Universidad de Costa Rica

Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
IE0623 Microprocesadores

I ciclo 2025

Separador 623

Estudiante: Bryan Cortés Espínola, C22422

Profesor: Geovanny Delgado Cascante

Índice

Índice de figuras

Índice de tablas

1. Resumen

El presente proyecto, titulado **Separador 623**, consistió en el diseño, desarrollo e implementación de un sistema embebido basado en microprocesador, cuyo objetivo principal es clasificar piezas (tuercas) según su velocidad de desplazamiento a lo largo de una banda de transporte. Para lograr esto, se utilizaron sensores de proximidad colocados estratégicamente, junto con una lógica de cálculo de velocidad implementada en lenguaje ensamblador sobre la plataforma *Dragon12+*.

El sistema fue desarrollado siguiendo un enfoque modular, estructurado en *tareas* implementadas como *máquinas de estado finito*. Cada tarea cumple una función específica dentro del sistema, tales como lectura de sensores, control de pantalla LCD, lectura del teclado matricial, conversión analógica-digital, control del brillo, procesamiento de pulsadores, despliegue de información en displays de 7 segmentos y control de la centrifugadora mediante relé. Estas tareas son ejecutadas secuencialmente por un **despachador central** sincronizado por una subrutina de atención a interrupciones que opera a una frecuencia de **50 kHz**, lo que garantiza una temporización precisa.

El sistema cuenta con tres modos de operación configurables mediante interruptores DIP: **Modo Stop**, que representa un estado seguro e inactivo; **Modo Configurar**, que permite al usuario ingresar un valor de velocidad umbral entre 10 y 80 cm/s; y **Modo Separar**, donde se activa el proceso de clasificación. Durante la operación, el sistema mide el tiempo que una tuerca tarda en desplazarse entre dos sensores, calcula la velocidad, la compara con el umbral definido por el usuario, y según el resultado, redirige la pieza a una de las dos líneas de empaque.

Además, se incorporaron mecanismos de validación y control de errores, incluyendo la detección de velocidades fuera de rango, condiciones de rebose y mensajes de advertencia visualizados en la pantalla LCD. Todo esto se complementa con una interfaz de usuario clara e intuitiva, que proporciona retroalimentación en tiempo real a través de indicadores visuales y displays.

La realización de este proyecto permitió integrar conocimientos fundamentales sobre arquitectura de microprocesadores, temporización, uso de periféricos y programación a bajo nivel. Asimismo, fomentó el desarrollo de habilidades prácticas en depuración, estructuración de código y documentación técnica, consolidando una experiencia formativa integral para el futuro ejercicio profesional en el área de sistemas digitales y control automático.

2. Diseño de la aplicación

Se va a proceder con el diseño de la aplicación Selector, el cual va a tener un diseño modular basado en tareas las cuales se construyen como máquinas de estados, las cuales van a tener control de tiempos y temporalización necesarios en el sistema por medio de La Máquina de tiempos, la cual corresponde a la subrutina de atención a interrupciones producida. mediante el módulo de Timer utilizado como **Output Compare**. Las tareas se van a ejecutar de manera secuencial en el despachador de tareas el cual se encarga de ejecutar un estado de una tarea por cada iteración.

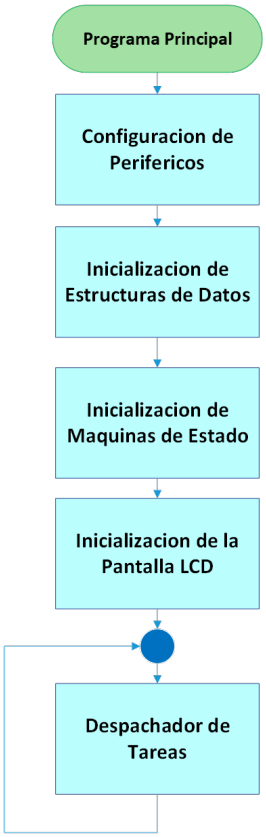

Figura 1: Diagrama de flujo del programa Principal

2.1. Memoria de Calculo

2.1.1. Output Compare

Se desea configurar el módulo Timer para que produzca interrupciones por salida de comparación a una frecuencia de 50 KHz, con el fin de tener una base de tiempo adecuada para el control de la pantalla LCD y las multiplexados, por lo que se debe tener un período de

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

El periodo de interrupción esta definido en la siguiente formula:

$$T_{oc} = \frac{PRS \times T_{c4}}{Bus_{CLK}}$$

Se toma el Valor de PRS = 16, ademas en la Tarjeta Dragon 12 el valor de Bus_{CLK} es de 24MHz, por lo tanto al despajar T_{c4} se obtiene que:

$$T_{c4} = \frac{Bus_{CLK} \times T_{oc}}{PRS} = \frac{24MHz \times 20\mu s}{16} = 30$$

Por lo cual la carga para tener interrupciones cada $20\mu s$ es de 30.

2.1.2. Convertidor Analogico Digital

Para el convertidor analógico digital se desea configurarlo a una frecuencia de muestreo $f_s = 600kHz$, la cual esta dada por:

$$f_s = \frac{Bus_{CLK}}{2(PRS+1)}$$

$$600kHz = \frac{24MHz}{2(PRS+1)}$$

Note que dicha expresión es cierta si PRS es igual a 19, Dando exacto la frecuencia de muestreo deseada.

2.1.3. Subrutina Calcula

En esta subrutina se debe calcular los siguientes Parámetros:

- **Velocidad**: Se calcula partir de la distancia entre los sensores y el tiempo que le tomó la tuerca pasar entre ellos

  $$Velocidad = \frac{DeltaS}{DeltaT}$$

  Para DeltaT se calcula por medio del TimerCal

  $$DeltaT = tTimerCal - TimerCal$$

  Se debe considerar que TimerCal esta en base de 100 ms

- **TimerIniPant**: Es el tiempo que le toma llegar la tuerca a la posición de inicio de mensaje

  $$TimerIniPant = \frac{DeltaE - DeltaM - DeltaS}{Velocidad}$$

  Se debe considerar que TimerIniPant dentro del programa es un Timer de base 100 ms, ademas de que se calcula y inicia cuando la tuerca llega al sensor 2, Por ello es necesario restarle la distancia DelTaS

- **TimerFinPant**: Es el tiempo que le toma llegar la tuerca a la posición de Fin de mensaje

  $$TimerFinPant = \frac{DeltaE - DeltaS}{Velocidad}$$

  Se debe considerar que TimerFinPant dentro del programa es un Timer de base 100 ms,ademas de que se calcula y inicia cuando la tuerca llega al sensor 2, Por ello es necesario restarle la distancia DelTaS

2.2. Configuración de Periféricos

Para esta aplicación se utilizan los siguientes periféricos:

- **Leds:** Se utilizan los LEDs conectados al puerto B en la Dragon 12 para indicar en qué modo se está operando, se envía en la Línea de empaque 1 o 2, y si hay rebase, se configuran como GPIOs y se declaran como salidas.
- **Relé** Se utiliza el relé conectado en el bit 2 del puerto E para el arranque de la centrifugadora, se configura como GPIO y se declara como salida.
- **Displays de 7 segmentos y Led Tricolor**: Para estos periféricos se coloca el puerto P como salida el nibble inferior para habilitar los displays y los tres bits menos significativos para el LED tricolor.
- **Pantalla LCD**: Se colocan los bits del 0 a 5 del puerto K como salidas para poder comunicarse con la Pantalla LCD.
- **Timer Output Compare**: Se inicializa para que que produzca una interrupción a una frecuencia de 50 KHz o 20μs y es la base del sistema para la maquina de tiempos.
- **Teclado**: Se declara el nibble superior del puerto A como salidas y en inferior como entradas para poder leer el teclado matricial.
- **ATD** Se inicializa el convertidor analógico digital 0 en el puerto 7 para medir los valores del potenciómetro. Se configura con una frecuencia de muestreo de 600 Khz, a 8 bits, con cuatro mediciones por ciclo y dos ciclos de reloj para el muestreo.

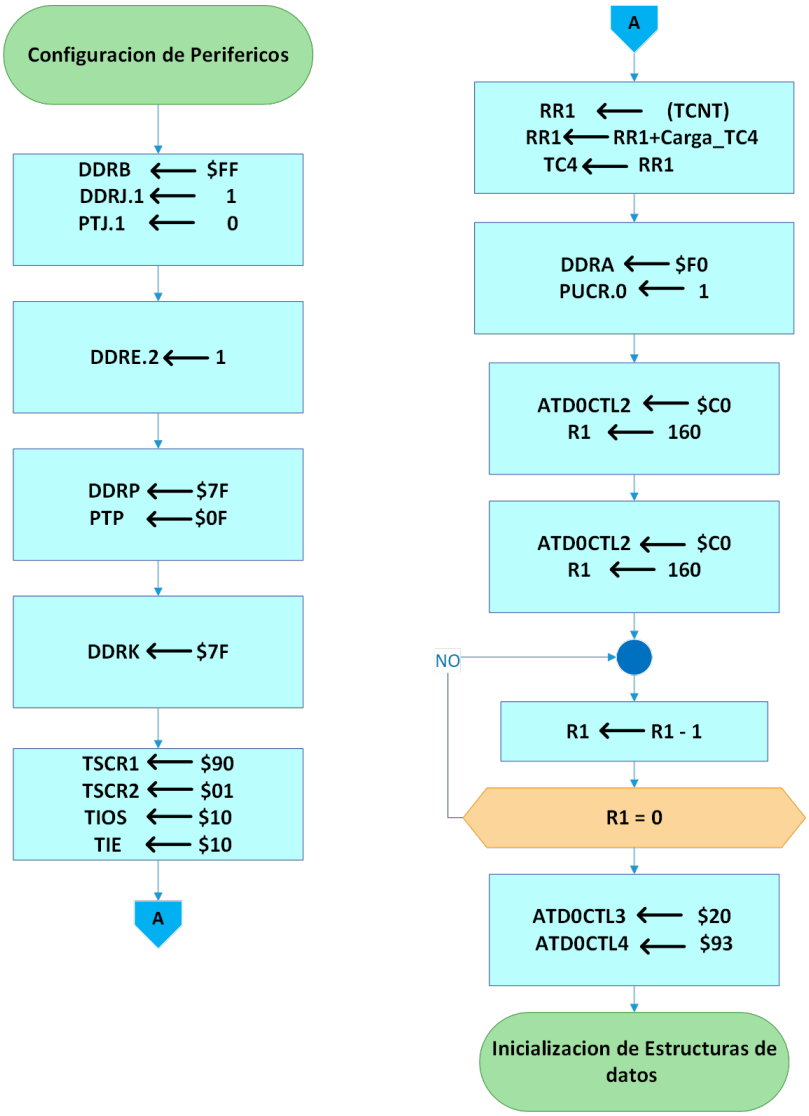

Figura 2: Diagrama de flujo de la Configuración de Periféricos

2.3. Inicializan de Estructuras de Datos

Esta sección del programa se encarga de asignar valores iniciales a todas las variables y estructuras de datos utilizadas en la aplicación. Su correcta configuración garantiza un arranque limpio y predecible del sistema al encenderse o reiniciarse el microcontrolador.

Inicializaciones Principales:

- **Temporizadores de base de tiempo:** Se cargan los valores constantes definidos para las bases de tiempo de 1 ms, 10 ms, 100 ms y 1 s, utilizados por la `Maquina_Tiempos`.
  - `Timer1mS, Timer10mS, Timer100mS, Timer1S`
- **Variables de entrada de teclado:** Se inicializan con `$FF` las variables `Tecla` y `Tecla_IN`, indicando que no se ha presionado ninguna tecla.
- **Contadores y banderas:**

  - `Cont_TCL` se limpia para reiniciar la cuenta de teclas ingresadas.
  - `Banderas_1` se borra para asegurar que no queden banderas activas desde un estado previo.
  - `MAX_TCL` se inicializa en 2, indicando la longitud máxima del arreglo de teclas.
- **Contadores de líneas de empaque:** Las variables `CantLE1` y `CantLE2` se limpian, reiniciando el conteo de tuercas enviadas a cada línea.
- **Valor de rebase:** Se inicializa `ValorRebase` con el valor 5, lo cual define el número máximo de tuercas permitidas en una línea antes de activar la condición de rebase.
- **Timer LED Testigo:** Se inicializa el temporizador `Timer_LED_Testigo` que se usa para controlar el parpadeo periódico del LED indicador de estado.
- **Variables de despliegue:** Se configura `Cont_Dig`, `Counter_Ticks` y `TimerDigito` para gestionar el barrido de los dígitos del display de 7 segmentos.
- **Brillo:** Se establece un valor inicial de 90 en la variable `Brillo`, que será luego escalado según la lectura del potenciómetro.
- **LEDs:** Se inicializa la variable `LEDS` en cero, indicando que todos los indicadores visuales están apagados al inicio.
- **Puntero de pila e interrupciones:** Se configura el Stack Pointer en la dirección `$3BFF` y se habilitan las interrupciones con `CLI`, lo que permite la atención a eventos temporales desde ese punto en adelante.

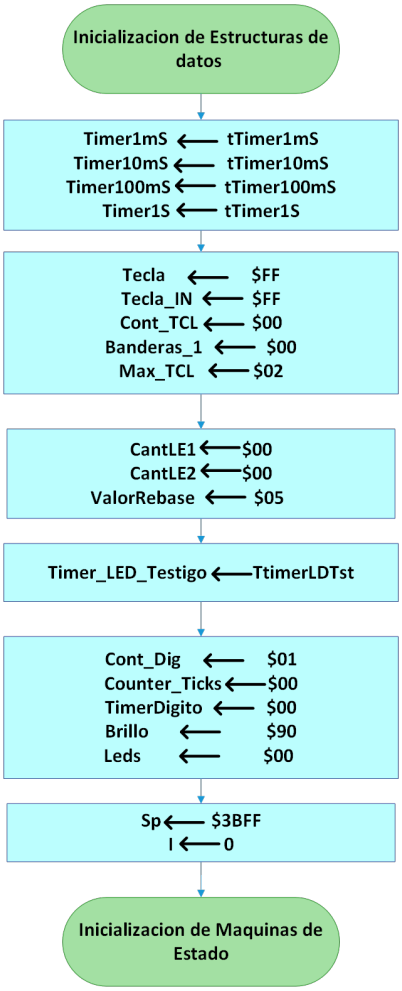

Figura 3: Diagrama de flujo de la inicializacion de estructura de Datos

2.4. Inicializan de Maquinas de Estado

Esta sección del programa asigna el estado inicial (Estado 1) a todas las máquinas de estado utilizadas en la aplicación. Cada tarea o subrutina que opera como una máquina de estados requiere una variable que almacene el estado actual, y que pueda ser consultada y modificada dinámicamente a lo largo de la ejecución del sistema.

