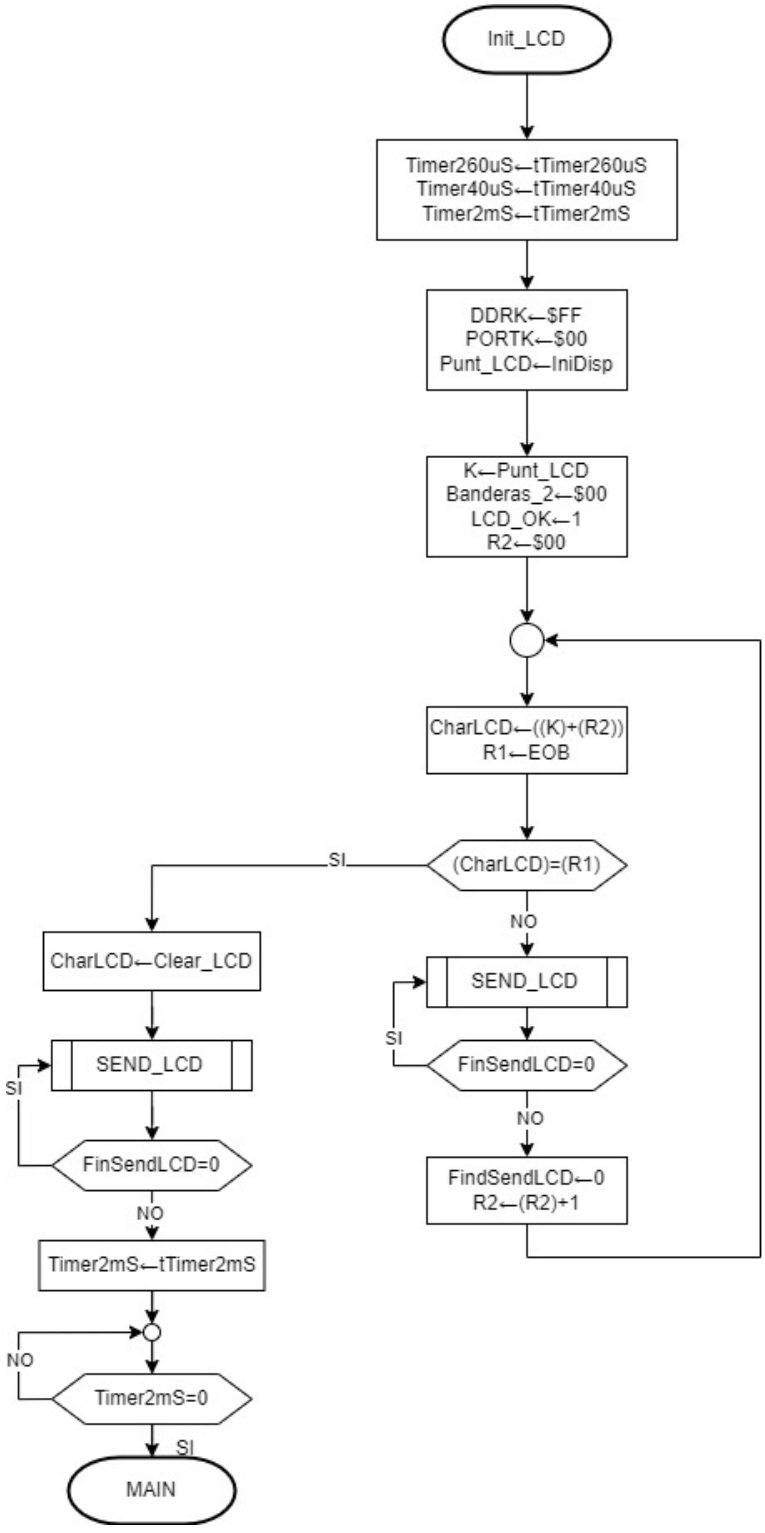

Figura 1: Diseño Rutina Init LCD

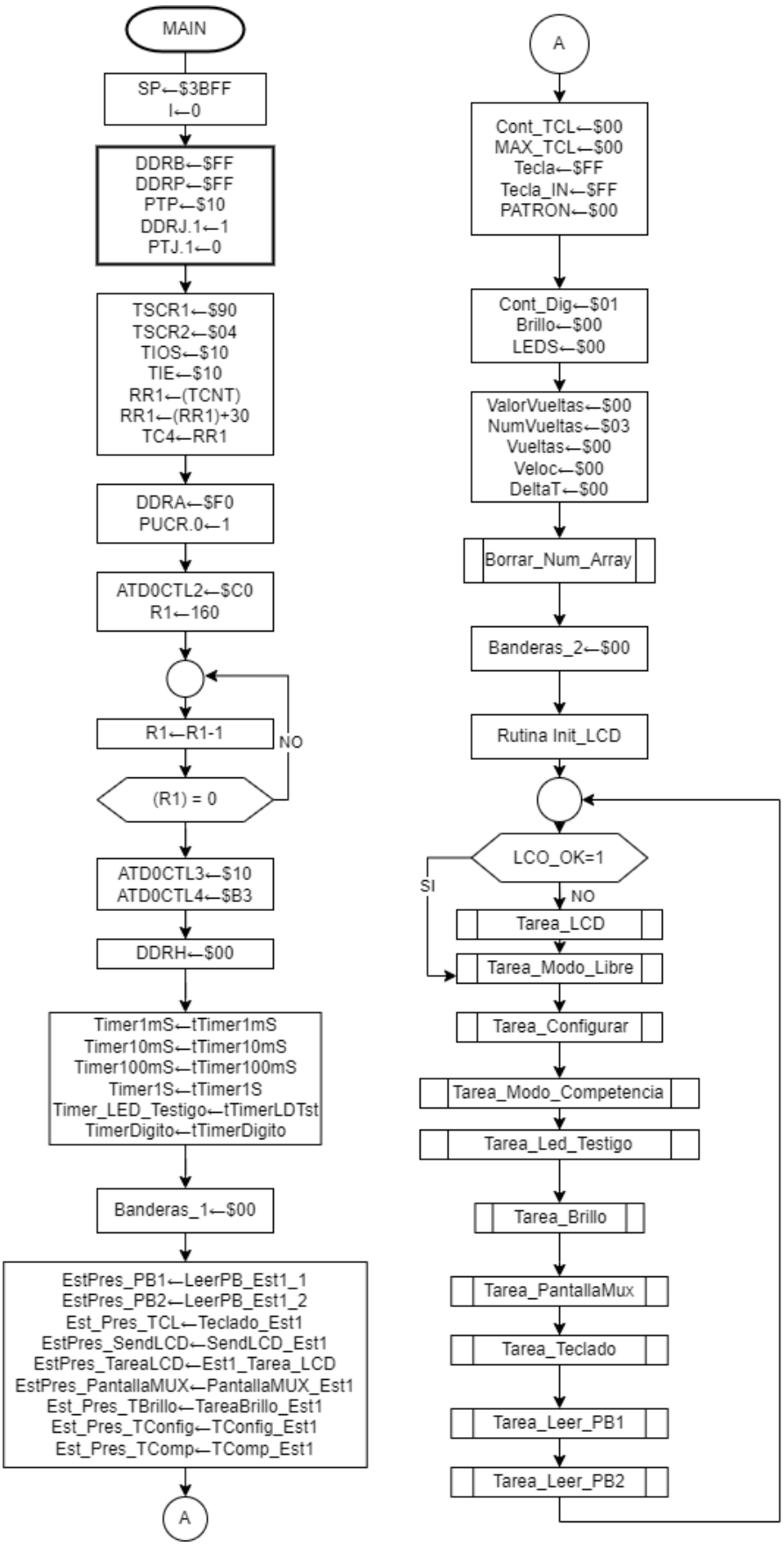

Figura 2: Diseño Programa Principal

2.4. Tarea Modo Libre

La tarea modo libre estará activa siempre que los switches PH7-PH6=0. Esta tarea se encarga de cargar el Mensaje Modo libre en la pantalla LCD y mantiene la pantalla apagada. Si se cambia el modo de los switches esta tarea no realiza ninguna función, hasta que se vuelva a poner el modo libre. El diseño de dicha tarea se muestra en la siguiente figura:

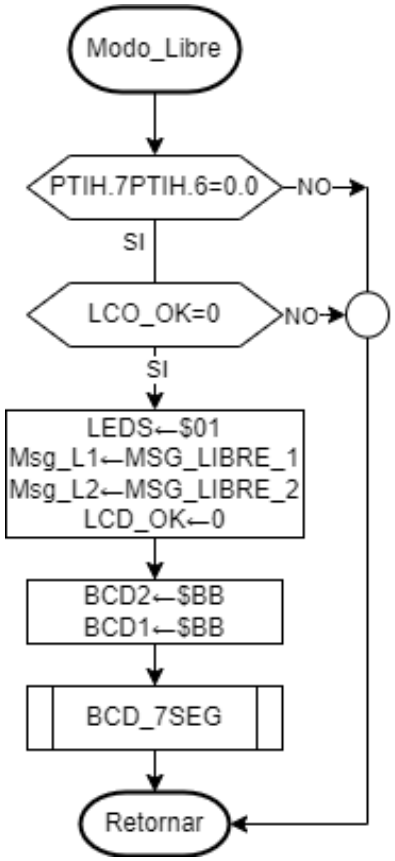

Figura 3: Diseño de Tarea Modo Libre

Vale resaltar que debido a el cambio de switches ocurre rápido en el estado 1 se agregó una etapa antes de mandar el mensaje para preguntarse si LCD OK=1 para poder enviar el mensaje completo y no solo la segunda linea, esto aplica para los tres estados iniciales de los modos tres modos de operación.

2.5. Tarea Modo Configurar

La Tarea Configurar se encarga de permitir al operario ingresar el numero de vueltas que va a recorrer el ciclista en el modo competencia. El valor lo ingresa y queda almacenado en la variable NumVueltas. El diseño se muestra en la Figura [4]. Los estados de la máquina deben realizar lo siguiente:

- **TConfig Est1**: Carga el mensaje de modo configuración en la pantalla LCD y apaga la pantalla de 7 segmentos si los switches están como PH7=0 PH6=1 y pasa al siguiente estado.
- **TConfig Est2**: Este estado se encarga de revisar si se ingresa un valor por medio del teclado matricial. Si el valor excede 25 vueltas o es menor que 3 vueltas no dejará que se ingrese el valor. Al ingresar un valor correcto se mostrara en la pantalla de 7 segmentos.

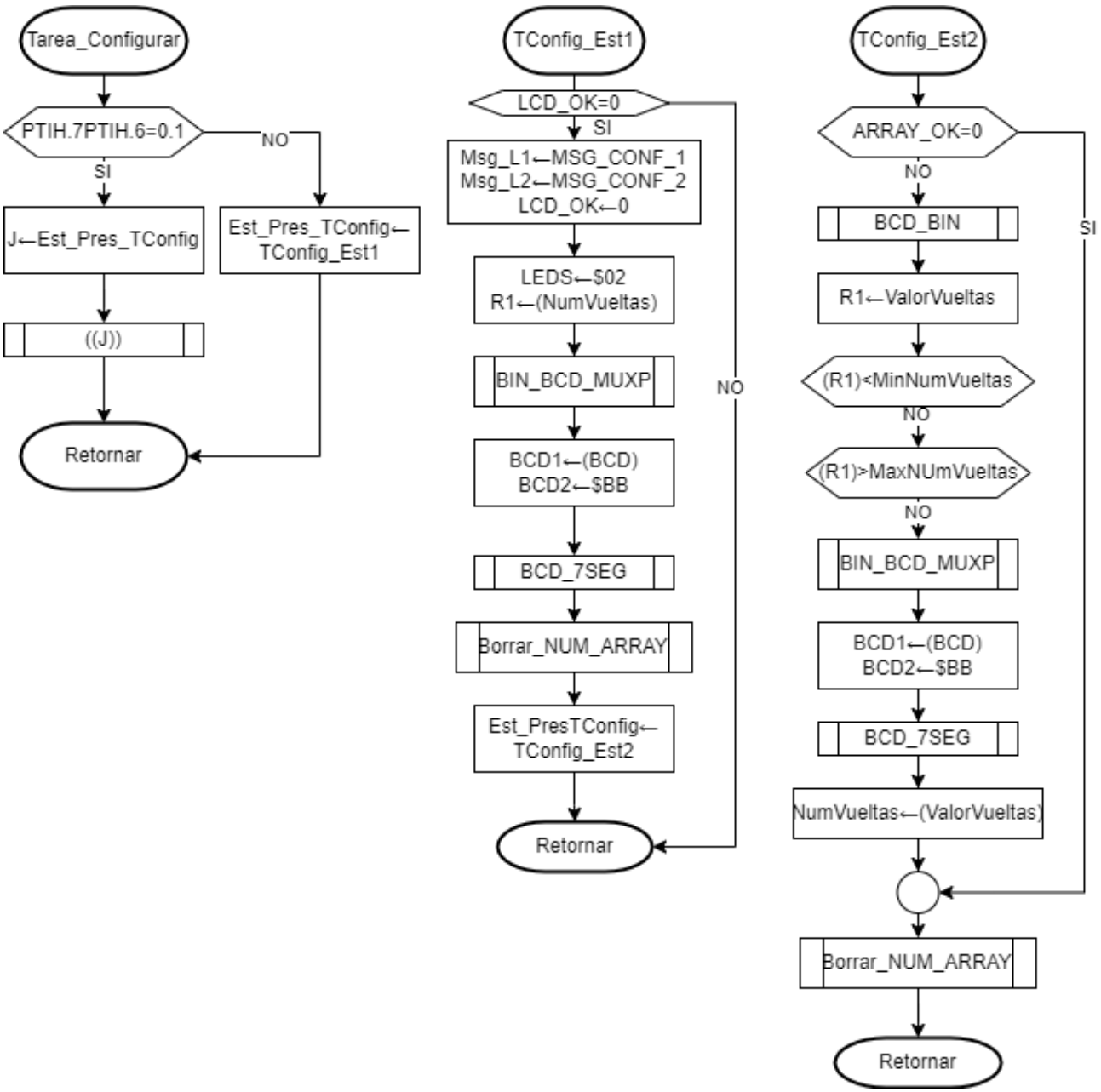

Figura 4: Diseño Modo Configurar

2.6. Tarea Modo Competencia

Esta tarea se encarga de manejar el modo competencia cuando se tienen los interruptores como PH7=1 y PH6=1. Esto esta definido como una maquina de 7 estados que su diseño se puede observar en la Figuras [5], [6] y [7]. En donde cada estado realiza la siguiente función:

- **TComp Est1**:El estado 1 se encarga de cargar el mensaje inicial del modo competencia y mantiene las pantallas de 7 segmentos apagadas. Si la cantidad de vueltas se cumplen este estado cargará el mensaje de fin de competencia y pasara al estado tres, de lo contrario pasará al estado 2.
- **TComp Est2**: El estado 2 se encarga de revisar si ya paso el ciclista por S1 si no pasa se queda en el mismo estado, al pasar por S1 pasa al estado 4.

- **TComp Est3**: El estado 3 se encarga de borrar el numero de vueltas ingresado y vueltas realizadas al finalizar la competencia.
- **TComp Est4**:El estado 4 se pregunta si ya el ciclista paso por S2, de este ser el caso calcula la velocidad del ciclista y los tiempos que le tarda en llegar a la meta y a la pantalla del velódromo, así como verificar que la velocidad esté dentro del rango permitido. De se este el caso pasa al estado 5. Si la velocidad no es la permitida carga el mensaje de Alerta y pone las rayas en los 7 segmentos y pasa al estado 6
- **TComp Est5**: En este estado se carga el mensaje de competencia en la LCD y las vueltas y velocidad en la pantalla de 7 segmentos, esto al concluir el tiempo que el tardó al ciclista recorrer los 200m entre S2 y la meta, al pasar el tiempo pasa al estado 7.
- **TComp Est6**: Este estado revisa si ya pasó el tiempo de error al darse la alerta de velocidad, cuando se acaba pasa al estado 1
- **TComp Est7**: Al llegar al estado 7, este se pregunta si ya el ciclista pasó por la pantalla del velódromo al ocurrir este evento, se devuelve al estado 1.