Funcionamiento General: Cada variable de estado se inicializa apuntando a su correspondiente etiqueta de estado inicial, generalmente denominada `Est1`. Esto garantiza que, al iniciar el sistema, todas las máquinas de estado comiencen desde su configuración predeterminada y sigan un flujo de operación controlado.

Máquinas de Estado Inicializadas:

- `EstPres_LeerPB2`: Lectura del botón PB2. Desarrollado en la sección 3.9 .
- `EstPres_LeerPB1`: Lectura del botón PB1. Desarrollado en la sección 3.9.
- `Est_Pres_TCL`: Lectura de teclado matricial. Desarrollado en la sección 3.10.
- `Est_Pres_LDTst`: Parpadeo del LED testigo. Desarrollado en la sección 3.11.

- `EstPres_PantallaMUX`: Control del display multiplexado. Desarrollado en la sección 3.6.
- `Est_Pres_TBrillo`: Lectura del potenciómetro para el brillo. Desarrollado en la sección 3.5
- `Est_Pres_LeerDS`: Lectura del selector de modo (DIP Switch). Desarrollado en la sección 3.4
- `Est_Pres_TConfig`: Estado inicial del modo Configurar. Desarrollado en la sección 3.2
- `Est_Pres_TSeparar`: Estado inicial del modo Separar. Desarrollado en la sección 3.3
- `EstPres_SendLCD`: Máquina que envía caracteres al LCD. Desarrollado en la sección 3.7
- `EstPres_TareaLCD`: Máquina que gestiona los mensajes en pantalla LCD. Desarrollado en la sección 3.8

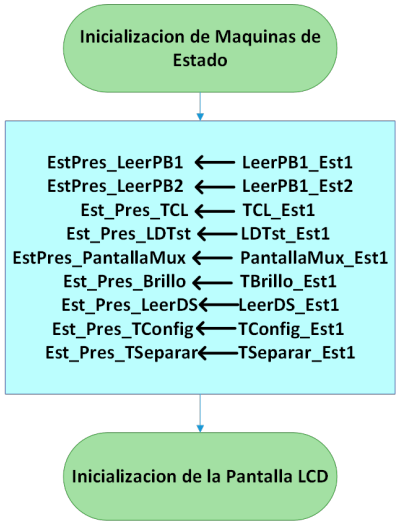

Figura 4: Diagrama de flujo de la inicializacion de las maquinas de estado

2.5. Inicializacion de la Pantalla LCD

En esta sección se inicializa y se borra la pantalla LCD, dejándola lista para su uso en la aplicación. Para ello se mandan los comandos de **FuncionSet1**, indicando el modo de 4 bits, 2 líneas y fuente de 5 x 8 puntos, **Entry Mode Set** y **Display ON/OFF Control** indicando Display encendido, cursor apagado y sin parpadeo. La correcta inicialización de la pantalla LCD asegura que los mensajes del sistema puedan ser desplegados desde el primer ciclo de ejecución.

2.5.1. Estructuras de Datos

Estructuras de Datos	
Variable	Descripción
MSG_L1	Tipo Word, Puntero del mensaje a mostrar en la primer Linea
MSG_L2	Tipo Word, Puntero del mensaje a mostrar en la segunda Linea
Banderas_2	Tipo Byte, Contiene las banderas asociadas a la Pantalla LCD
Punt_LCD	Tipo Word, Puntero para barrer los mensajes del LCD
CharLCD	Tipo Byte, Variable para guardar el dato a enviar al LCD
Timer2ms	Tipo Byte, Variable para el timer de 2ms
Valores y banderas	
Nombre	Descripción
MSG_STOP_L1	Dirección del Mensaje a mostrar en la primera Linea
MSG_STOP_L2	Dirección del Mensaje a mostrar en la segunda Linea
EOB	Valor de $FF, indica el fin del bloque de mensaje
IniDSP	Dirección de la tabla de los comandos para inicializar el LCD
Clear_LCD	Valor de $01, Comando para el borrado de la LCD
FinSendLCD	Bandera, el bit 2 de la variable Banderas_2, indica si se envio el dato al LCD
tTimer2ms	Valor de 2 para cargar el timer de 2ms

Tabla 1: Estructuras de Datos utilizados en la Inicializacion de la Pantalla LCD

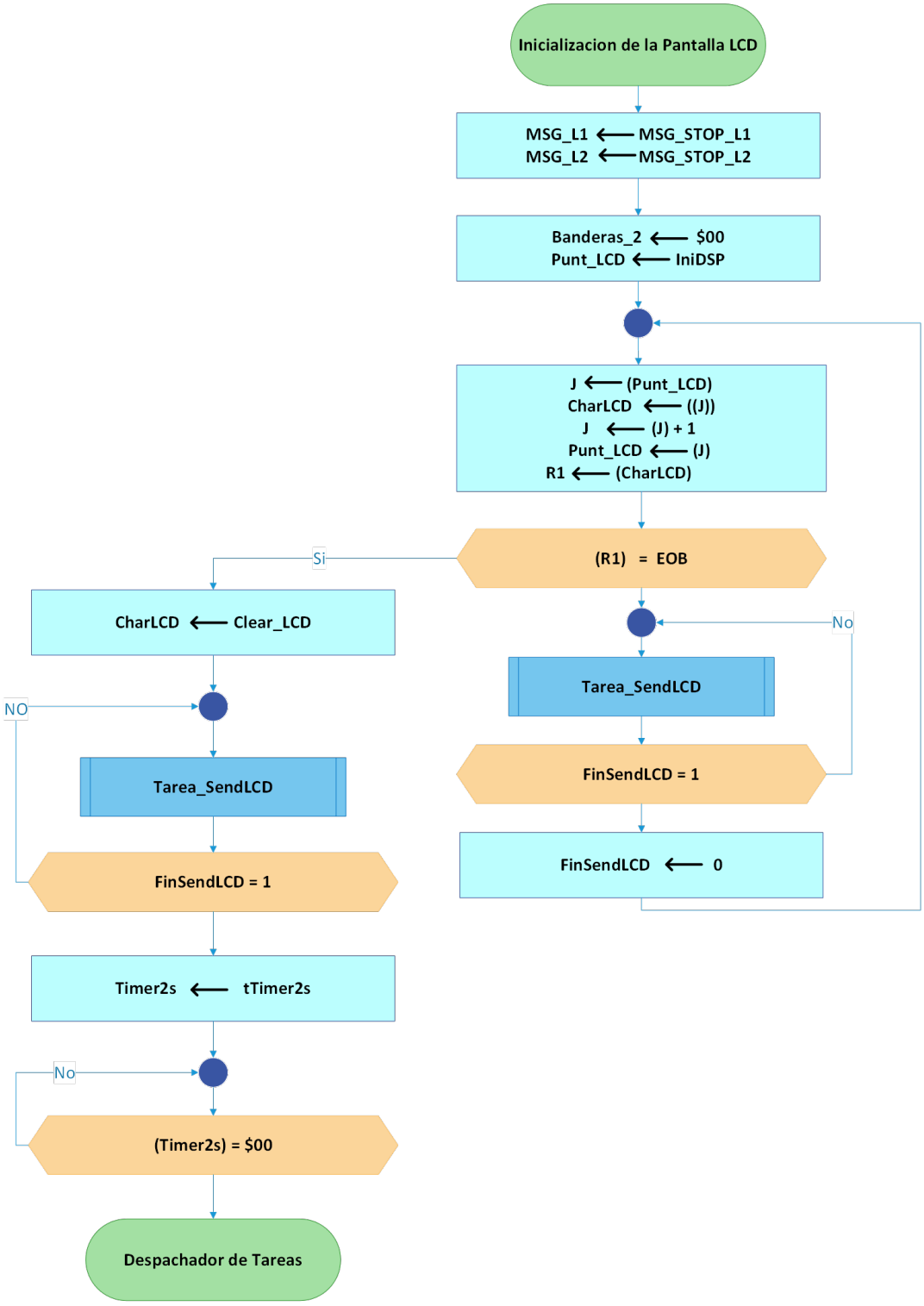

Figura 5: Diagrama de flujo de la inicializacion de la Pantalla LCD

2.6. Despachador de tareas

El Despachador de tareas se encarga de ejecutar las tareas del sistema, en el va ejecutando las tareas por estados ciclicamente.

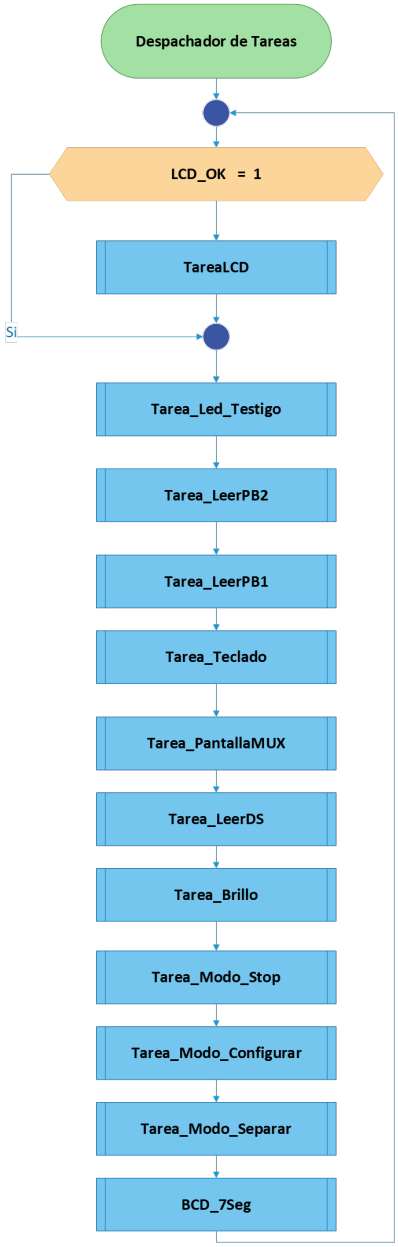

Figura 6: Diagrama de flujo del Despachador de tareas

3. Tareas

3.1. Modo Stop

El Modo STOP representa el estado inactivo y seguro del sistema Separador 623. En este modo, el sistema permanece en espera, con la centrífuga desactivada, sin procesar tuercas ni permitir el ingreso de nuevos parámetros de configuración.

Este modo cumple la función de estado inicial tras el encendido del sistema, así como de punto de reposo cuando el operador desea detener temporalmente la operación del equipo. Durante este estado, se muestra un mensaje de bienvenida en la pantalla LCD, se apagan los dígitos del display de 7 segmentos y únicamente permanece encendido el LED indicador correspondiente al modo STOP.

Parámetros de entrada:

- `Valor_DS` : Accedido por memoria, Verifica si es valor es igual a $00, si lo es se entra en modo Stop, Caso contrario no se tiene ningún efecto.

3.1.1. Estructuras de Datos

Estructuras de Datos	
Variable	Descripción
ValorDS	Tipo Byte, Variable a cual se muestra los valores ingresados en los dip switches.
Leds	Tipo Byte, Variable a cual se muestra en los Leds.
MSG_L1	Tipo Word, Puntero del mensaje a mostrar en la primer Linea.
MSG_L2	Tipo Word, Puntero del mensaje a mostrar en la segunda Linea.
BCD1	Tipo Byte, Variable en BCD que se muestra en los Displays.
BCD2	Tipo Byte, Variable en BCD que se muestra en los Displays.
Valores y banderas	
Nombre	Descripción
MSG_STOP_L1	Dirección del Mensaje a mostrar en la primera Linea.
MSG_STOP_L2	Dirección del Mensaje a mostrar en la segunda Linea.
LDStop	Valor para el Led del bit 7 del Puerto B que indica el Modo Stop.
LCD_Ok	Bandera, Bit 1 de Banderas_2, indica si hay un nuevo mensaje mostrar en la LCD.
OFF	Valor de $BB, Indica que los displays deben estar apagado.

Tabla 2: Estructuras de Datos utilizados en el modo Stop

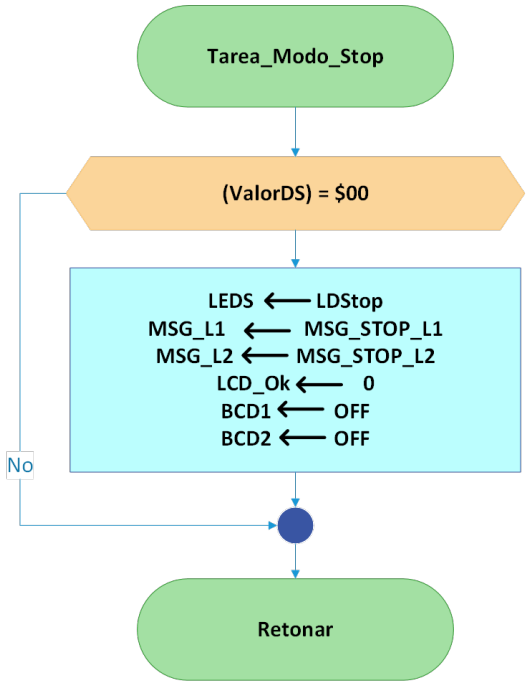

Figura 7: Diagrama de flujo de la Tarea modo Stop

3.2. Modo Configurar

El **Modo CONFIGURAR** permite al operador programar el valor de la **velocidad umbral** que se utilizará para clasificar las tuercas durante el proceso de separación. Este valor debe estar dentro del intervalo de velocidades válidas definido por el sistema, es decir, entre $V_{\text{min}} = 10$ cm/s y $V_{\text{max}} = 80$ cm/s.

Al ingresar a este modo, se activa el LED indicador correspondiente y se despliega un mensaje de configuración en la pantalla LCD. El sistema suspende toda acción relacionada con el modo SEPARAR y habilita el uso del teclado matricial para ingresar el nuevo valor de V_u. El valor introducido se valida y, si está dentro del rango permitido, se convierte a formato binario y se almacena en la variable `VelUmbral`, actualizando simultáneamente la visualización en el display de 7 segmentos. En caso de que el valor sea inválido, se descarta la entrada y se reinicia el proceso de captura.