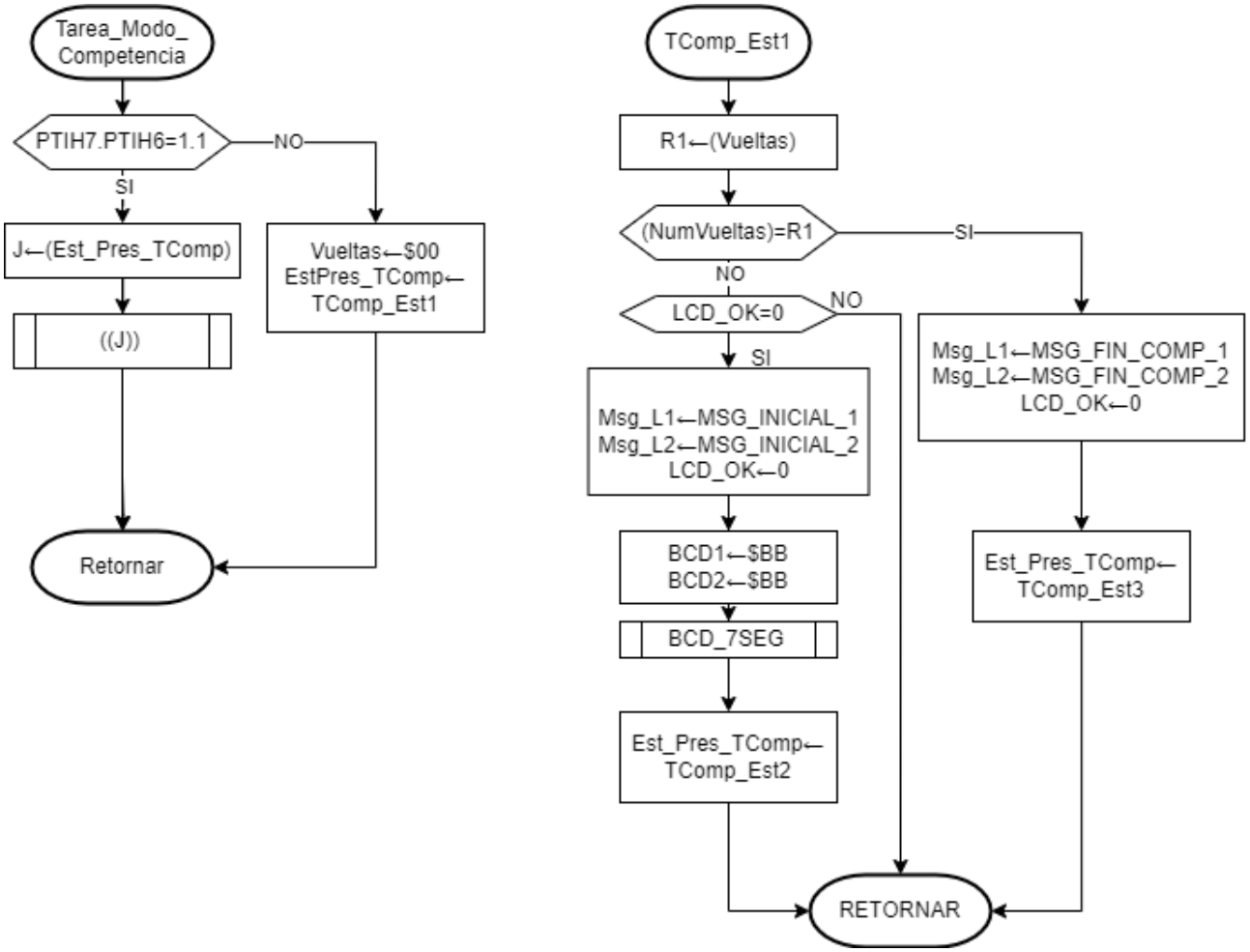

Figura 5: Diseño de Tarea Modo Competencia Estado 1

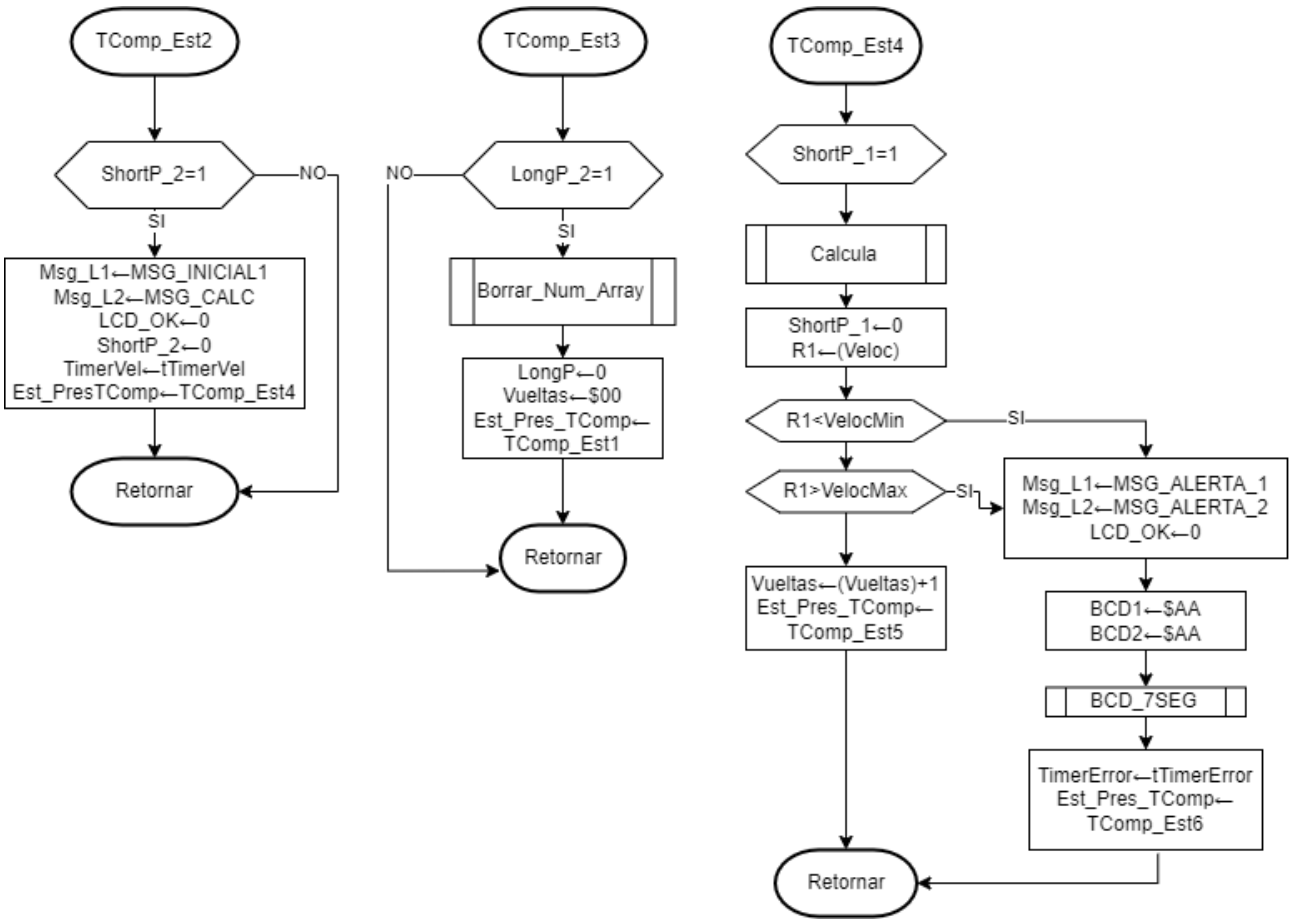

Figura 6: Diseño de Tarea Modo Competencia Estados 2 a 4

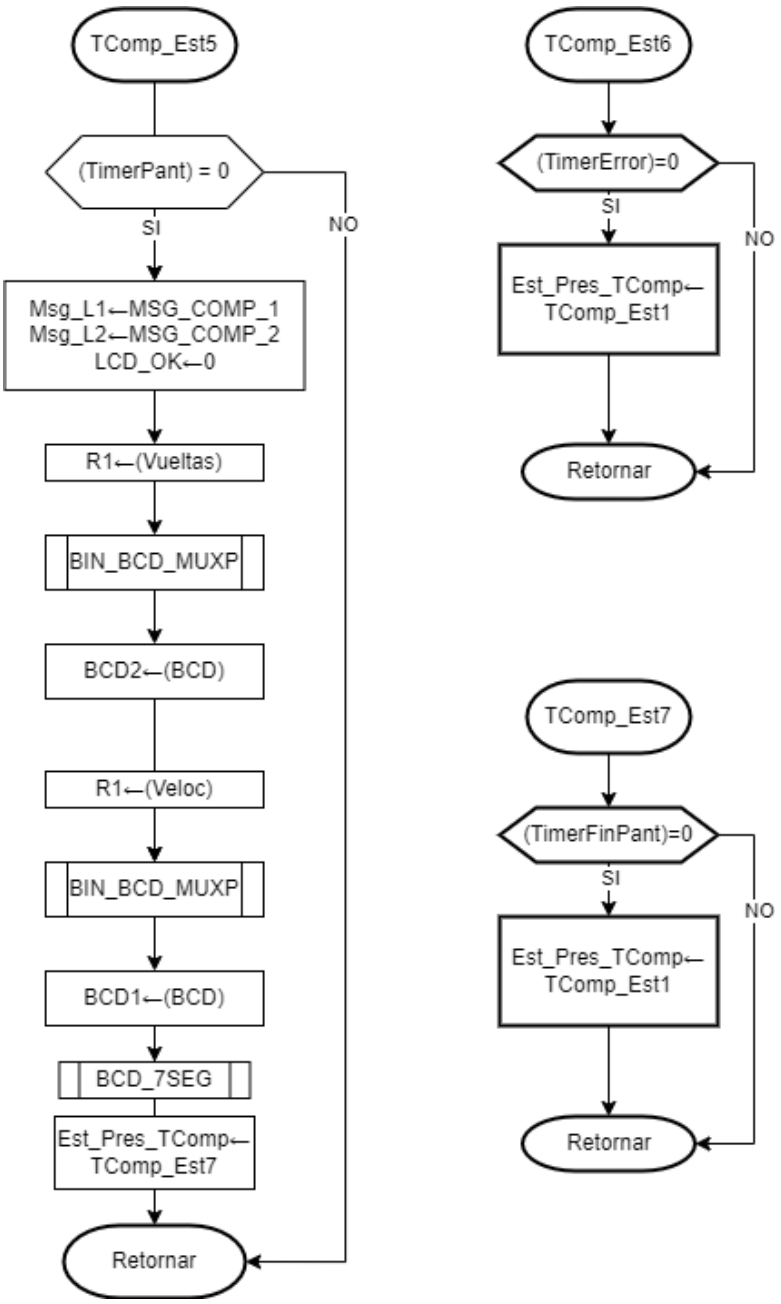

Figura 7: Diseño de Tarea Modo Competencia Estados 5 a 7

2.7. Tarea Brillo

Esta tarea utiliza el convertidos analógico digital con el potenciómetro del PAD7 para encontrar el valor del brillo de los displays de 7 segmentos. Este esta configurado para realizar 2 conversiones con 4 periodos de reloj y una frecuencia de operación de 600kHz. El diseño de los tres estados que conforman esta máquina se muestran en la Figura 8. La función de cada uno se describe a continuación:

- **TComp Est1**: En este estado se carga el TimerBrillo para que se realice un ciclo de conversión cada 400mS y se pasa al estado 2, vale resaltar que se disminuyo el periodo a 100mS para que al variar el potenciómetro el cambio en el brillo sea más suave.
- **TComp Est2**: En este estado se espera a que pase el tiempo de TimerBrillo para inicializar el ciclo de conversión, luego pasa al estado 3.

- **TComp Est3** En este estado se calcula el valor promedio leído en el convertidor analógico digital y se guarda en la variable Brillo.

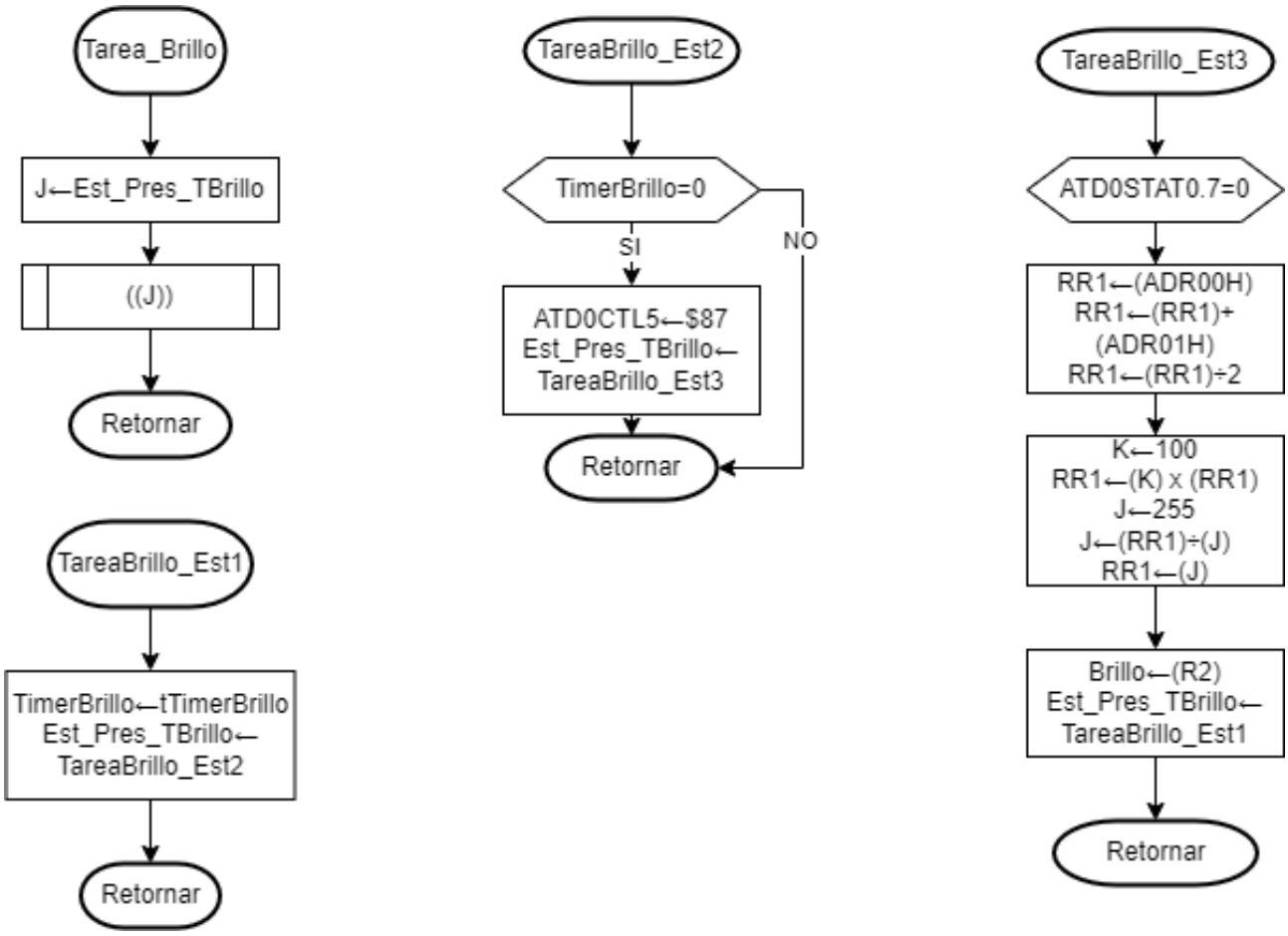

Figura 8: Diseño Tarea Brillo

2.8. Tarea Teclado

La Tarea Teclado es una máquina de estados que se encarga de leer el teclado matricial presente en la Dragon 12. Esta tarea permite que el usuario ingrese valores al programa en tiempo de ejecución. El teclado posee los números del cero al nueve y un botón de borrado y enter. Los valores ingresados se guardan en un Array llamado Num Array. El diseño de esta tarea viene presente en las Figuras [9] [10] y [11].

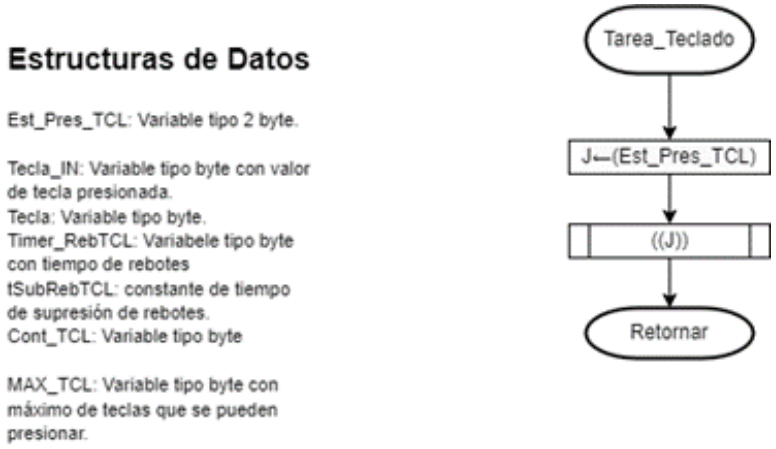

Figura 9: Diseño de Tarea Teclado

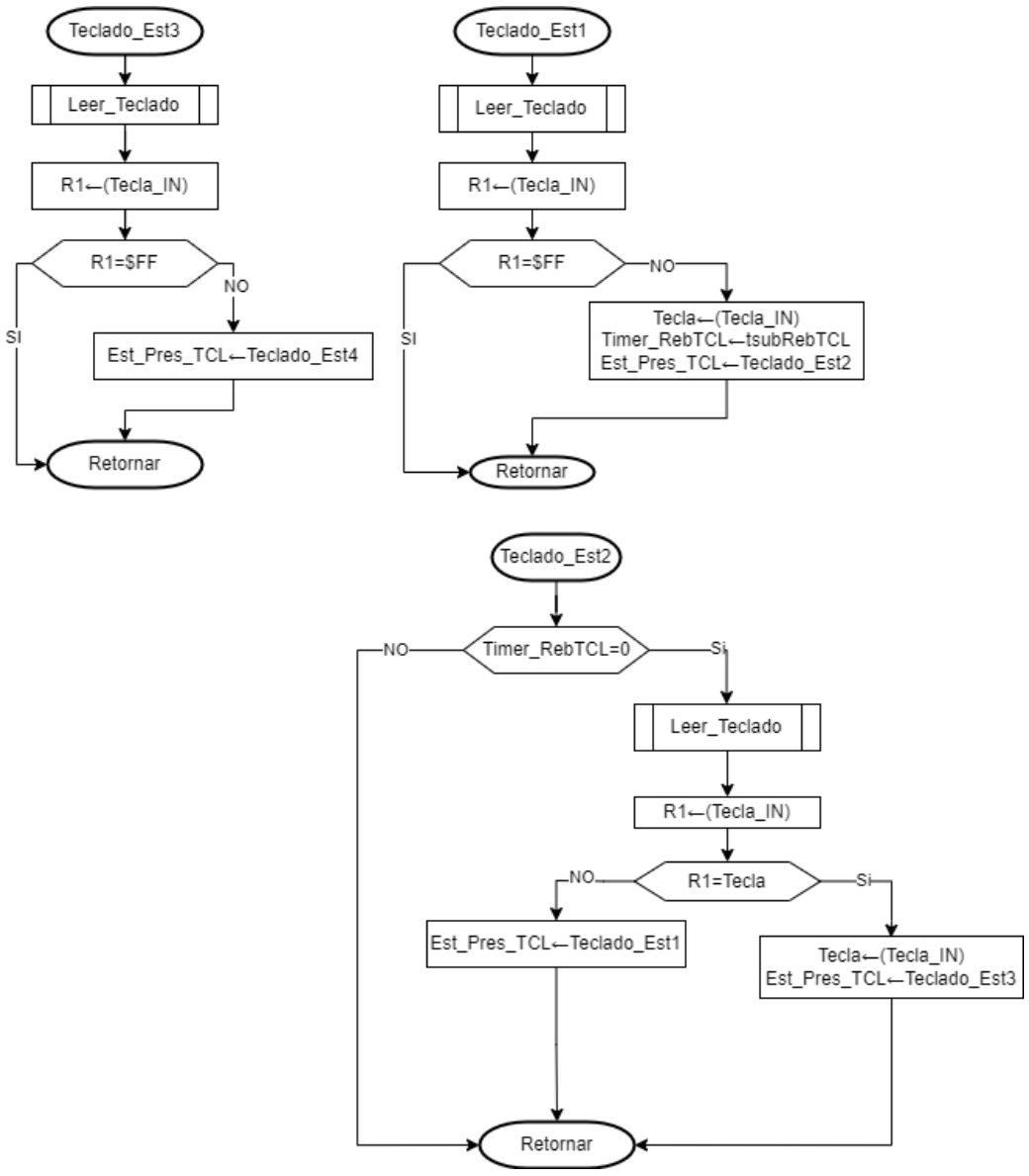

Figura 10: Diseño de Tarea Teclado: Estados 1,2 y3

De la Figura 10 se observa que el estado uno se encarga de verificar si se presionó alguna tecla al realizar el llamado a la subrutina leer teclado. Si el valor es distinto de $FF implica que se presiono un botón del teclado y se pasa al estado 2 donde se verifica que se haya leído un valor y no haya sido ruido, si es ruido pasa al estado uno, si no es ruido pasa al estado 3 con el fin de esperar a que se suelte la tecla para evitar roll over si ya se deja de presionar la tecla se pasa al estado 4 que esta presente en la Figura [11] y esta se encarga de realizar las acciones de borrado y enter.

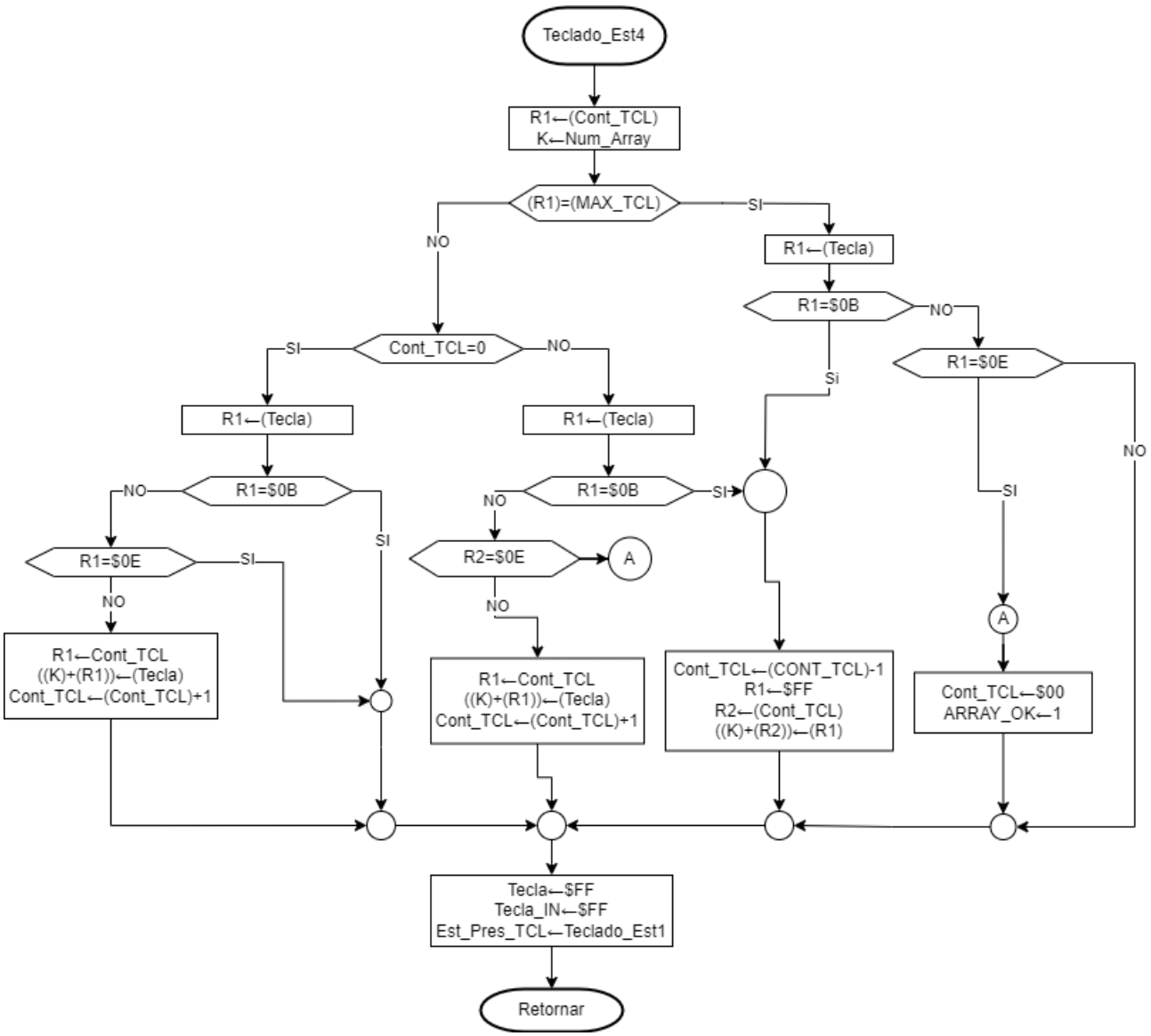

Figura 11: Diseño estado cuatro de Tarea Teclado

Vale destacar que esta tarea llamada a la subrutina Leer Teclado la cual tiene el diseño realizado en la tarea 4 que se observa en la Figura 12 y se encarga de leer el teclado de manera multiplexada leyendo fila por fila viendo si se presiona alguna de las teclas y guardando el valor en tecla_IN.

Estructuras de Datos

Teclas_IN: Variable tipo byte contiene valor de tecla ingresada
Teclas: contiene dirección base del arreglo teclas
PATRON: Variable tipo Byte

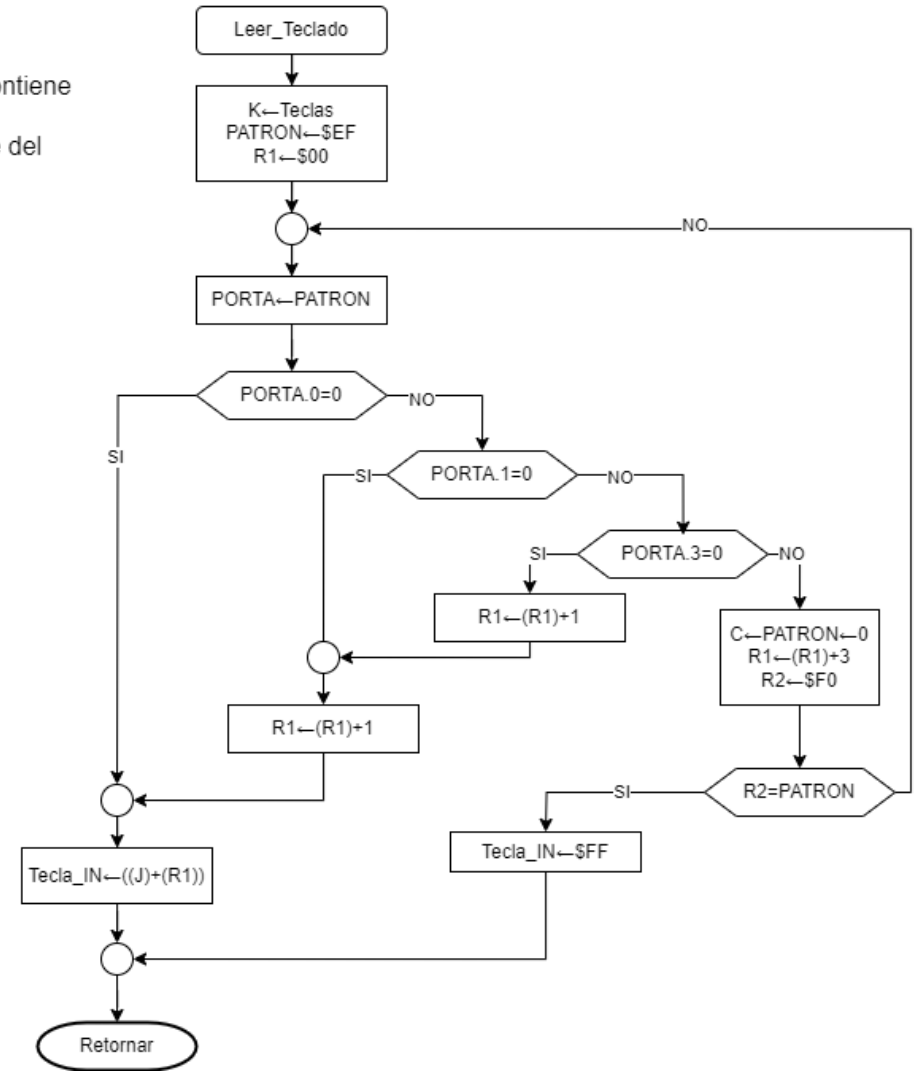

Figura 12: Diseño de Subrutina Leer Teclado

2.9. Tarea Led Testigo

Esta Tarea se encarga de barrer el Led RGB que esta conectado al puerto P en los pines cuatro, cinco y seis. Variando el color cada segundo, para indicar que el programa esta corriendo en todo momento. El diseño de esta tarea se muestra en la Figura [13].