El sistema permanece en el modo CONFIGURAR hasta que el operador cambie manualmente el modo de operación mediante los interruptores. Esto permite realizar múltiples configuraciones consecutivas de manera flexible.

Parámetros de entrada:

- `Valor_DS` : Accedido por memoria, Verifica si es valor es igual a $40, si lo es se entra en modo Configurar, Caso contrario no se tiene ningún efecto.

Parámetros de Salida:

- `VelUmbral` : Accedido por memoria, Es la Velocidad Umbral ingresada por el usuario.

3.2.1. Estructuras de Datos

Estructuras de Datos	
Variable	Descripción
ValorDS	Tipo Byte, Variable a cual se muestra los valores ingresados en los dip switches.
Leds	Tipo Byte, Variable a cual se muestra en los Leds.
MSG_L1	Tipo Word, Puntero del mensaje a mostrar en la primer Linea.
MSG_L2	Tipo Word, Puntero del mensaje a mostrar en la segunda Linea.
Est_Pres_TConfig	Tipo Word, Puntero del estado Presente de la Tarea.
BCD1	Tipo Byte, Variable en BCD que se muestra en los Displays.
BCD2	Tipo Byte, Variable en BCD que se muestra en los Displays.
VelUmbral	Tipo Byte, Contiene el valor de la velocidad umbral.
ValorVelUmbral	Tipo Byte, Variable auxiliar para la velocidad umbral.
Valores y banderas	
Nombre	Descripción
MSG_CONFIG_L1	Dirección del Mensaje a mostrar en la primera Linea.
MSG_CONFIG_L2	Dirección del Mensaje a mostrar en la segunda Linea.
LDStop	Valor para el Led del bit 7 del Puerto B que indica el Modo Stop.
LCD_Ok	Bandera, indica si hay un nuevo mensaje mostrar en la LCD.
OFF	Valor de $BB, Indica que los displays deben estar apagado.
ArrayOK	Bandera, Indica si se ingreso una secuencia de teclas.
num_array	Dirección del arreglo de teclas ingresadas
Vmax	Valor de 80, Es la máxima velocidad de umbral, en cm/s
Vmin	Valor de 10, Es la mínima velocidad de umbral, en cm/s

Tabla 3: Estructuras de Datos utilizados en el modo Configurar

3.2.2. Diagramas de flujo

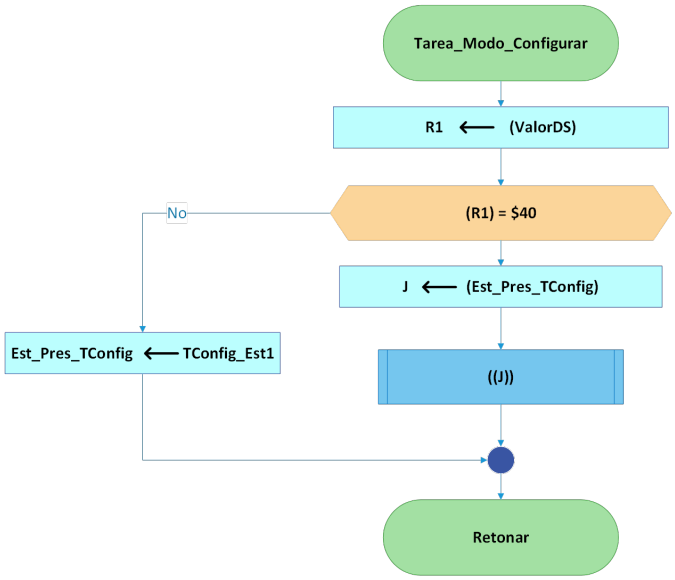

Figura 8: Diagrama de flujo de la Tarea modo Configurar

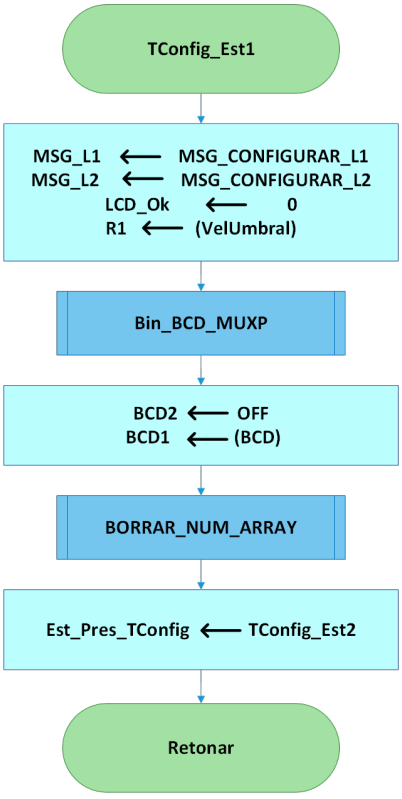

Figura 9: Diagrama de flujo del estado 1 la Tarea modo Configurar

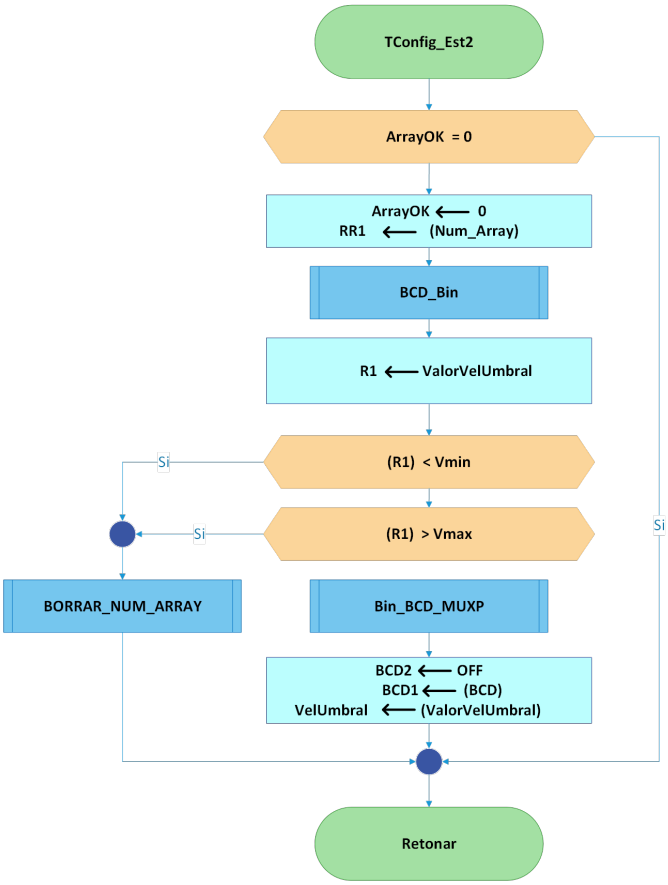

Figura 10: Diagrama de flujo del estado 2 la Tarea modo Configurar

3.3. Modo Separar

El **Modo SEPARAR** es el estado operativo principal del sistema *Separador 623*, en el cual se lleva a cabo el proceso de clasificación de tuercas en función de su velocidad lineal a lo largo del canal de empaque. Este modo se activa mediante los interruptores de modo y permanece activo hasta que el operador decida cambiar de estado o se alcance una condición de rebose.

Una vez en este modo, el sistema permanece en espera hasta que el operador presione el botón de inicio, lo cual activa la centrífuga. A partir de ese momento, el sistema monitoriza los sensores de proximidad (P1 y P2) para medir el tiempo que una tuerca tarda en recorrer una distancia conocida, y con ello calcular su velocidad. En función de si esta velocidad supera o no el valor de umbral V_u, la tuerca se direcciona a la Línea de Empaque 1 o 2, activando el LED correspondiente.

Durante este proceso, el sistema actualiza la información en la pantalla LCD y en el display de 7 segmentos, mostrando la velocidad calculada y el conteo de tuercas por línea. Si se detecta una velocidad fuera de rango, se emite un mensaje de alerta y se solicita al operador retirar la pieza defectuosa. En caso de alcanzar la cantidad máxima de tuercas permitidas en una línea (definida por el parámetro `ValorRebase`), se detiene automáticamente la centrífuga y se activa el LED de rebose. El operador debe presionar nuevamente el botón de inicio para reanudar el ciclo.

Parámetros de entrada:

- `Valor_DS` : Accedido por memoria, Verifica si es valor es igual a $C0, si lo es se entra en modo Separar, Caso contrario no se tiene ningún efecto.

- `VelUmbral` : Accedido por memoria, Es la Velocidad Umbral ingresada por el usuario. Se usa para separar las tuercas.
- `ValorRebase`: Accedido Por memoria, Cantidad máxima permitida de tuercas por linea de empaque

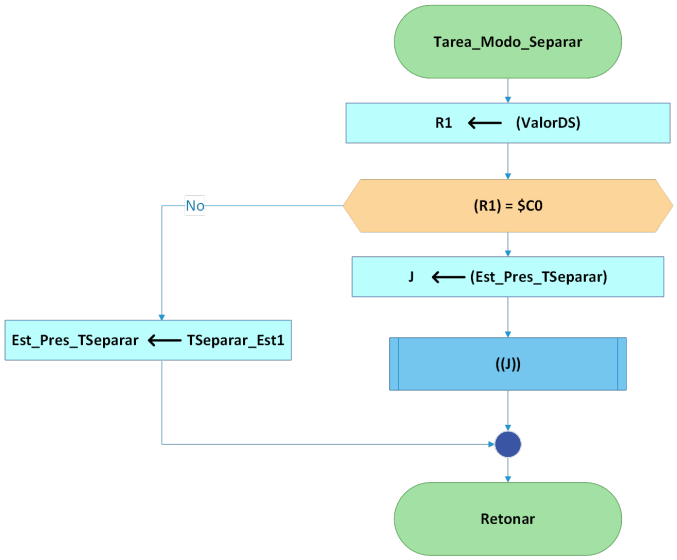

Figura 11: Diagrama de flujo de la Tarea modo Separar

3.3.1. Estado 1 – Inicialización del modo SEPARAR

- Se activa el indicador luminoso correspondiente al modo SEPARAR.
- Se despliega un mensaje de bienvenida en la pantalla LCD, indicando que el sistema está listo para iniciar el proceso.
- Se apagan los displays de 7 segmentos.
- Se transiciona al estado 2, en espera del inicio por parte del operador.

Estructuras de Datos	
Variable	Descripción
Leds	Tipo Byte, Variable a cual se muestra en los Leds.
MSG_L1	Tipo Word, Puntero del mensaje a mostrar en la primer Linea.
MSG_L2	Tipo Word, Puntero del mensaje a mostrar en la segunda Linea.
Est_Pres_TSeparar	Tipo Word, Puntero del estado Presente de la Tarea.
BCD1	Tipo Byte, Variable en BCD que se muestra en los Displays.
BCD2	Tipo Byte, Variable en BCD que se muestra en los Displays.
Valores y banderas	
Nombre	Descripción
MSG_SEPARAR_L1	Dirección del Mensaje a mostrar en la primera Linea.
MSG_SEPARAR_L2	Dirección del Mensaje a mostrar en la segunda Linea.
LCD_Ok	Bandera,indica si hay un nuevo mensaje mostrar en la LCD.
OFF	Valor de $BB, Indica que los displays deben estar apagado.
TSeparar_Est2	Direccion de la subrutina del estado 2

Tabla 4: Estructuras de Datos utilizados en el Estado 1 del Modo Separar

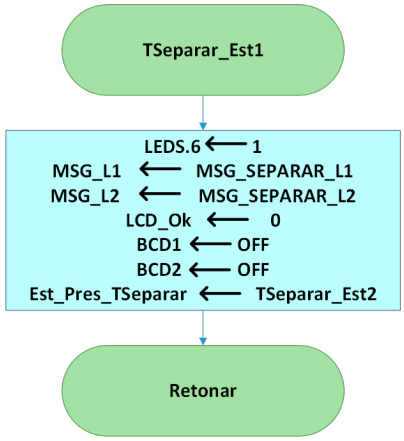

Figura 12: Diagrama de flujo del estado 1 de la tareas modo Separar

3.3.2. Estado 2 – Esperando botón de inicio

- El sistema espera la activación del botón de inicio (long press).
- Una vez detectado, se despliega el mensaje correspondiente y se activa la centrífuga.
- Se reinician las variables y se transiciona al estado 3.

Estructuras de Datos	
Variable	Descripción
Leds	Tipo Byte, Variable a cual se muestra en los Leds.
MSG_L1	Tipo Word, Puntero del mensaje a mostrar en la primer Linea.
MSG_L2	Tipo Word, Puntero del mensaje a mostrar en la segunda Linea.
Est_Pres_TSeparar	Tipo Word, Puntero del estado Presente de la Tarea.
Valores y banderas	
Nombre	Descripción
MSG_SEPARAR_L1	Dirección del Mensaje a mostrar en la primera Linea.
MSG_SEPARAR_L2	Dirección del Mensaje a mostrar en la segunda Linea.
LCD_Ok	Bandera,indica si hay un nuevo mensaje mostrar en la LCD.
TSeparar_Est3	Direccion de la subrutina del estado 3.
LongP2	Bandera, Indica si se produjo un LongPress.
ShortP2	Bandera, Indica si se produjo un ShortPress.
PortRELE	Valor para el puerto E.

Tabla 5: Estructuras de Datos utilizados en el Estado 2 del Modo Separar

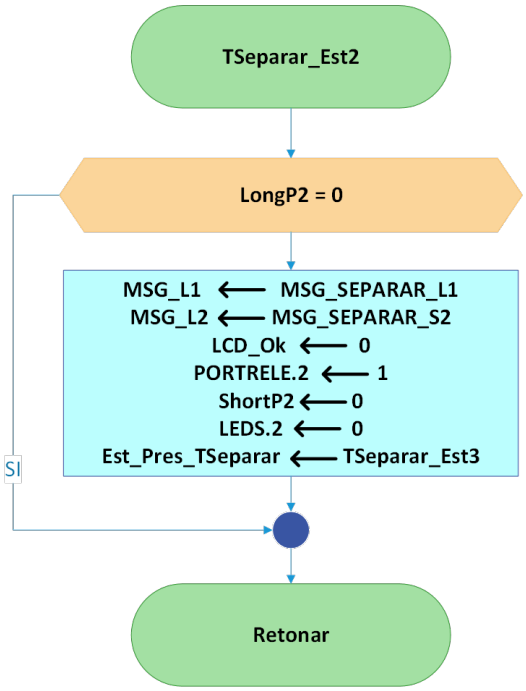

Figura 13: Diagrama de flujo del estado 2 de la tareas modo Separar

3.3.3. Estado 3 – Esperando sensor P1

- Se espera la detección de una tuerca por el sensor P1.
- Al detectarse, se inicia un temporizador de medición y se transiciona al estado 4.