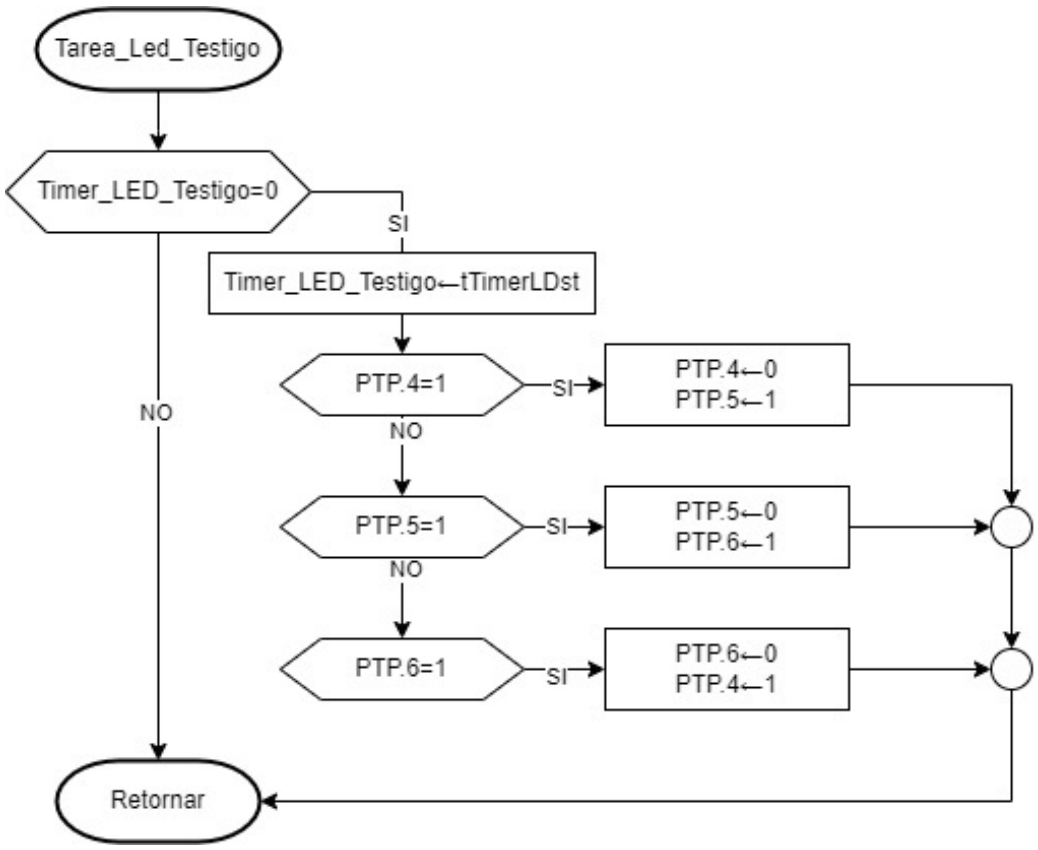

Figura 13: Diseño de tarea Led_Testigo

2.10. Tarea Leer PB1 y Tarea Leer PB2

Esta Tarea fue diseñada en clase, para un botón. Ahora que se tienen dos botones se definen los siguientes valores para poder leer la entrada de PH0 que es el S2 y PH3 que es el sensor PH0. Vale resaltar que a todas la estructuras de datos se les agrego un índice 1 o 2 dependiendo de el sensor que revisa cada tarea.

- MaskPB = $01
- MaskPB_2 = $08

Las dos Tareas tienen el mismo diseño funcional para sus estados la diferencia son las banderas y el el pad de PH que revisa como entrada. De este modo se definen los siguientes estados:

- **LeerPB Est1**: Se lee si se presiona PB ya sea PH3 o PH0, se carga el contadores de rebotes y los contadores de Short Press y Long Press y se pasa al estado 2. De no presionarse ningún botón se queda en Est 1.
- **LeerPB Est2**: Se espera a que se acabe el tiempo de supresión de rebotes,valida si el botón sigue presionado, si esta inactivo se devuelve al estado 1, si PB sigue activo pasa al estado 3.
- **LeerPB Est3**: En este estado se verifica si al presionar PB se dio un Short Press, de ser este el caso se levanta la bandera de Short Press si PB sigue presionado pasa al estado 4.
- **LeerPB Est4**: Se espera a que termine el Timer LongP y a que se libere PB para levantar bandera de LongP y se regresa al estado 1. Si se libera antes de Timer LongP termine se toma como un Short Press.

Esta Descripción de estados aplica tanto para Tarea Leer PB1 y Tarea Leer PB2 como se observa en los diagramas de las 14 y 15

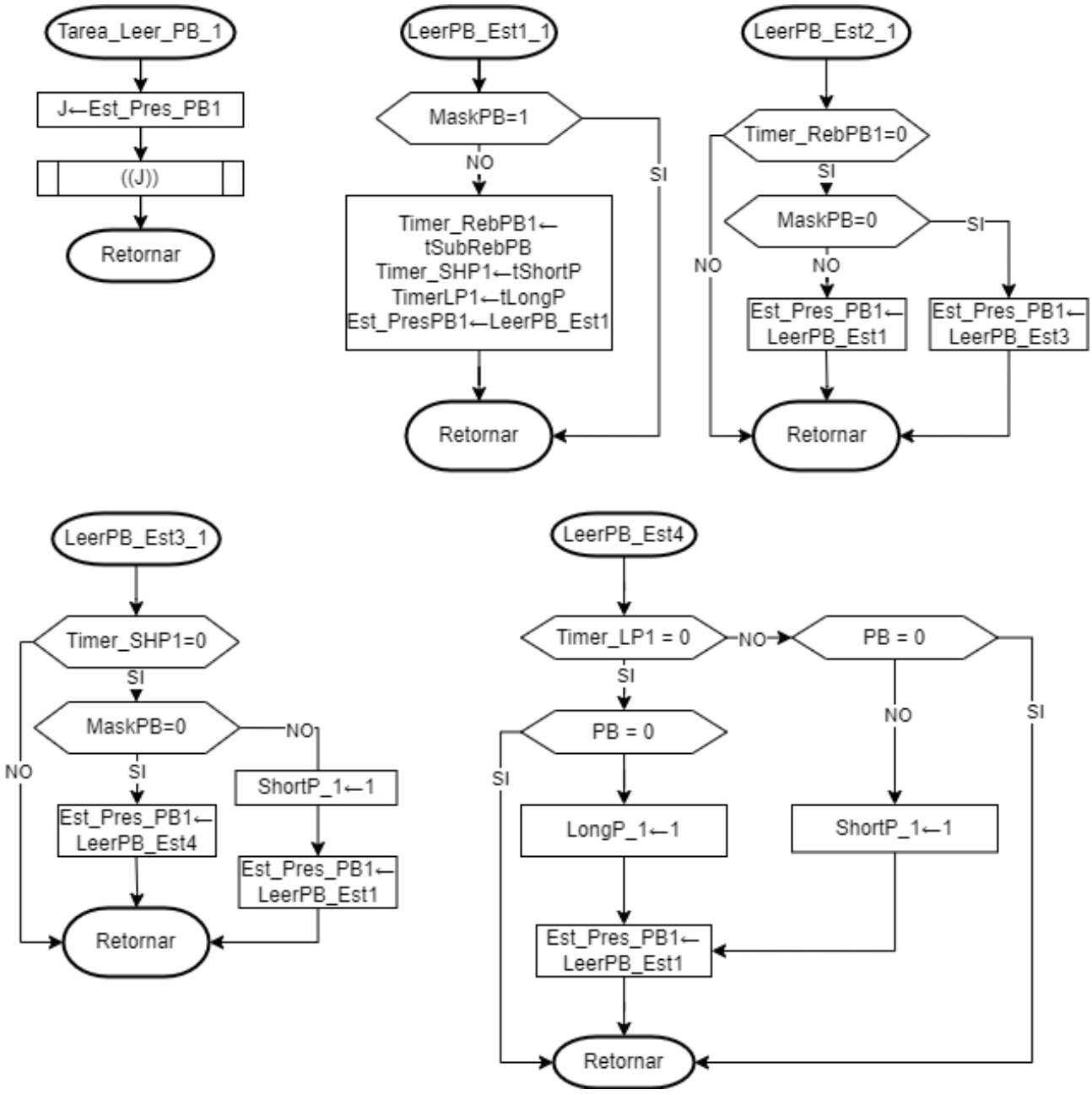

Figura 14: Diseño de Tarea PB1

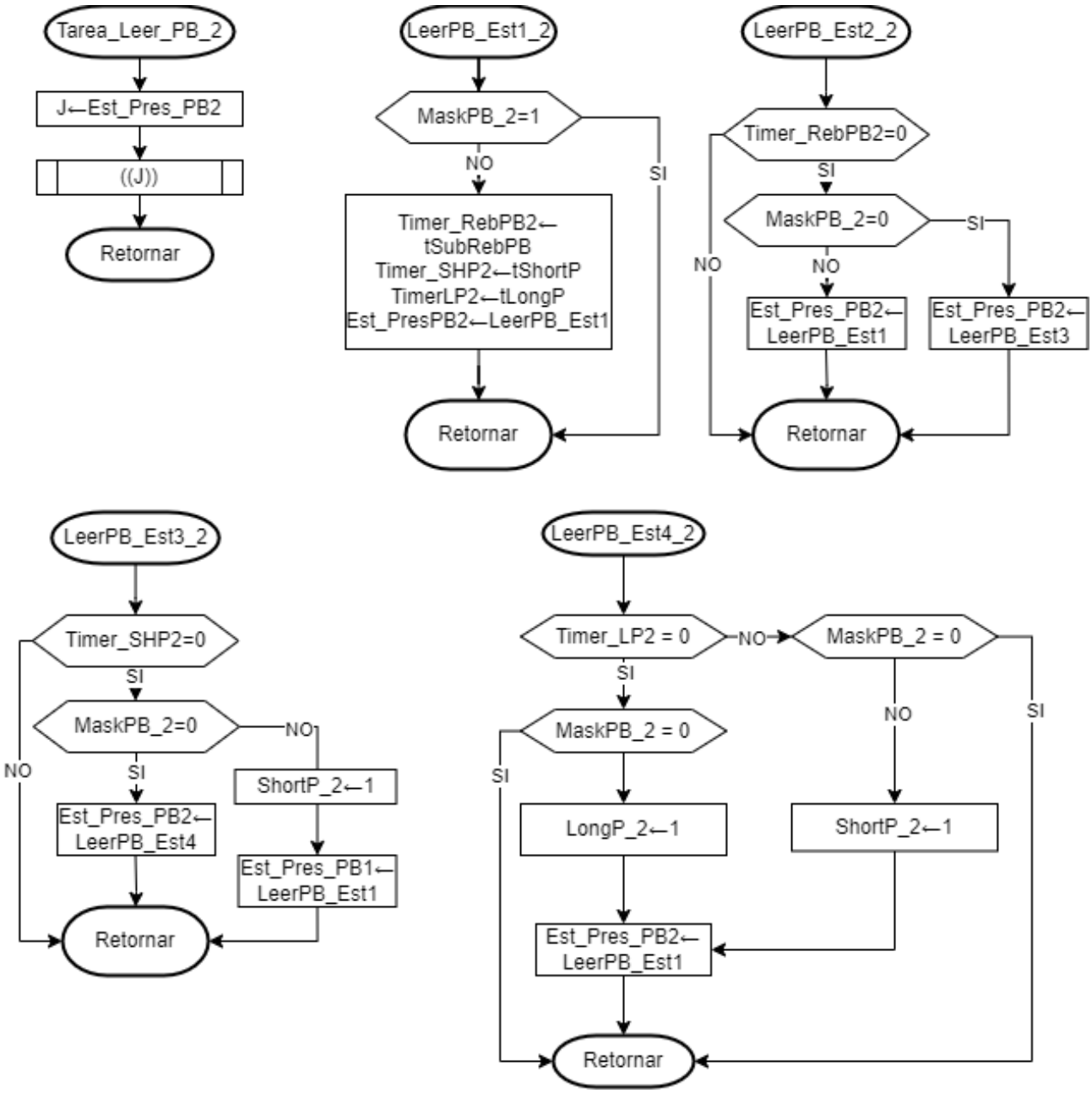

Figura 15: Diseño de Tarea PB2

2.11. Tarea PantallaMUX

Esta Tarea es una máquina de estados se encarga de multiplexar los displays de 7 segmentos para poder desplegar valores numéricos en estas pantallas. El estado 1 enciende un display a la vez los cuales están conectados a los pines 0,1,2,3 del puerto P y carga el patrón del dígito en el puerto B proveniente de la las variables DSP1, DSP2, DSP3 y DSP4 para poder observarlo en la pantalla, además de esto el estado dos permite definir el brillo de los leds y el display de 7 segmentos mediante la variable brillo, la cual, en el caso de esta aplicación será controlada mediante el potenciómetro de la Dragon 12, además esto define el tiempo que el display estará encendido. El diseño de esta tarea se muestra en la Figura 16.

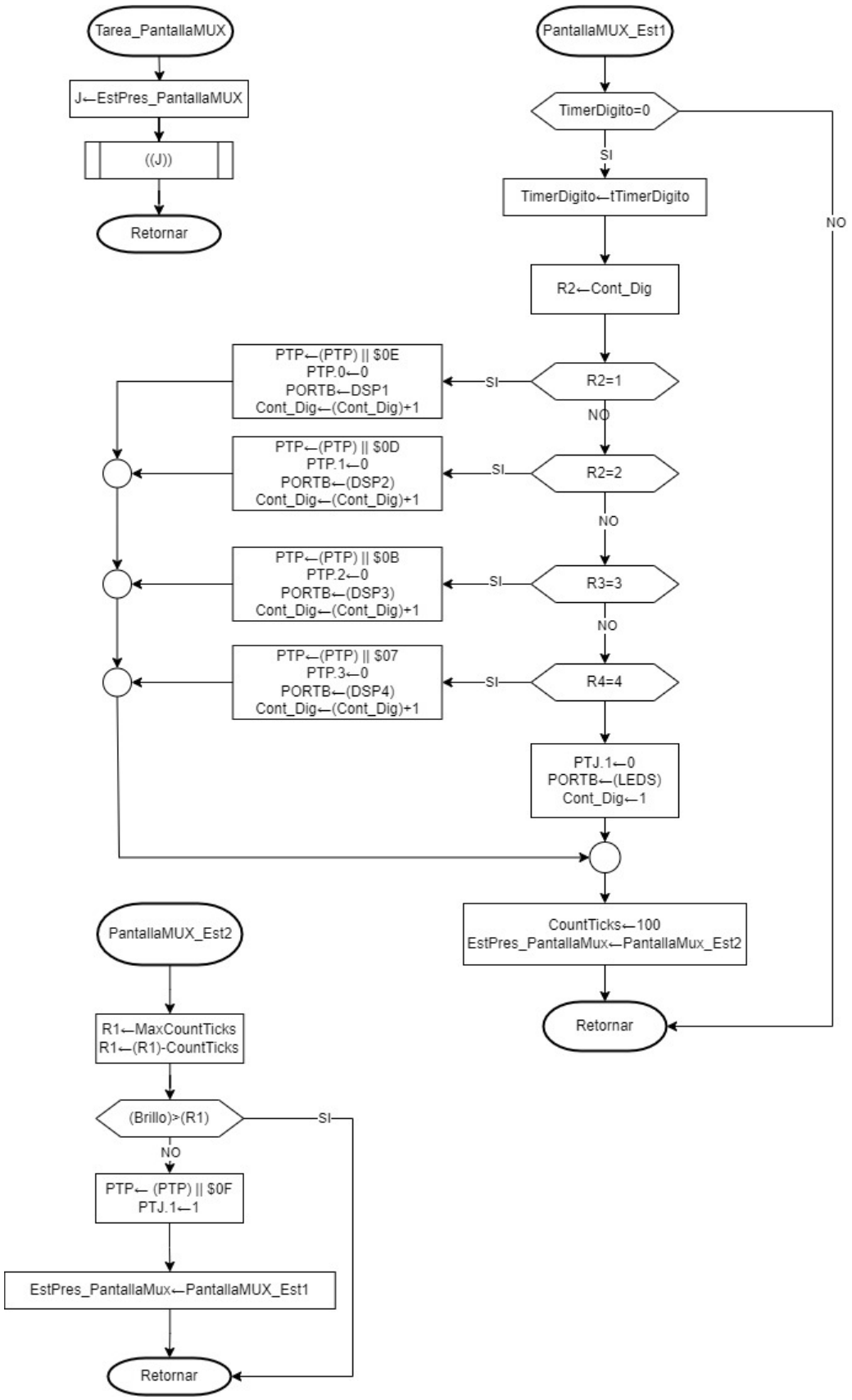

Figura 16: Diseño de Tarea Pantalla Mux

2.12. Tarea LCD y Send LCD

La Tarea LCD, es una máquina de dos estados que se encarga de cargar los mensajes de la linea uno y dos de la pantalla LCD y por medio de la Tarea SendLCD envía el mensaje a la pantalla LCD para que sea desplegado en la pantalla.

- El Primer estado verifica en cual linea se está escribiendo en la LCD carga la dirección del mesnaje dependiendo de la linea y se llama a Send LCD, luego se pasa al estado 2
- Este estado verifica que ya se envió todo el mensaje por la LCD tanto la linea 1 como la linea 2 y se mantiene en este estado hasta que se termine de enviar el mensaje que se quiere desplegar en cada linea, si se envían las dos lineas también levanta la bandera de LCD OK y se devuelve al estado 1.

El diseño se observa en la Figura [17]. Como parámetros de entrada recibe las direcciones del mensaje mediante las variables MSG _ L1 y MSG _ L2 y la salida la entrega a la pantalla LCD.

Ahora bien la Tarea SendLCD es la que se encarga de realizar el protocolo estroboscópico para poder enviar un byte de contenido a la pantalla LCD. Este byte lo envía mediante la variable CharlCD. Los datos se envían mediante el puerto K de la dragon 12. El diseño de los 4 estados de la máquina de estados se observan en la Figura [18]. Los Estados de esta Tarea se definen a continuación:

- El primer estado se encarga de enviar el nibble de la parte alta de lo que se desea enviar en charLCD además revisa si lo que se esta mandando es un dato o un comando, se carga el tiempo de 260 μs y se pasa al estado 2.
- En el estado dos se espera a que se termine el tiempo de 260 μs, al terminar se pasa el nibble de la parte baja a la LCD donde al igual que en el estado 1 se revisa si es un comando o un dato para poder indicarselo a la pantalla por medio del PORTK.0, se cargan los 260 μs y se pasa al estado 3.
- Al finalizar el tiempo de 260μs se carga el timer de 40μs y se pasa al estado 4
- En este estado se espera a finalizar el tiempo de 40 μs se devuelve al estado 1.