Estructuras de Datos	
Variable	Descripción
MSG_L1	Tipo Word, Puntero del mensaje a mostrar en la primer Linea.
MSG_L2	Tipo Word, Puntero del mensaje a mostrar en la segunda Linea.
Est_Pres_TSeparar	Tipo Word, Puntero del estado Presente de la Tarea.
TimerCal	Tipo Byte, Timer en base de 100 ms.
Valores y banderas	
Nombre	Descripción
MSG_SEPARAR_L1	Dirección del Mensaje a mostrar en la primera Linea.
MSG_SEPARAR_S2	Dirección del Mensaje a mostrar en la segunda Linea.
LCD_Ok	Bandera,indica si hay un nuevo mensaje mostrar en la LCD.
TSeparar_Est4	Dirección de la subrutina del estado 3.
ShortP1	Bandera, Indica si se produjo un ShortPress.
tTimerCal	Valor de carga al TimerCal Con un valor de 100.

Tabla 6: Estructuras de Datos utilizados en el Estado 3 del Modo Separar

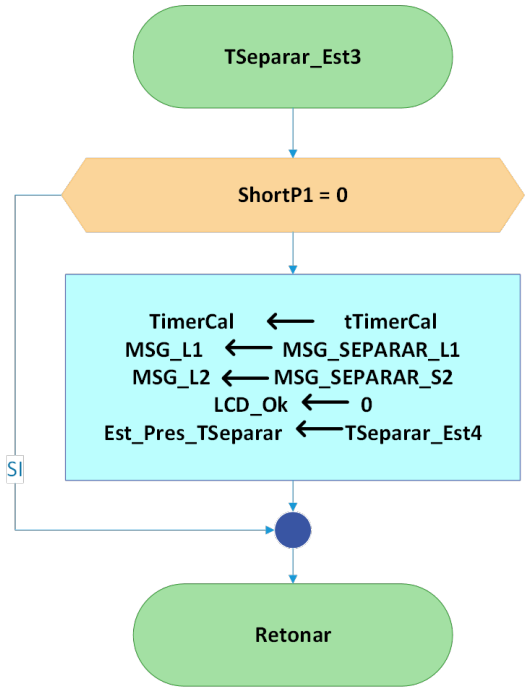

Figura 14: Diagrama de flujo del estado 3 de la tareas modo Separar

3.3.4. Estado 4 – Esperando sensor P2 y cálculo

- Se espera la activación del sensor P2.
- Una vez detectada la tuerca en P2, se calcula la velocidad a partir del tiempo transcurrido.
- Se valida que la velocidad esté dentro del rango permitido ($V_{min} \leq v \leq V_{max}$):

  - Si es válida, se calculan los temporizadores para la presentación de resultados.
  - Si es inválida, se muestran los valores de error en pantalla y se transiciona al estado 6.
- Si la velocidad es válida, se continúa al estado 5.

Estructuras de Datos	
Variable	Descripción
MSG_L1	Tipo Word, Puntero del mensaje a mostrar en la primer Linea.
MSG_L2	Tipo Word, Puntero del mensaje a mostrar en la segunda Linea.
Est_Pres_TSeparar	Tipo Word, Puntero del estado Presente de la Tarea.
Velocidad	Tipo Byte, Contiene el valor de velocidad medido.
TimerIniPant	Tipo Byte, contador en base de 100 ms
TimerFinPant	Tipo Byte, contador en base de 100 ms
BCD1	Tipo Byte, Variable en BCD que se muestra en los Displays.
BCD2	Tipo Byte, Variable en BCD que se muestra en los Displays.
BCD	Tipo Byte, Variable del resultado de Bin_BCD_MUXP.
TimerError	Tipo Byte, Timer en base de 1 s.
Valores y banderas	
Nombre	Descripción
MSG_SEPARAR_XX	Dirección del Mensaje a mostrar en la Pantalla LCD.
LCD_Ok	Bandera,indica si hay un nuevo mensaje mostrar en la LCD.
TSeparar_EstX	Dirección de la subrutina del estado X.
ShortP2	Bandera, Indica si se produjo un ShortPress.
OFF	Valor de $BB, Indica que los displays deben estar apagado.
Guion	Valor de $AA, Muestra un guion en los displays.
tTimerError	Valor para la carga del TimerError, con un valor de 2.

Tabla 7: Estructuras de Datos utilizados en el Estado 4 del Modo Separar

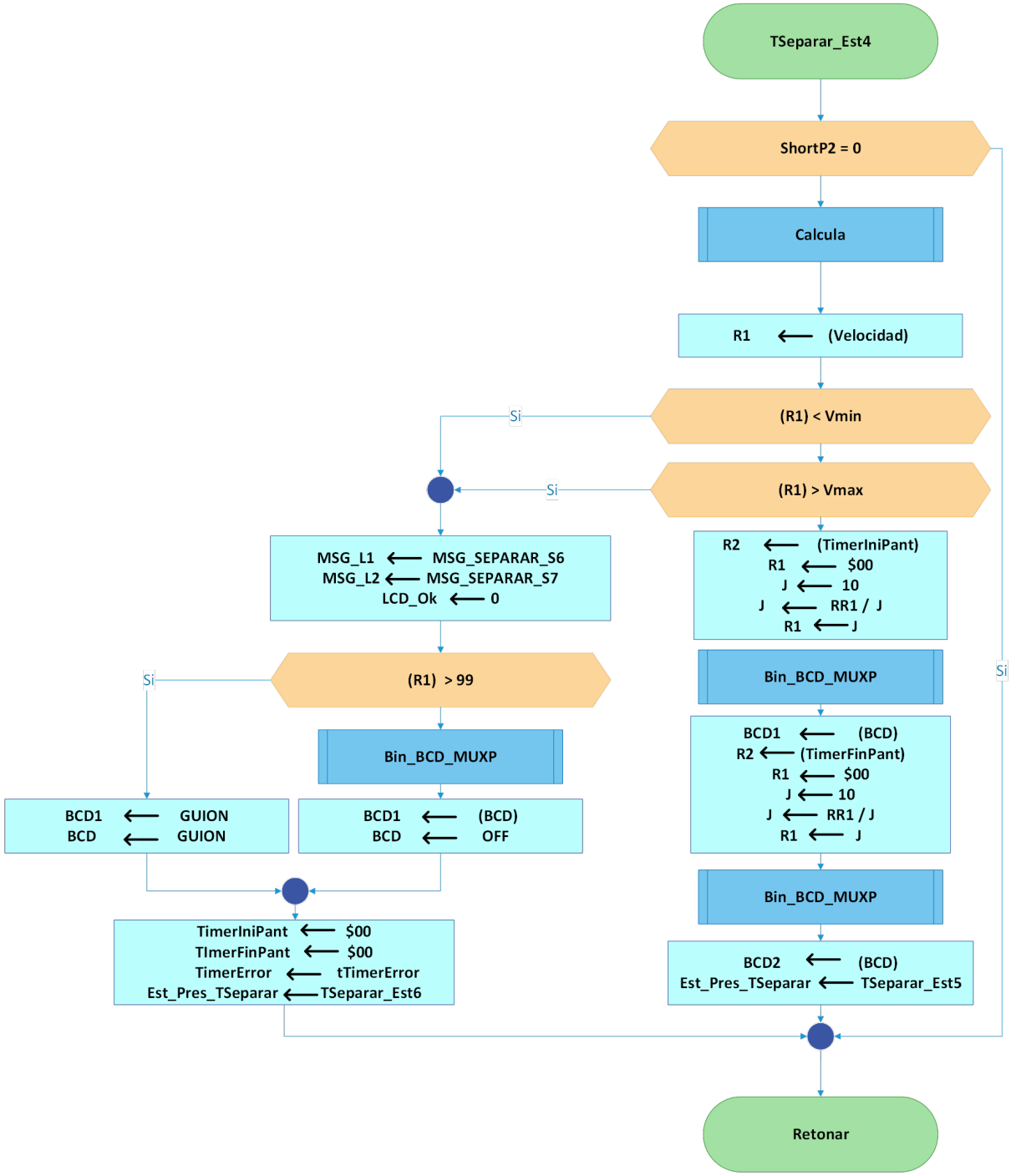

Figura 15: Diagrama de flujo del estado 4 de la tareas modo Separar

3.3.5. Estado 5 – Separar y Verificar rebase

- Una vez finalizado el temporizador `TimerIniPant`, se determina la línea de empaque:
  - Si la velocidad es mayor que la velocidad umbral ($v > Vu$), la tuerca se dirige a la Línea de Empaque 1.
  - En caso contrario, se dirige a la Línea de Empaque 2.

- Se incrementa el contador de tuercas en la línea correspondiente.
- Se verifica si el contador ha alcanzado el `ValorRebase`:
  - Si no se ha alcanzado, se muestran los datos en pantalla y se transiciona al estado 7.
  - Si se ha alcanzado, se detiene la centrífuga, se reinician los contadores y se activa el indicador de rebase, volviendo al estado 1.

Estructuras de Datos	
Variable	Descripción
MSG_L1	Tipo Word, Puntero del mensaje a mostrar en la primer Linea.
MSG_L2	Tipo Word, Puntero del mensaje a mostrar en la segunda Linea.
Est_Pres_TSeparar	Tipo Word, Puntero del estado Presente de la Tarea.
Velocidad	Tipo Byte, Contiene el valor de velocidad medido.
TimerIniPant	Tipo Byte, contador en base de 100 ms
BCD1	Tipo Byte, Variable en BCD que se muestra en los Displays.
BCD2	Tipo Byte, Variable en BCD que se muestra en los Displays.
BCD	Tipo Byte, Variable del resultado de Bin_BCD_MUXP.
CantLE1	Tipo Byte, Cantidad de tuerca en la linea de empaque 1.
CantLE2	Tipo Byte, Cantidad de tuerca en la linea de empaque 2.
Leds	Tipo Byte, Datos a mostrar en los LEDS
Valores y banderas	
Nombre	Descripción
MSG_SEPARAR_XX	Dirección del Mensaje a mostrar en la Pantalla LCD.
LCD_Ok	Bandera,indica si hay un nuevo mensaje mostrar en la LCD.
TSeparar_EstX	Dirección de la subrutina del estado X.
LongP2	Bandera, Indica si se produjo un LongPress.
PortRELe	Puerto E ubicado el Relé.

Tabla 8: Estructuras de Datos utilizados en el Estado 5 del Modo Separar

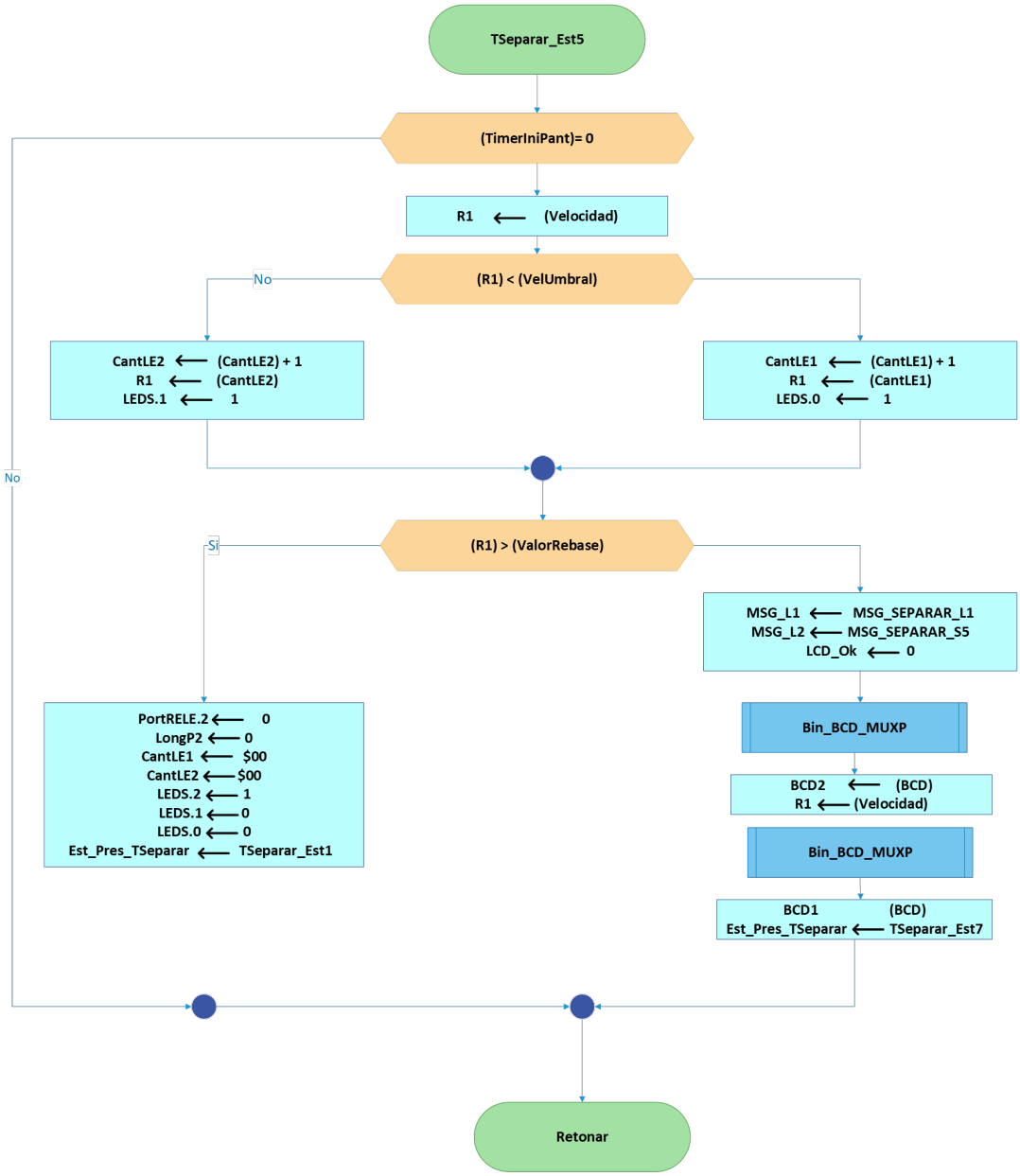

Figura 16: Diagrama de flujo del estado 5 de la tareas modo Separar

3.3.6. Estado 6 – Velocidad fuera de rango

- Se visualiza un mensaje de advertencia en la pantalla LCD.
- En los displays de 7 segmentos se muestra la velocidad medida o guiones si esta supera el límite.
- La información se mantiene visible durante un periodo definido por `TimerError`.
- Finalizado el tiempo, se regresa al estado 2.