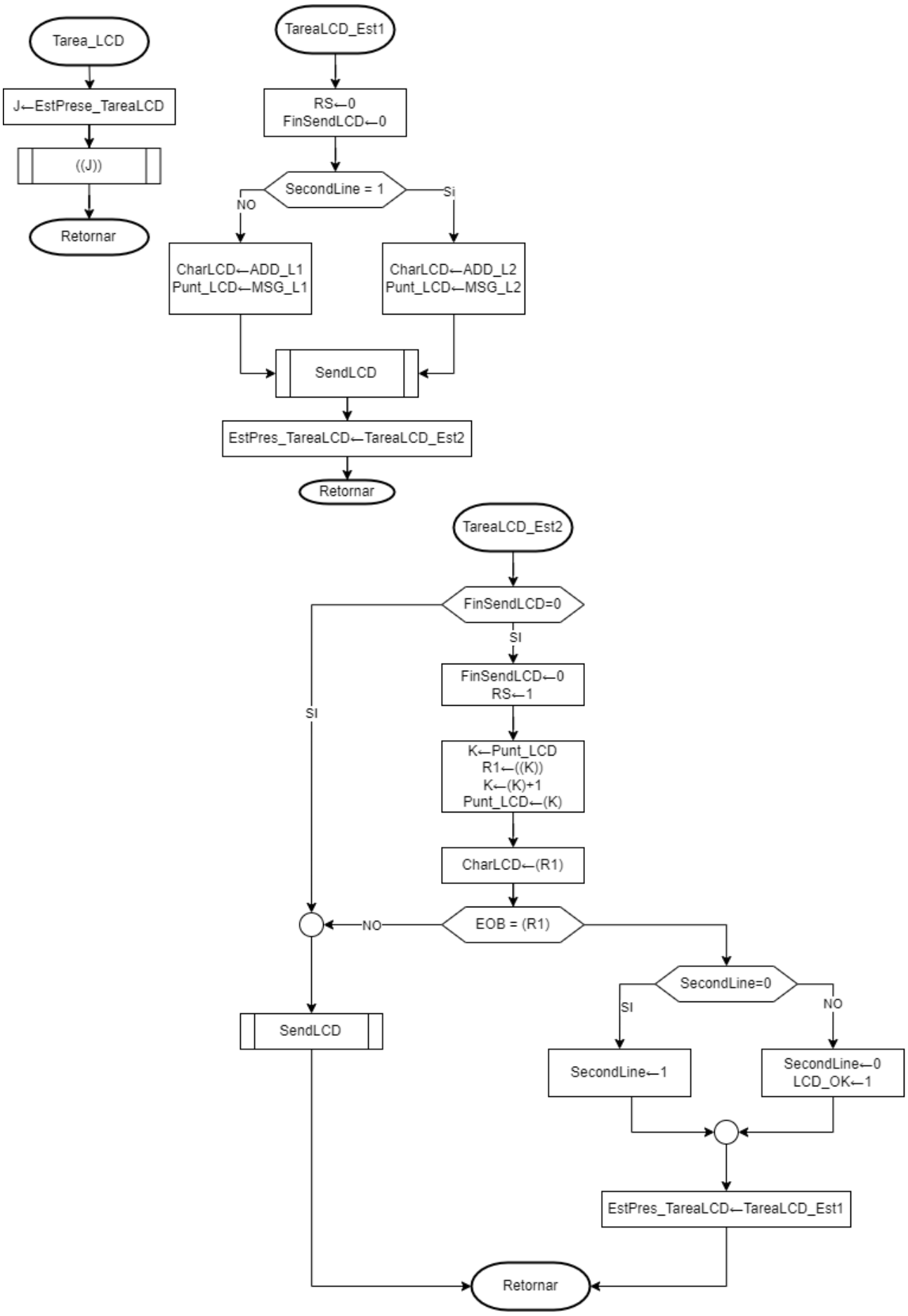

Figura 17: Diseño de la Tarea LCD

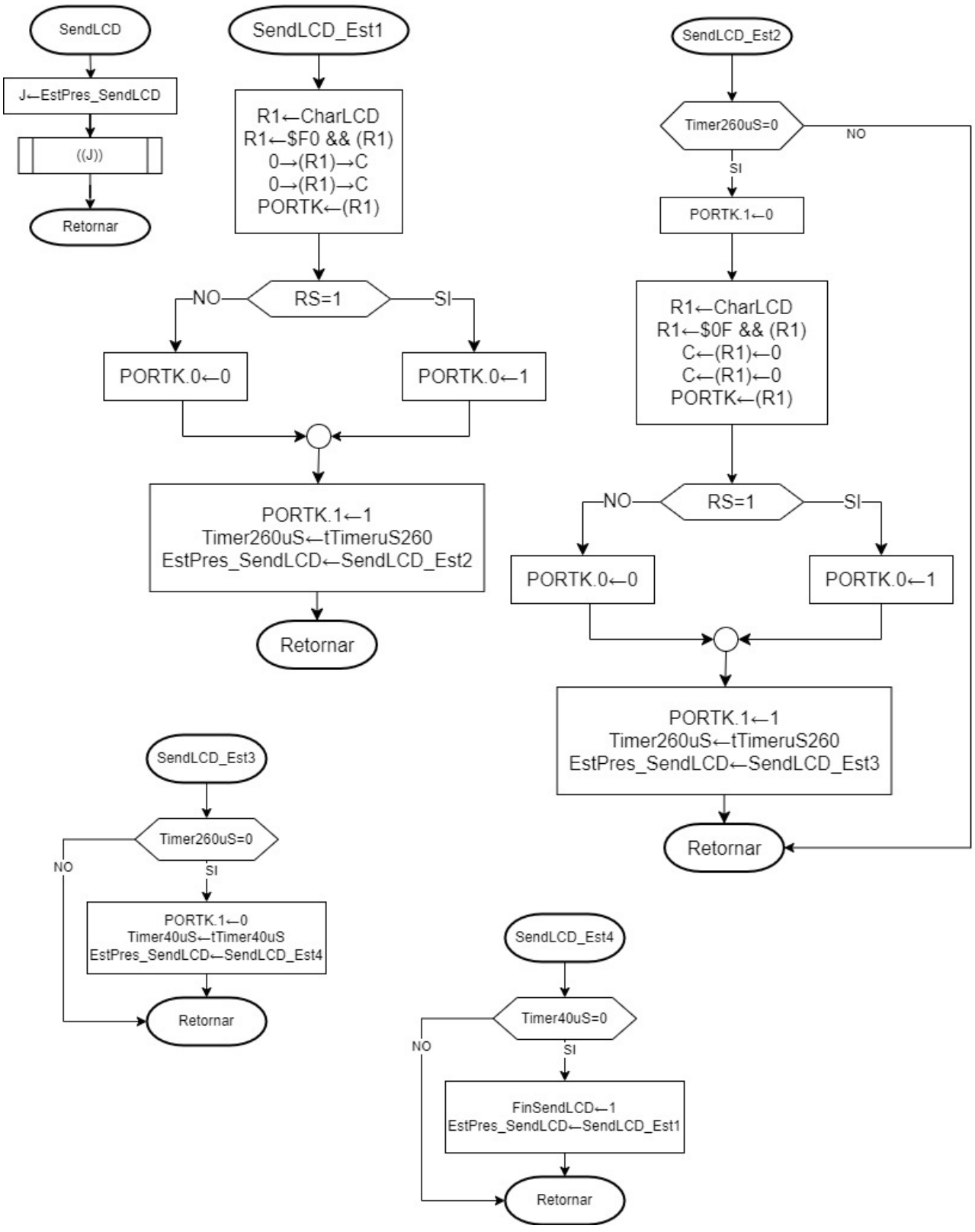

Figura 18: Diseño de Tarea SendLCD

2.13. Subrutina BCD-BIN

Esta Subrutina se encarga de de recibir la dirección de Num Array y convierte el valor presente en este array de BCD a binario guardándolo en la variable ValorVueltas. El diseño se observa en la figura 19.

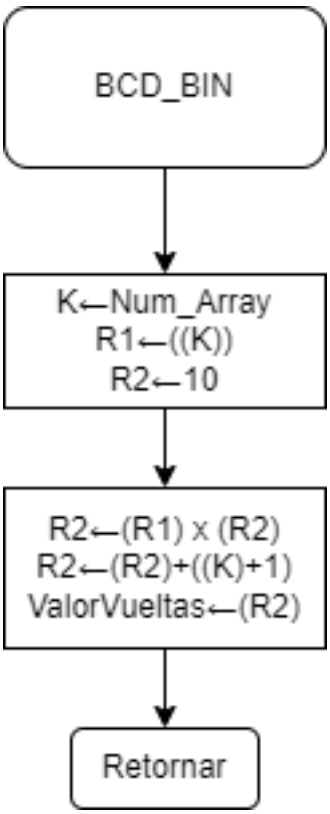

Figura 19: Diseño de Subrutina BCD-BIN

2.14. Subrutina Calcula

Esta subrutina busca calcular las ecuaciones descritas previamente en la memoria de cálculo para encontrar las variables DeltaT, Veloc, TimerPant y TimerFinPant en sus respectivas unidades a partir del párametro TimerVel. De este modo, se definieron 3 valores que se utilizaron en el diseño de la Figura **??**.

- FactorConv: Valor para encontrar Veloc.
- FactorConv2: Valor para encontrar TimerPant.
- FactorConv3: Valor para encontrar TimerFinPant.

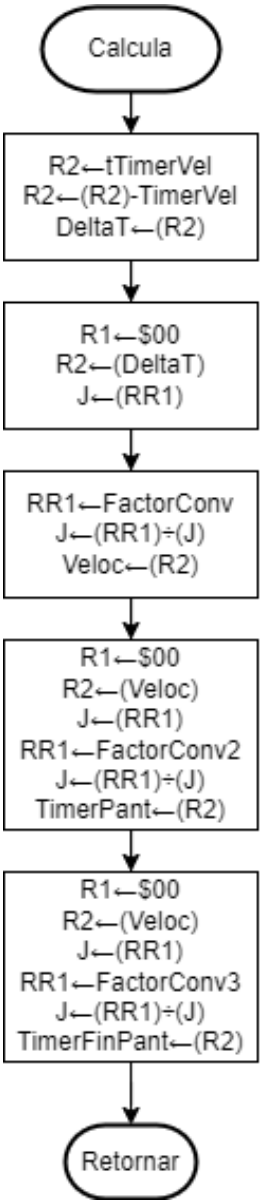

Figura 20: Diseño de subrutina calcula.

2.15. Subrutina BIN-BCD-MUXP

Esta subrutina se encarga de convertir números binarios a números en BCD. El numero binario a convertir lo recibe por medio del acumulador A. El número en BCD obtenido lo entrega por medio de una variable llamada BCD. El diseño de esta subrutina se encuntra en la Figura 21.

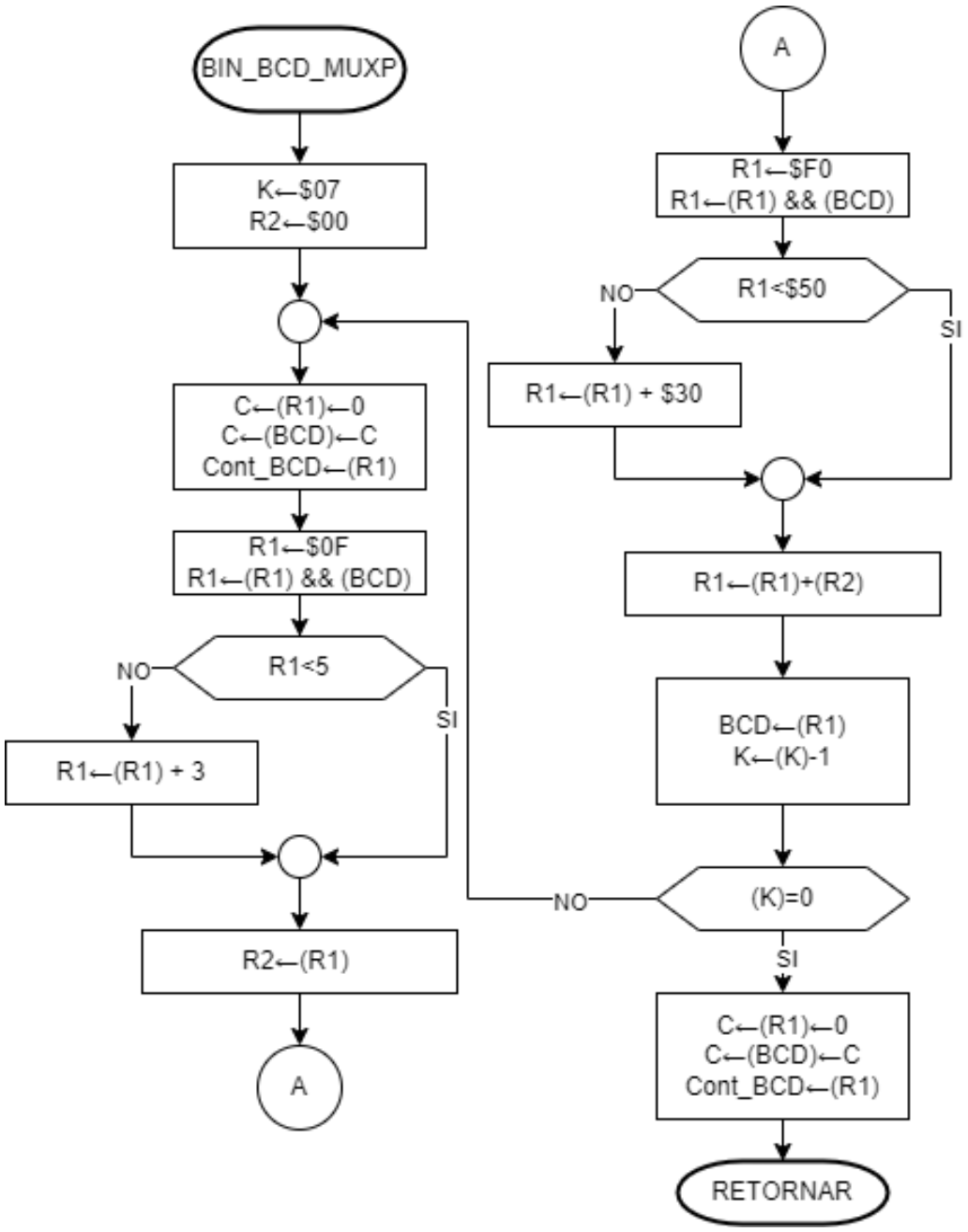

Figura 21: Diseño de Subrutina BIN-BCD-MUXP

2.16. Subrutina BCD-7SEG

Esta subrutina se encarga de cargar los valores en BCD para desplegar en los displays de 7 Segmentos. Los valores de entrada los recibe por medio de las variables BCD1 y BCD2 y el valor de salida lo envía a por medio de las variables DSP1, DSP2, DSP3 y DSP4 al cargar el valor de la tabla Segment mediante direccionamiento indirecto indexado por acumulador A. El diseño se muestra en la Figura 22