Estructuras de Datos	
Variable	Descripción
Est_Pres_TSeparar	Tipo Word, Puntero del estado Presente de la Tarea.
TimerError	Tipo Byte, contador en base de 100 ms
Leds	Tipo Byte, Datos a mostrar en los LEDS
Valores y banderas	
Nombre	Descripción
TSeparar_EstX	Dirección de la subrutina del estado X.

Tabla 9: Estructuras de Datos utilizados en el Estado 6 del Modo Separar

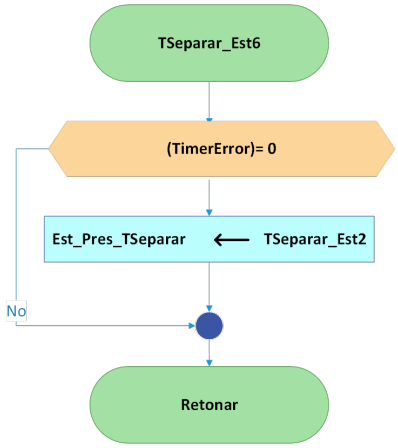

Figura 17: Diagrama de flujo del estado 6 de la tareas modo Separar

3.3.7. Estado 7 – Finalización del ciclo

- El sistema espera a que la tuerca haya pasado la escobilla.
- Una vez cumplido el tiempo definido por `TimerFinPant`, se apagan los indicadores de línea (LE1 o LE2).
- Se regresa al estado 2, esperando la siguiente tuerca.

Estructuras de Datos	
Variable	Descripción
Est_Pres_TSeparar	Tipo Word, Puntero del estado Presente de la Tarea.
TimerFinPant	Tipo Byte, contador en base de 100 ms
Valores y banderas	
Nombre	Descripción
TSeparar_EstX	Dirección de la subrutina del estado X.

Tabla 10: Estructuras de Datos utilizados en el Estado 6 del Modo Separar

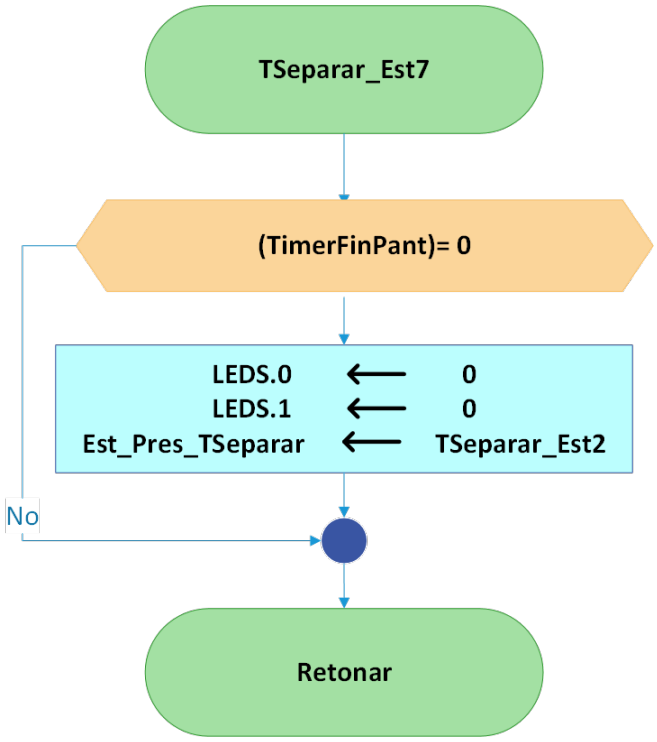

Figura 18: Diagrama de flujo del estado 7 de la tareas modo Separar

3.4. Tarea Leer DS

La **Tarea** `Leer_DS` es la encargada de leer el estado actual de los interruptores DIP (Dip Switches) que definen el **modo de operación** del sistema *Separador 623*. Los valores leídos se almacenan en una variable global llamada `Valor_DS`, la cual es utilizada por el despachador de tareas para determinar si el sistema debe ejecutar el Modo STOP, CONFIGURAR o SEPARAR.

Esta tarea es fundamental para garantizar que el sistema responda correctamente a los cambios de modo realizados por el operador.

Los interruptores de modo están conectados a las líneas **PH7 y PH6** del puerto H de la tarjeta Dragon12+. Cada combinación representa un modo:

- `OFF - OFF`: Modo STOP
- `OFF - ON`: Modo CONFIGURAR
- `ON - ON`: Modo SEPARAR
- **Estado 1:** Se realiza una primera lectura del estado de los dipswitches (PH7 y PH6) y se guarda en una variable temporal (`Temp_DS`). Simultáneamente, se inicia un temporizador de rebote (`Timer_RebDS`) con un valor predefinido para permitir que los rebotes se disipen. Luego, se transiciona al Estado 2.
- **Estado 2:** Se espera que el temporizador `Timer_RebDS` finalice. Una vez transcurrido el tiempo de espera, se vuelve a leer el valor actual de los dipswitches y se compara con `Temp_DS`. Si ambos valores coinciden, se considera una lectura estable y se actualiza la variable `Valor_DS`. En caso contrario, se descarta la lectura. Finalmente, se retorna al Estado 1.

Parámetros de Salida:

- `Valor_DS` : Accedido por memoria, Valor leido de los Dip Switches.

3.4.1. Estructuras de datos

Estructuras de Datos	
Variable	Descripción
Est_Pres_TLeerDS	Tipo Word, Puntero del estado Presente de la Tarea.
Temp_DS	Tipo Byte, Contiene el valor temporal leido de los DS.
Timer_RebDS	Tipo Byte, Timer en base de 1 ms para supresion de rebotes.
Valor_DS	Tipo Byte, Valor de los DS Leido.
Valores y banderas	
Nombre	Descripción
PortPB	Alias para el Puerto H Conectado a los DS.
LeerDS_EstX	Dirección de la subrutina del estado X.
tTimerRebDs	Valor para la carga Timer_RebDS, con un valor de 20.

Tabla 11: Estructuras de Datos utilizados en la tarea LeerDS

3.4.2. Diagramas de Flujo

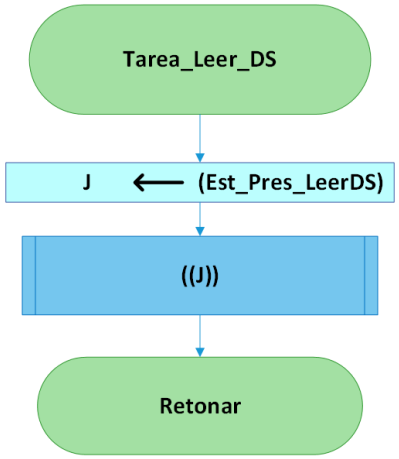

Figura 19: Diagrama de flujo de la tarea LeerDS

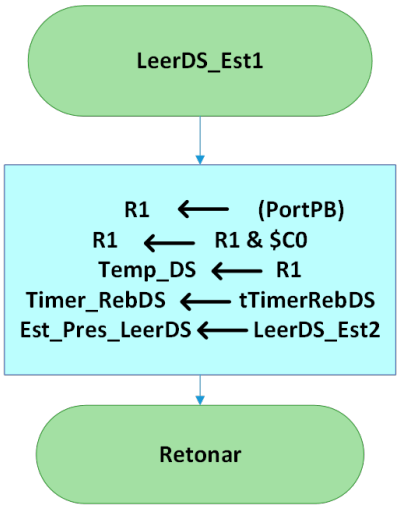

Figura 20: Diagrama de flujo del Estado 1 de la tarea LeerDS

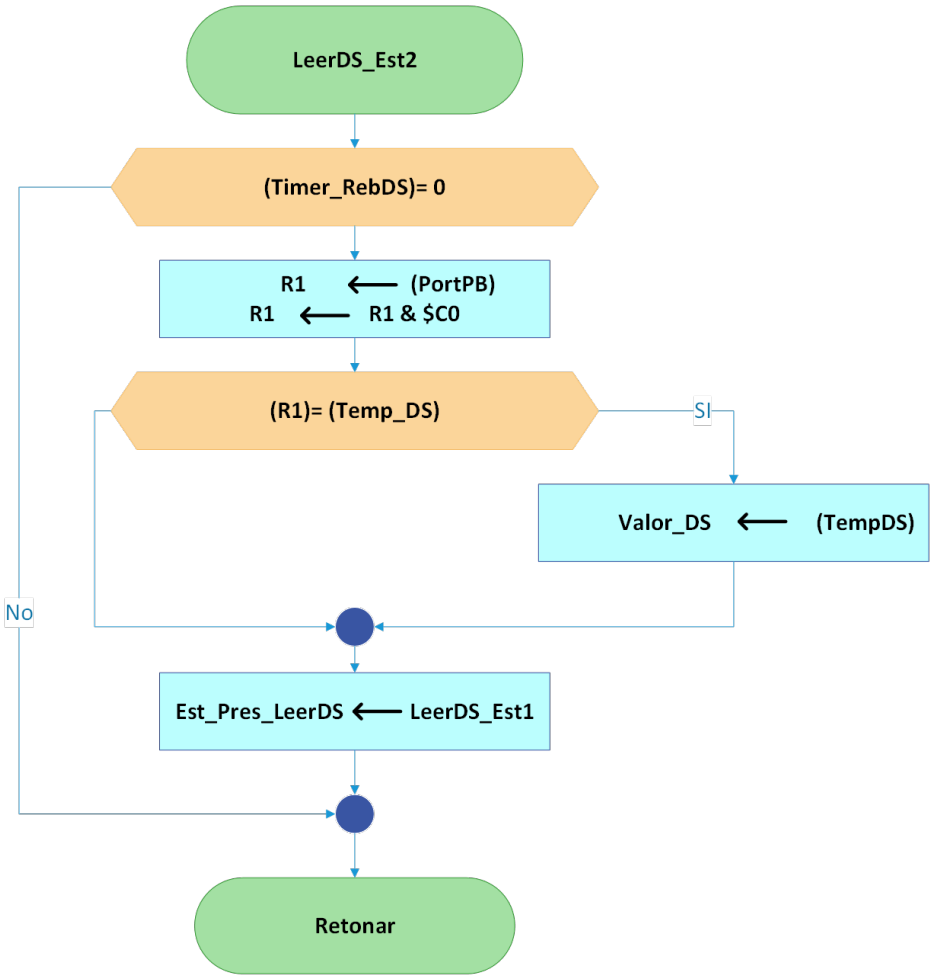

Figura 21: Diagrama de flujo del Estado 2 de la tarea LeerDS

3.5. Tarea Brillo

La **Tarea Brillo** es responsable de ajustar el nivel de brillo tanto del *display de 7 segmentos* como de los *LEDs indicadores*, en función del valor leído desde el potenciómetro conectado a la entrada del convertidor analógico-digital de la tarjeta Dragon12+.

Esta funcionalidad se implementa mediante una **máquina de estados de tres pasos**, diseñada para realizar el muestreo periódico del potenciómetro, ejecutar la conversión A/D y calcular el valor proporcional de brillo. Dicho valor es aplicado al hardware de salida visual por temporización, con el fin de mejorar la visibilidad según las condiciones de operación del entorno.

- **Estado 1 – Espera para iniciar conversión:** Se carga el temporizador `TimerBrillo` con un valor predefinido (`tTimerBrillo`), que determina el intervalo de actualización del brillo (por ejemplo, cada 300 ms). Una vez iniciado el temporizador, se transiciona al siguiente estado.
- **Estado 2 – Inicio de conversión A/D:** Cuando el temporizador `TimerBrillo` finaliza, se inicia una secuencia de conversión analógica a digital utilizando el canal correspondiente al potenciómetro (PAD7). Luego se avanza al estado 3.
- **Estado 3 – Cálculo del brillo:** Al finalizar la conversión, se obtiene el valor promedio de las lecturas del ADC (por ejemplo, tras realizar 4 muestreos). A partir de este valor, se calcula un nivel de brillo proporcional en el rango de 0 a 100 utilizando una progresión lineal. Finalmente, se retorna al estado 1 para esperar el siguiente ciclo.

Parámetros de Salida

- **Brillo**: Devuelto por memoria. Valor de 0 a 100 del nivel de brillo deseado.

3.5.1. Estructuras de datos

Estructuras de Datos	
Variable	Descripción
Est_Pres_TBrillo	Tipo Word, Puntero del estado Presente de la Tarea.
TimerBrillo	Tipo Byte, Timer en base de 100 ms.
Brillo	Tipo Byte, Contiene el nivel de brillo deseado de 0-100.
Valores y banderas	
Nombre	Descripción
SCF	Bit 7 de ATD0STAT0, indica si se completó el ciclo de conversion
TBrillo_EstX	Dirección de la subrutina del estado X.
tTimerBrillo	Valor para la carga TimerBrillo, con un valor de 3.

Tabla 12: Estructuras de Datos utilizados en la tarea Brillo

3.5.2. Diagramas de flujo

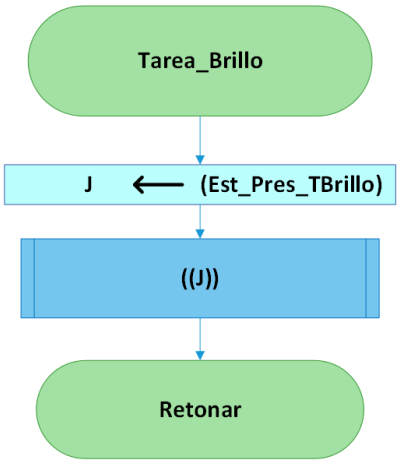

Figura 22: Diagrama de flujo de la tarea Brillo

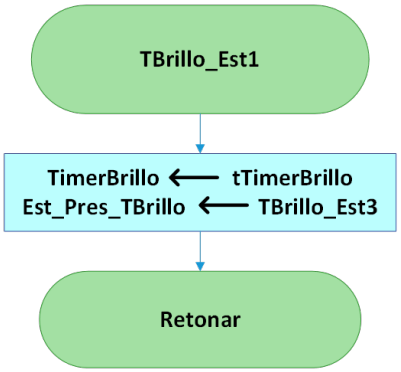

Figura 23: Diagrama de flujo del Estado 1 de la tarea Brillo

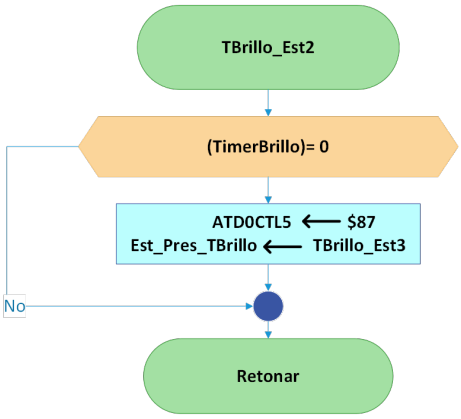

Figura 24: Diagrama de flujo del Estado 2 de la tarea Brillo

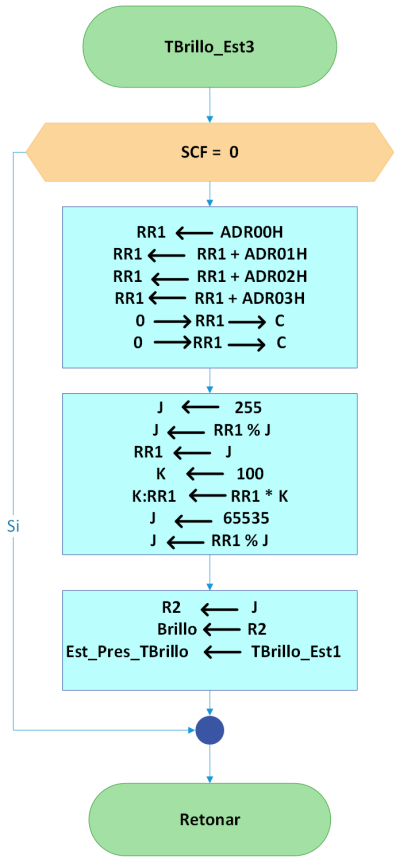

Figura 25: Diagrama de flujo del Estado 3 de la tarea Brillo

3.6. Tarea Pantalla Mux

La **Tarea** `PantallaMUX` es la encargada de gestionar el multiplexado dinámico de los cuatro dígitos del display de 7 segmentos del sistema. Dado que los dígitos comparten el mismo bus de datos, esta tarea controla qué dígito se enciende y durante cuánto tiempo, realizando una conmutación secuencial a alta velocidad para generar la ilusión de encendido simultáneo. Además, esta tarea implementa un mecanismo de control de brillo mediante modulación por temporización.

El sistema utiliza una **máquina de estados de dos pasos** para manejar el ciclo de encendido y apagado de cada dígito, ajustando su duración visible con base en un valor proporcional de brillo calculado previamente.

3.6.1. Estado 1 – Mostrar dígito activo

En este estado, se verifica si el temporizador `TimerDigito` ha finalizado. Si no ha expirado, el sistema espera; de lo contrario, se procede a:

- Recargar el temporizador `TimerDigito`.
- Consultar el valor del contador `Cont_Dig` para determinar cuál de los dígitos debe encenderse.
- Apagar todos los dígitos si el contador está fuera del rango válido.
- Encender el dígito correspondiente (DIG1–DIG4) mediante la activación de su línea de control (PTP) y el envío del patrón a `PORTB`.

- Incrementar el valor del contador `Cont_Dig`.
- Establecer el valor de `Counter_Ticks` para la fase de apagado.
- Avanzar al Estado 2.

3.6.2. Estado 2 – Control de apagado y brillo

Este estado se encarga de apagar el dígito activo luego de un periodo proporcional al nivel de brillo establecido.

Parámetros de entrada:

- `Dps1, Dps2, Dps3, Dps4` Accedidos en memoria, son los datos a mostrar en los displays.
- `Leds` Accedido en memoria, son los datos a mostrar en los Leds.
- `Brillo` Accedido en memoria, El el nivel de brillo deseado.

3.6.3. Estructuras de Datos

Estructuras de Datos	
Variable	Descripción
Est_Pres_PantallaMux	Tipo Word, Puntero del estado Presente de la Tarea.
TimerDigito	Tipo Byte, Timer en base de 1 ms.
Brillo	Tipo Byte, Contiene el nivel de brillo deseado de 0-100.
Cont_Dig	Tipo Byte, Indica cual diplays se va mostrar.
Dsp1	Tipo Byte, Contenido a mostrar en el display 1.
Dsp2	Tipo Byte, Contenido a mostrar en el display 2.
Dsp3	Tipo Byte, Contenido a mostrar en el display 3.
Dsp4	Tipo Byte, Contenido a mostrar en el display 4.
Counter_ticks	Tipo Word, Timer para la regulacion de Brillo.
Leds	Tipo Byte, Variable a cual se muestra en los Leds.
Valores y banderas	
Nombre	Descripción
PantallaMux_EstX	Dirección de la subrutina del estado X.
tTimerDigito	Valor para la carga TimerDigito, con un valor de 2.
MaxCounterTicks	Valor maximo de CounterTicks, con un valor de 100.

Tabla 13: Estructuras de Datos utilizados en la tarea Pantalla MUX

3.6.4. Diagramas de Flujo

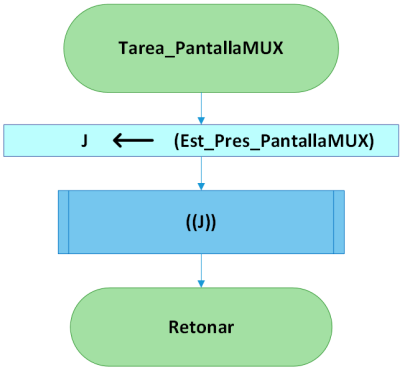

Figura 26: Diagrama de flujo de la tarea Pantalla Mux

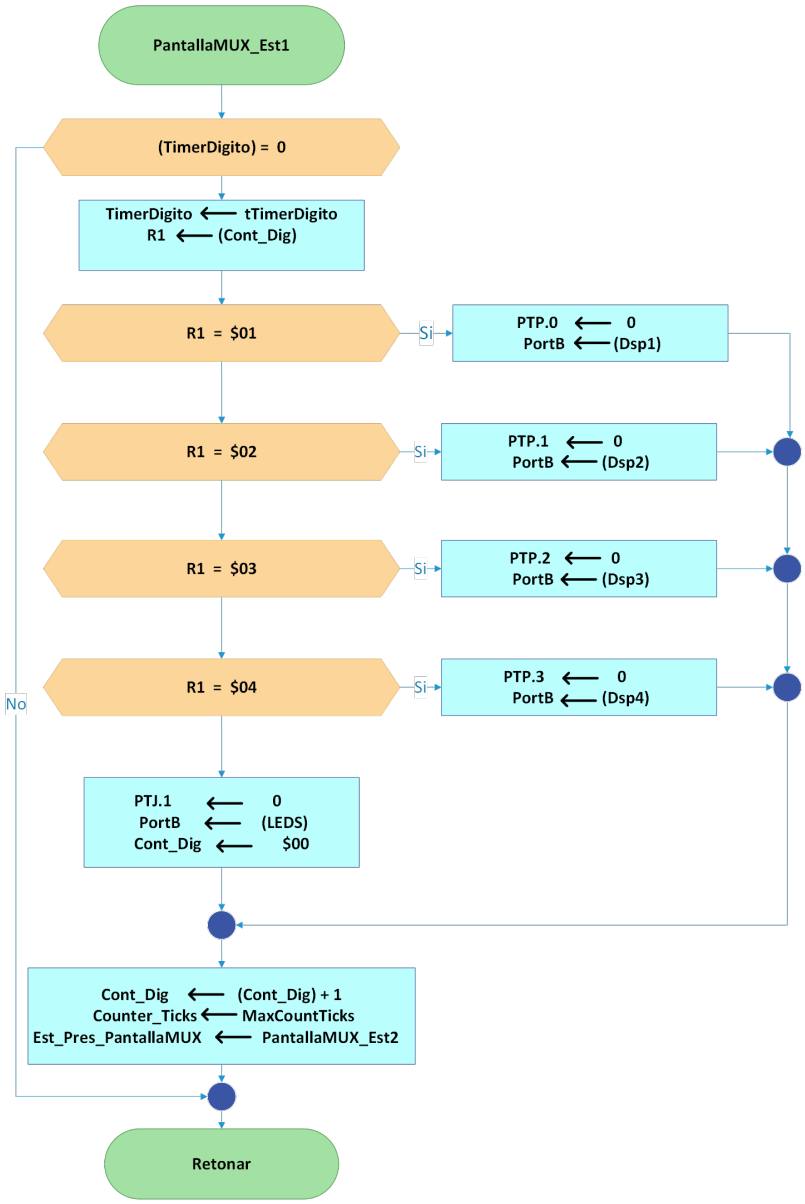

Figura 27: Diagrama de flujo del Estado 1 de la tarea Pantalla Mux

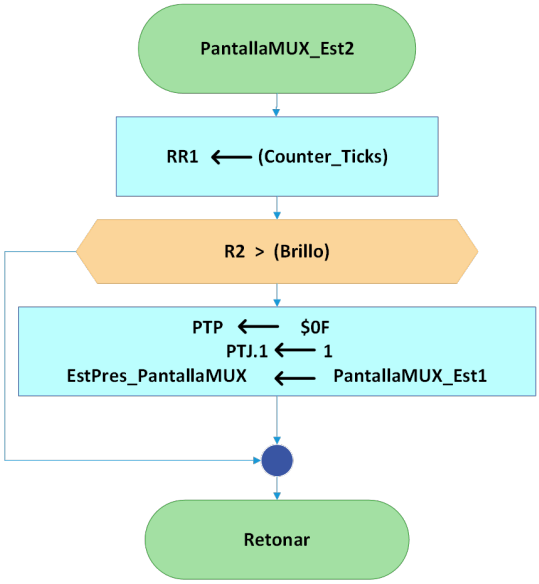

Figura 28: Diagrama de flujo del Estado 2 de la tarea Pantalla Mux

3.7. Tarea SendLCD

La **Tarea** `SendLCD` se encarga de enviar un byte al display LCD del sistema, ya sea como comando o como carácter. Dado que el LCD opera en **modo de 4 bits**, el byte se divide en dos mitades: el nibble alto y el nibble bajo. Cada nibble se transmite secuencialmente a través del puerto `PortK`, acompañado por las señales de control `RS` y `ENABLE`.

Esta tarea está implementada como una **máquina de estados de cuatro pasos**, sincronizada mediante temporizadores que garantizan los retardos necesarios entre las operaciones de escritura.

3.7.1. Estado 1 – Envío del nibble alto

- Se extraen los 4 bits más significativos de `CharLCD` y se colocan en `PortK`.
- Se configura el bit `RS` (0 para comandos, 1 para datos).
- Se activa el bit `ENABLE` para habilitar la escritura.
- Se inicia el temporizador `Timer260uS`.
- Se transiciona al Estado 2.

3.7.2. Estado 2 – Envío del nibble bajo

- Se espera a que finalice `Timer260uS`.
- Se desactiva `ENABLE`.
- Se extraen los 4 bits menos significativos de `CharLCD`, se desplazan y se colocan en `PortK`.
- Se configura nuevamente el bit `RS`.
- Se activa `ENABLE`.

- Se recarga `Timer260uS`.
- Se transiciona al Estado 3.

3.7.3. Estado 3 – Finalización de ENABLE

- Se espera la expiración del `Timer260uS`.
- Se desactiva `ENABLE`.
- Se inicia `Timer40uS`.
- Se transiciona al Estado 4.

3.7.4. Estado 4 – Finalización del ciclo de envío

- Se espera a que termine `Timer40uS`.
- Se establece la bandera `FinSendLCD`, que indica que la transmisión ha finalizado correctamente.
- Se retorna al Estado 1 para esperar un nuevo dato.

Parámetros de entrada: Todos se acceden por memoria

- `CharLCD`: Contiene el byte que se desea enviar al LCD.
- `RS`: Bandera que indica si el contenido de `CharLCD` es un comando o un carácter.

Parámetros de Salida:

- `FinSendLCD`: Bandera que indica que el proceso de envío del dato ha finalizado correctamente.

3.7.5. Estructuras de Datos

Estructuras de Datos	
Variable	Descripción
Est_Pres_SendLCD	Tipo Word, Puntero del estado Presente de la Tarea.
CharLCD	Tipo Byte, Dato a mandar al LCD.
Timer260us	Tipo Word, Timer para medir 260 μs
Timer40us	Tipo Word, Timer para medir 40 μs
Valores y banderas	
Nombre	Descripción
SendLCD_Estx	Dirección de la subrutina del estado X.
RS	Bandera, indica si se va a mandar un dato o comando.
FinSendLCD	Bandera, indica el fin del envio de un dato.
tTimer260us	Valor para la carga Timer260us, con un valor de 13.
tTimer40us	Valor para la carga Timer40us, con un valor de 2.