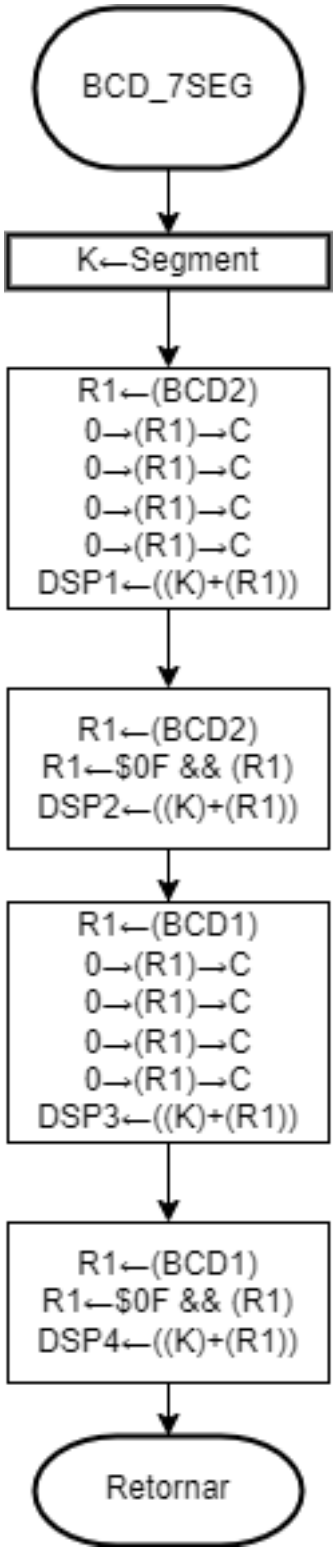

Figura 22: Diseño Subrutina BCD-7SEG

2.17. Subrutina Borrar NumArray

Esta Subrutina se encarga de borrar los valores en NumArray con $FF. Recibe el parámetro de entrada la dirección de Num Array y lo carga en el índice X, borra los valores uno por uno hasta alcanzar el total números ingresados en el array estan almacenados en MAX _TCL y retorna el array borrado por la misma dirección. El diseño se muestra en la Figura 23.

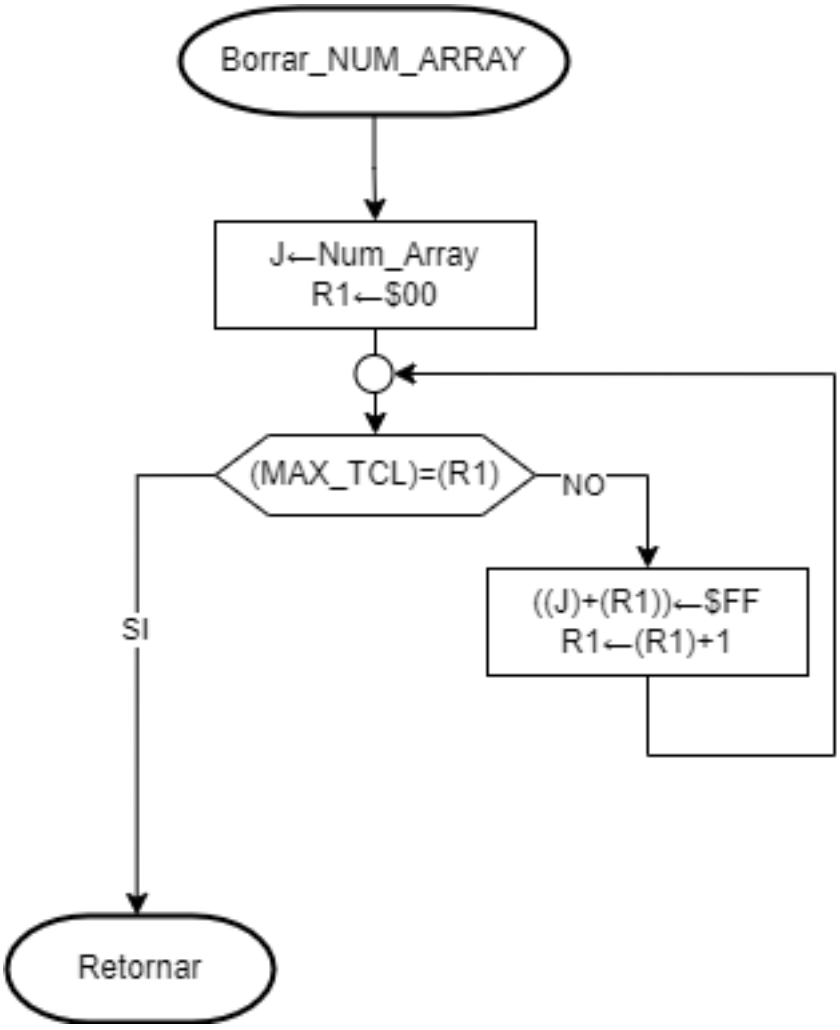

Figura 23: Diseño Subrutina NumArray.

2.18. Máquina de Tiempos

Esta subrutina atención a interrupción de output compare al canal 4 del puerto T se encarga de producir tiempos en el orden Base de $20\mu S$ 1mS, 10mS, 100mS y 1S. De este modo permite generar timers para realizar la temporización que sea al realizar una acción en el programa o cambiar de estado. El diseño se presenta en la Figura [24]. Sus parámetros de entrada son la tabla de tiempos y para que de este modo pueda descontar el tiempo en las posiciones de memoria de la tabla.

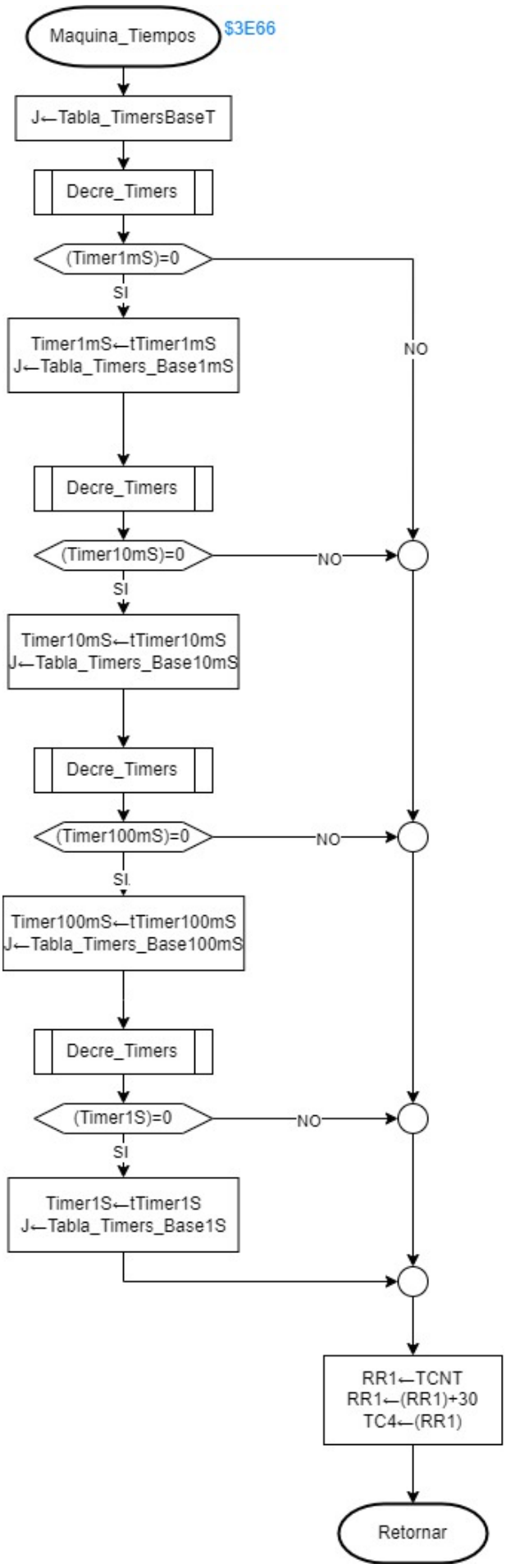

Figura 24: Diseño Maquina de tiempos

3. Conclusiones

- La Dragon 12 permitió realizar la aplicación sin ninguna complicación y demuestra que es una herramienta útil para aprender a utilizar este tipo de sistemas.
- Desarrollar una aplicación requiere tiempo y esfuerzo pero es algo que si se es ordenado y se trabaja con esfuerzo se puede realizar.
- Me gustó más trabajar con CISC que con RISC, esto es una conclusión a la que se llega tras comparar el curso de estructuras de computadores y el curso de microprocesadores.

4. Recomendaciones

- Ser bastante ordenado a la hora de trabajar, esto implica siempre realizar primero el diagrama de flujo, después codificar y luego el debugging. Si se realiza primero la codificación nada sale bien.
- Ir probando una tarea a la vez para ver que todo el programa funcione y si hay un error será más fácil de encontrarlo.
- Si hay errores saber utilizar los comandos del debug 12 para poder darse una idea adonde están estos errores y poder identificarlos con velocidad ya que este proceso puede durar bastante.
- Si no encuentra el error en algún caso, seguir con las otras partes del proyecto y volver después a solucionarlo.
- Empezar el proyecto con tiempo y así se estará menos estresado y permite pensar con mayor tranquilidad.
- Por ultimo, estudiar bastante y querer aprender ya que así uno encuentra los errores y soluciona los problemas con mayor facilidad.

5. Referencias

[1] G. Delgado, (2023). Material del Curso IE-0623. UCR.

[2] T. Almy, (2011). Designing with Microcontrollers – The 68HCS12.

[3] Motorola Inc., Freescale Semiconductor, ATD_10B8C Block User Guide, V02.11, Original Release Date: 27 OCT 2000, Revised: 24 March 2005.

[4] Motorola Inc., Freescale Semiconductor, MC9S12DP256B Device User Guide, V02.15 Original Release Date: 29 Mar 2001, Revised: Jan 11, 2005.