Tabla 14: Estructuras de Datos utilizados en la tarea SendLCD

3.7.6. Diagrama de flujo

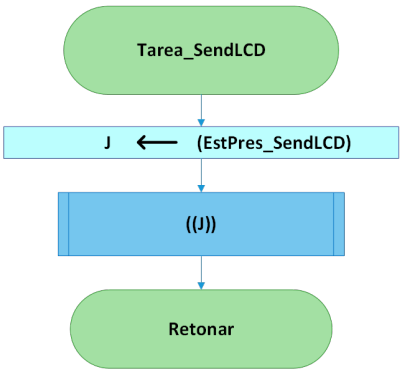

Figura 29: Diagrama de flujo de la tarea Send LCD

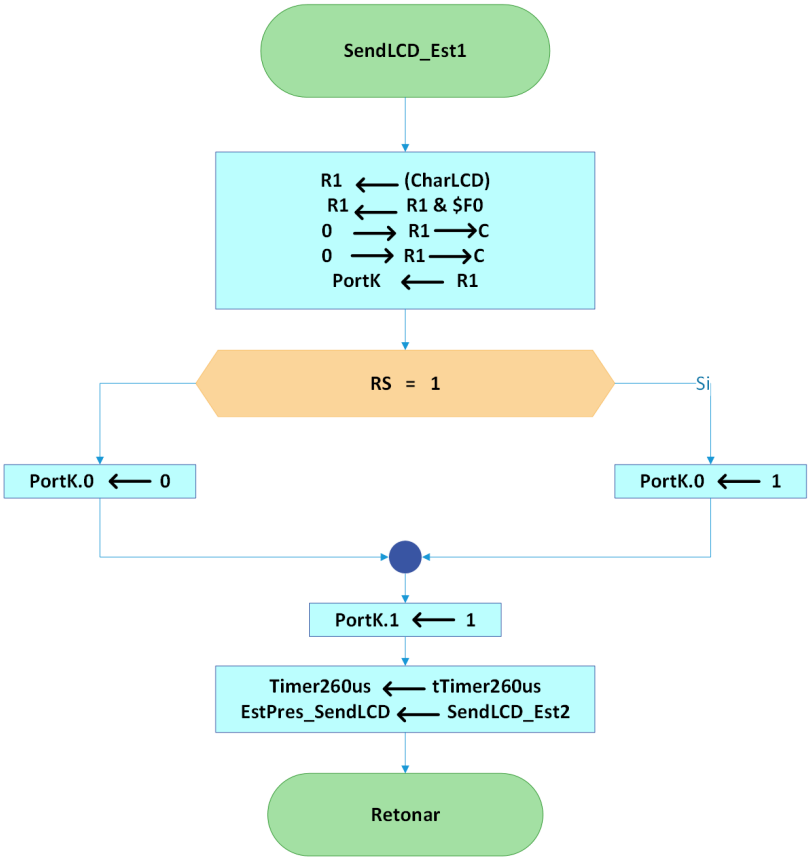

Figura 30: Diagrama de flujo del Estado 1 de la tarea SendLCD

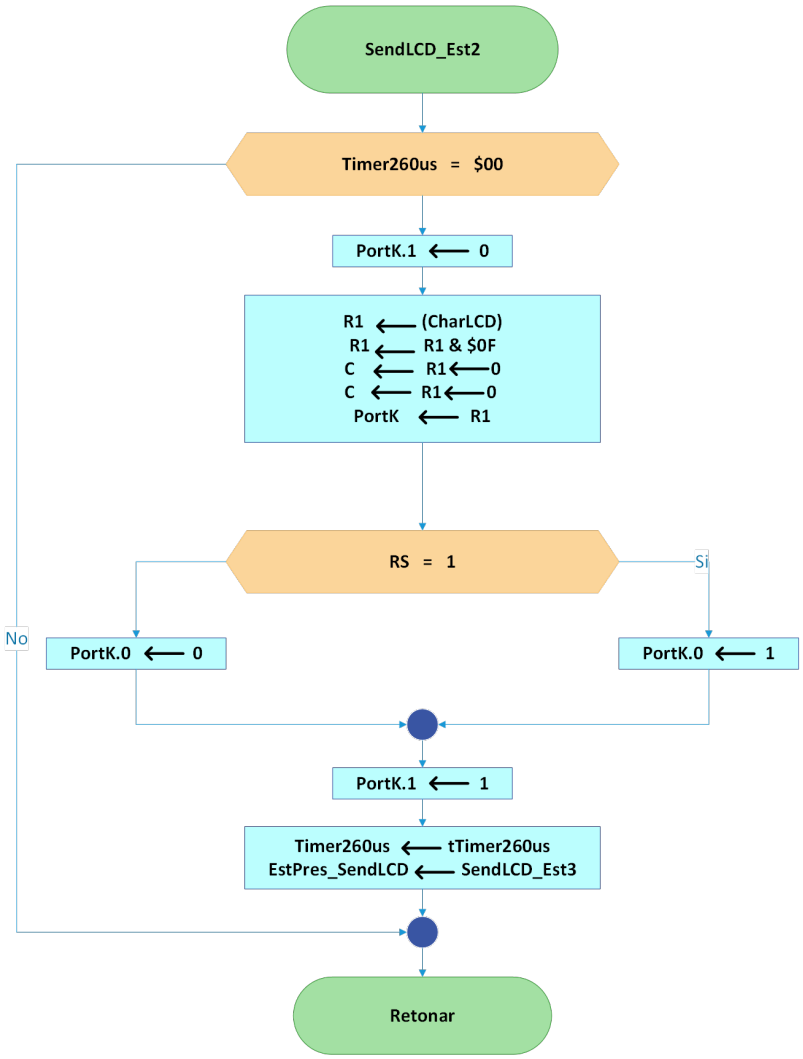

Figura 31: Diagrama de flujo del Estado 2 de la tarea SendLCD

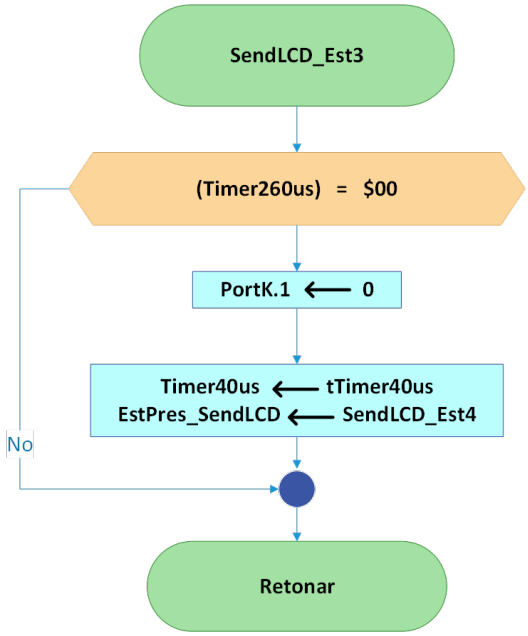

Figura 32: Diagrama de flujo del Estado 3 de la tarea SendLCD

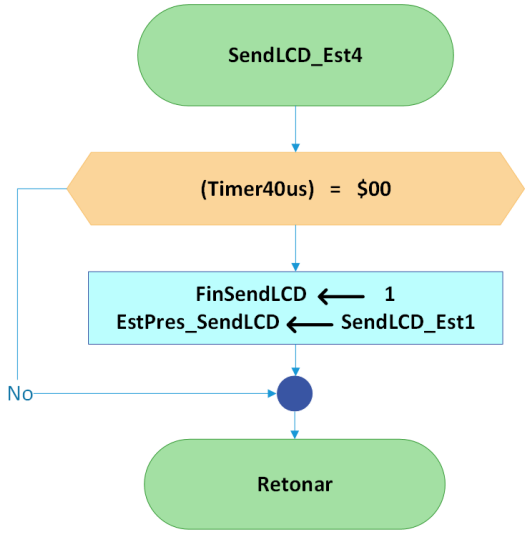

Figura 33: Diagrama de flujo del Estado 4 de la tarea SendLCD

3.8. Tarea LCD

La **Tarea** `TareaLCD` es la encargada de gestionar el envío secuencial de mensajes a la pantalla LCD del sistema. Esta tarea trabaja en conjunto con la subrutina `Tarea_SendLCD`, que se encarga de transmitir cada byte al LCD utilizando el protocolo de comunicación en modo de 4 bits.

El proceso se organiza como una **máquina de estados de dos pasos**, que permite enviar primero el contenido de la línea 1 y luego el de la línea 2. El control se basa en el uso de banderas y temporizadores para garantizar el correcto posicionamiento del cursor y la transmisión sin bloqueos.

3.8.1. Estado 1 – Posicionamiento del cursor

- Se limpian las banderas `FinSendLCD` y `RS`.
- Se evalúa la bandera `Second_line`:
  - Si está apagada, se carga la dirección de la línea 1 (`ADD_L1`) en `CharLCD` y se apunta a `MSG_L1`.
  - Si está encendida, se carga `ADD_L2` y se apunta a `MSG_L2`.
- Se llama a `Tarea_SendLCD` para enviar la instrucción de posicionamiento.
- Se transiciona al Estado 2.

3.8.2. Estado 2 – Envío del contenido del mensaje

- Se espera la finalización del envío anterior, comprobando la bandera `FinSendLCD`.
- Se limpia dicha bandera y se activa la bandera `RS` (modo datos).
- Se carga el siguiente carácter desde la dirección apuntada por `Punt_LCD`.
- Si el carácter no es `EOB`, se llama a `Tarea_SendLCD` y se permanece en este estado.
- Si el carácter es `EOB`:
  - Si se estaba enviando la línea 1, se activa `Second_line` y se vuelve al Estado 1.
  - Si se estaba enviando la línea 2, se limpia `Second_line`, se activa la bandera `LCD_OK` y se vuelve al Estado 1.

Parámetros de entrada: Todos se acceden por memoria

- `MSG_L1`, `MSG_L2`: Direcciones en memoria que contienen los mensajes para la línea 1 y línea 2 del LCD, respectivamente.
- `ADD_L1`, `ADD_L2`: Comandos para posicionar el cursor al inicio de la línea 1 o línea 2.
- `Banderas_2`: Contiene los bits `RS`, `FinSendLCD`, `Second_Line` y `LCD_OK`, que controlan el flujo de escritura.

Parámetros de Salida:

- `Banderas_2.LCD_OK`: Se activa cuando ambos mensajes (línea 1 y línea 2) se han escrito completamente en el LCD.

3.8.3. Estructuras de Datos

Estructuras de Datos	
Variable	Descripción
EstPres_TareaLCD	Tipo Word, Puntero del estado Presente de la Tarea.
CharLCD	Tipo Byte, Dato a mandar al LCD.
PuntLCD	Tipo Word, Puntero para barrer el mensaje a enviar
Valores y banderas	
Nombre	Descripción
TareaLCD_Estx	Dirección de la subrutina del estado X.
RS	Bandera, indica si se va a mandar un dato o comando.
FinSendLCD	Bandera, indica el fin del envio de un dato.
Second_Line	Bandera, indica si se envia en la segunda linea del LCD.
EOB	Valor de $FF, indica el final de mensaje

Tabla 15: Estructuras de Datos utilizados en la tarea TareaLCD

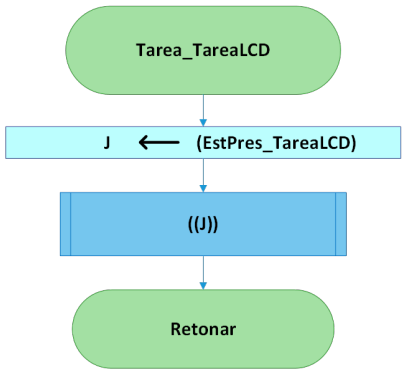

Figura 34: Diagrama de flujo de la tarea Tarea LCD

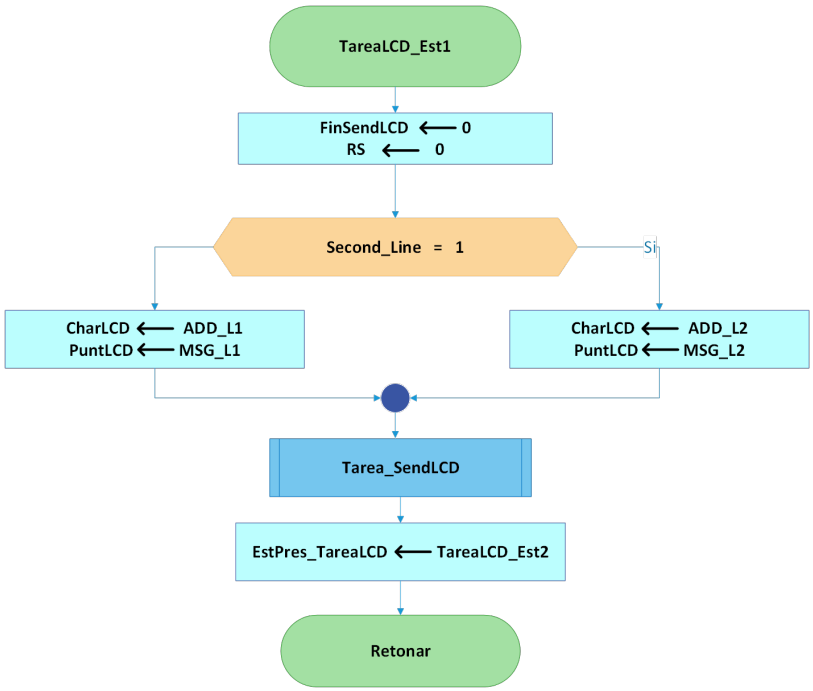

Figura 35: Diagrama de flujo del Estado 1 de la tarea TareaLCD

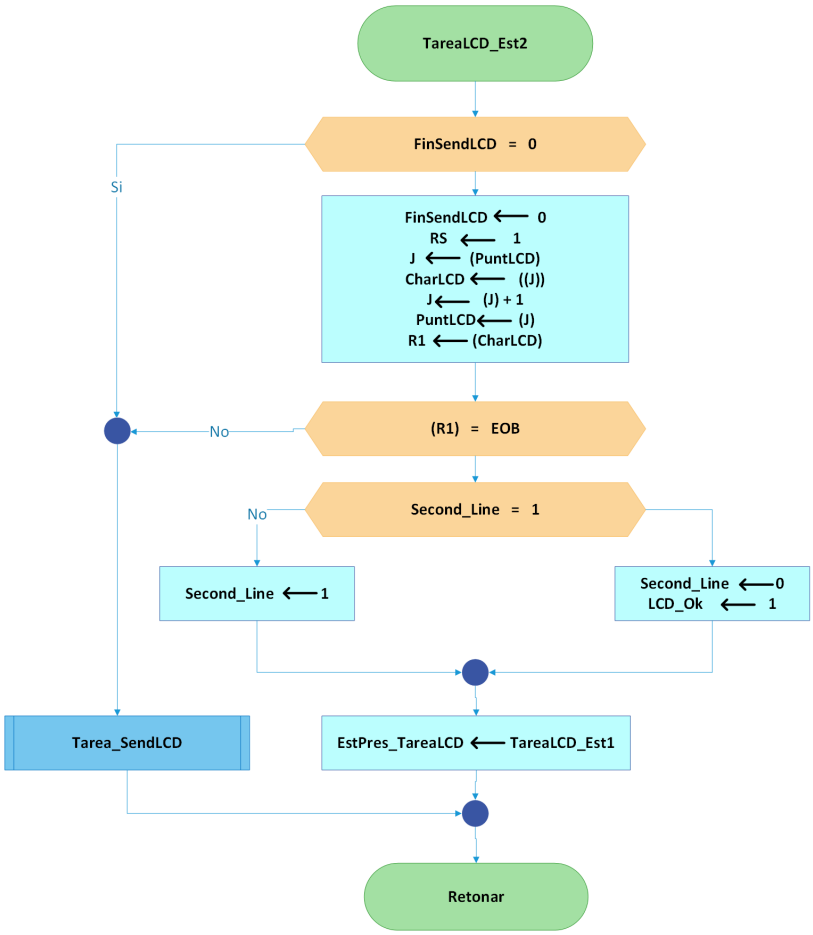

Figura 36: Diagrama de flujo del Estado 2 de la tarea TareaLCD

3.9. Tarea LeerPB

La **Tarea** `LeerPB` implementa una máquina de estados para gestionar la lectura segura del botón físico PB. Esta tarea permite detectar si una pulsación fue **corta (ShortPress)** o **larga (LongPress)**, con el uso de temporizadores y control de rebote mecánico.

3.9.1. Estado 1 – Detección inicial de pulsación

- Se verifica si el botón PB ha sido presionado (`brset PortPB,MaskPB1`).
- Si no ha sido presionado, se permanece en este estado.
- Si fue presionado:
  - Se cargan los temporizadores de rebote (`Timer_RebPB1`), pulsación corta (`Timer_SHP1`) y pulsación larga (`Timer_LP1`).
  - Se transiciona al Estado 2.

3.9.2. Estado 2 – Supresión de rebote

- Se espera la expiración del temporizador `Timer_RebPB1`.

- Una vez vencido, se verifica si el botón aún permanece presionado.
- Si no lo está, se considera rebote y se retorna al Estado 1.
- Si continúa presionado, se transiciona al Estado 3.

3.9.3. Estado 3 – Clasificación de ShortPress

- Se espera que finalice el temporizador `Timer_SHP1`, que define el umbral mínimo para clasificar como pulsación corta.
- Si el botón se libera antes de ese tiempo, se activa la bandera `ShortP1` y se retorna al Estado 1.
- Si el botón permanece presionado tras ese tiempo, se pasa al Estado 4.

Estado 4 – Clasificación de LongPress

- Se evalúa si el temporizador `Timer_LP1` ha finalizado.
- Si el botón aún está presionado al vencer `Timer_LP1`, se activa la bandera `LongP1`.
- Si el botón se libera antes de ese tiempo, se activa la bandera `ShortP1`.
- Luego de marcar la pulsación correspondiente, se retorna al Estado 1.

Parámetros de Salida:

- `Banderas_1.ShortP`: Bandera que se activa cuando se detecta una pulsación corta.
- `Banderas_1.LongP`: Bandera que se activa cuando se detecta una pulsación larga.

3.9.4. Estructuras de Datos

Estructuras de Datos	
Variable	Descripción
EstPres_LeerPB	Tipo Word, Puntero del estado Presente de la Tarea.
Timer_RebPB	Tipo Byte, Timer para supresion de rebotes.
Timer_SHP	Tipo Byte, Timer para ShortPress
Timer_LP	Tipo Byte, Timer para LongPress
Valores y banderas	
Nombre	Descripción
LeerPB_Estx	Dirección de la subrutina del estado X.
tSubRebPB	Valor para la carga del Timer_RebPB.
tShortP	Valor para la carga del Timer_SHP.
tLongP	Valor para la carga del Timer_LP.
LongP	Bandera, Indica se introduce un LongP
ShortP	Bandera, Indica se introduce un ShortP
PB	Bit 0 o 3 del puerto H donde se ubica el botón.

Tabla 16: Estructuras de Datos utilizados en la tarea LeerPB

3.9.5. Diagramas de flujo

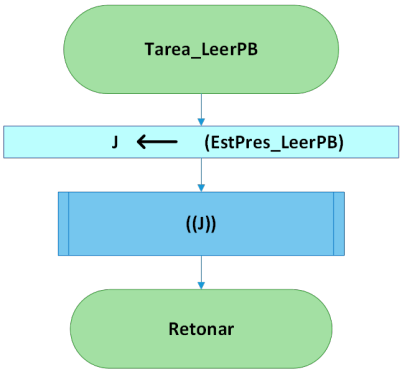

Figura 37: Diagrama de flujo de la Tarea LeerPB

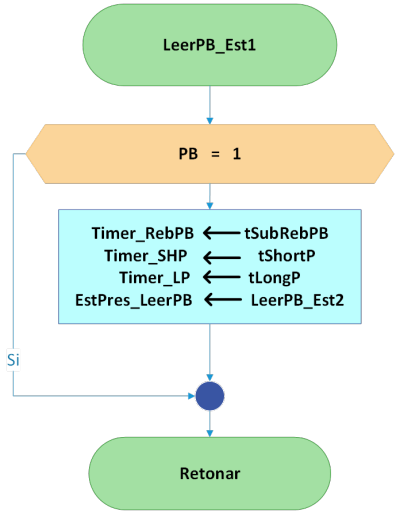

Figura 38: Diagrama de flujo del Estado 1 de la Tarea LeerPB

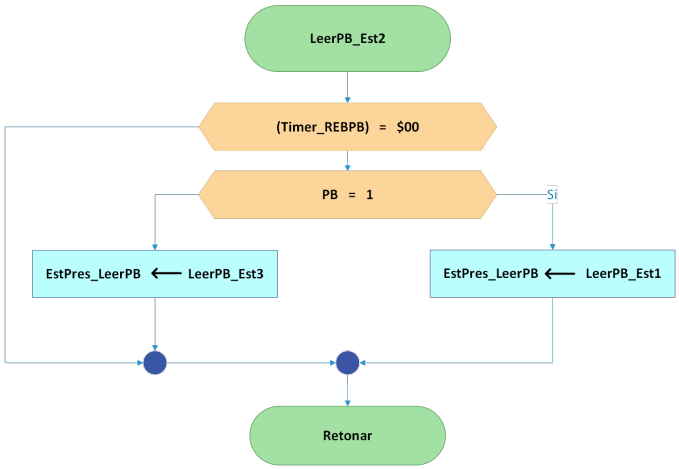

Figura 39: Diagrama de flujo del Estado 2 de la Tarea LeerPB

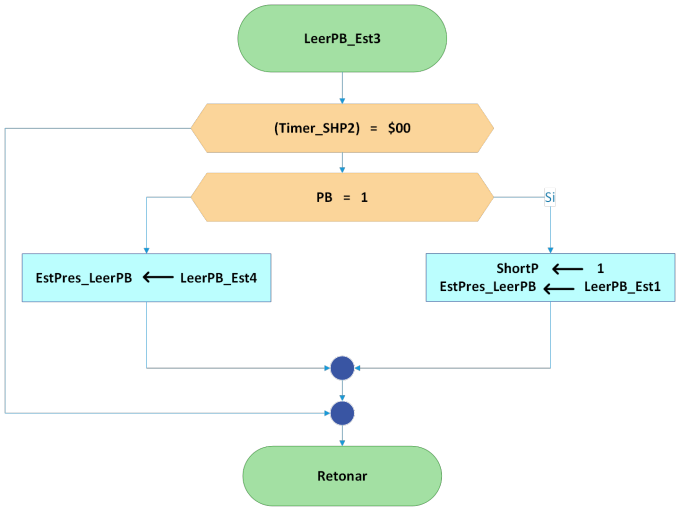

Figura 40: Diagrama de flujo del Estado 3 de la Tarea LeerPB

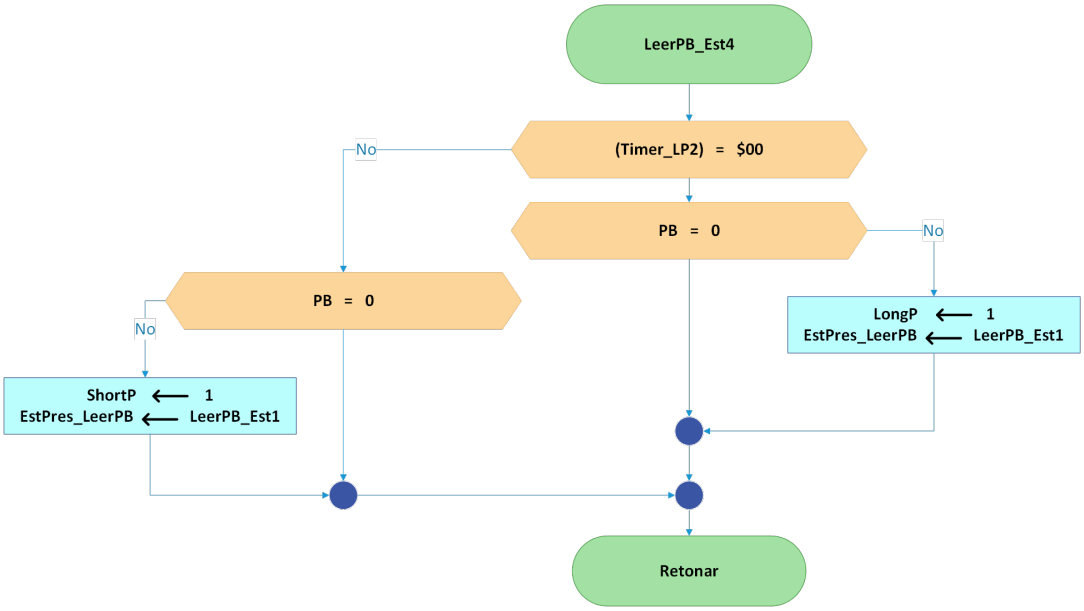

Figura 41: Diagrama de flujo del Estado 4 de la Tarea LeerPB

3.10. Tarea Teclado

La **Tarea** `Tarea_Teclado` se encarga de gestionar la lectura del teclado matricial, aplicando técnicas de supresión de rebotes, validación de entradas y almacenamiento secuencial de los valores ingresados. Esta tarea opera como una **máquina de estados de cuatro pasos**, que garantiza una lectura robusta y evita múltiples registros por una misma pulsación.

El valor ingresado se almacena en el arreglo `Num_Array`, el cual representa una secuencia teclas ingresadas. Esta entrada puede incluir caracteres numéricos y teclas especiales como `BORRAR` o `ENTER`, con las respectivas acciones que cada una desencadena.

Estado 1 – Detección de tecla

- Se llama a la subrutina `Leer_Tecla`.
- Si no se detecta ninguna tecla (`Tecla = $FF`), se permanece en este estado.
- Si se detecta una tecla, se guarda el valor en `Tecla_IN`, se inicia el temporizador `Timer_RebTCL` y se transiciona al estado 2.

Estado 2 – Supresión de rebotes

- Se espera a que expire el temporizador de rebote.
- Se verifica si la tecla aún está siendo presionada.
- Si la pulsación continúa, se pasa al estado 3.
- Si la tecla fue liberada o cambió, se retorna al estado 1.

Estado 3 – Espera de liberación

- Se monitorea continuamente la liberación de la tecla.
- Cuando `Tecla = $FF`, es decir, la tecla ha sido soltada, se transiciona al estado 4.

Estado 4 – Procesamiento de la tecla

- Se evalúa el contenido de `Tecla_IN` para determinar la acción a ejecutar:
  - Si es un valor numérico y no se ha superado el máximo (`MAX_TCL`), se guarda en `Num_Array`.
  - Si es `BORRAR`, se elimina el último dígito ingresado.
  - Si es `ENTER`, se limpia el contador y se activa la bandera `ArrayOK`, indicando que la entrada está completa.
- Al finalizar cualquier acción, se borra `Tecla_IN` y se retorna al estado 1.

Parámetros de entrada:

- `Tecla`: Variable que almacena la tecla leída por la subrutina `Leer_Tecla`.
- `MAX_TCL`: Número máximo de teclas permitidas para ingresar.
- `Num_Array`: Dirección base del arreglo donde se almacenan las teclas ingresadas.
- `Cont_TCL`: Contador que lleva la cantidad de teclas ingresadas.

Parámetros de Salida:

- `Num_Array`: Arreglo de memoria donde se guarda cada tecla ingresada.

3.10.1. Estructuras de Datos

Estructuras de Datos	
Variable	Descripción
Est_Pres_TCL	Tipo Word, Puntero del estado Presente de la Tarea.
Tecla	Tipo Byte, Tecla Ingresada por el usuario
Tecla_IN	Tipo Byte, Variable auxiliar para tecla
Timer_RebTCL	Tipo Byte, Timer para suprimir Rebotes
Num_Array	Arreglo de teclas validas ingresadas
Cont_TCL	Tipo Byte, Numero de teclas ingresadas
MAX_TCL	Tipo Byte, longuitud maxima de num_array
Valores y banderas	
Nombre	Descripción
TCL_Estx	Dirección de la subrutina del estado X.
ArrayOk	Bandera, indica si se ingreso una secuencia de teclas validas
BORRAR	Valor de $0B, indica borrar la ultima tecla ingresada
ENTER	Valor de $0E, indica que se ingresa las teclas presionadas

Tabla 17: Estructuras de Datos utilizados en la tarea Teclado

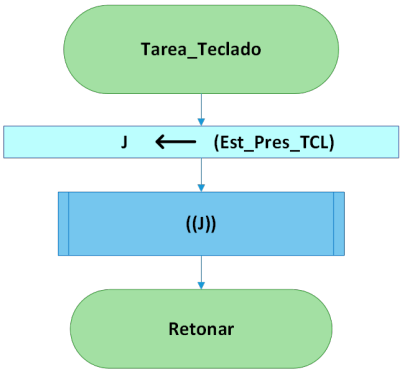

Figura 42: Diagrama de flujo de la Tarea Teclado

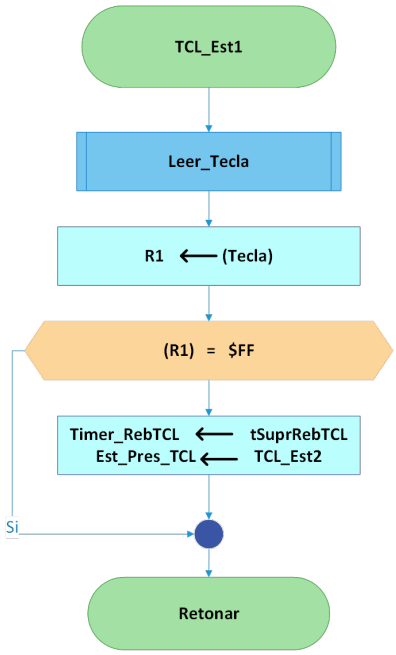

Figura 43: Diagrama de flujo del Estado 1 de la Tarea Teclado

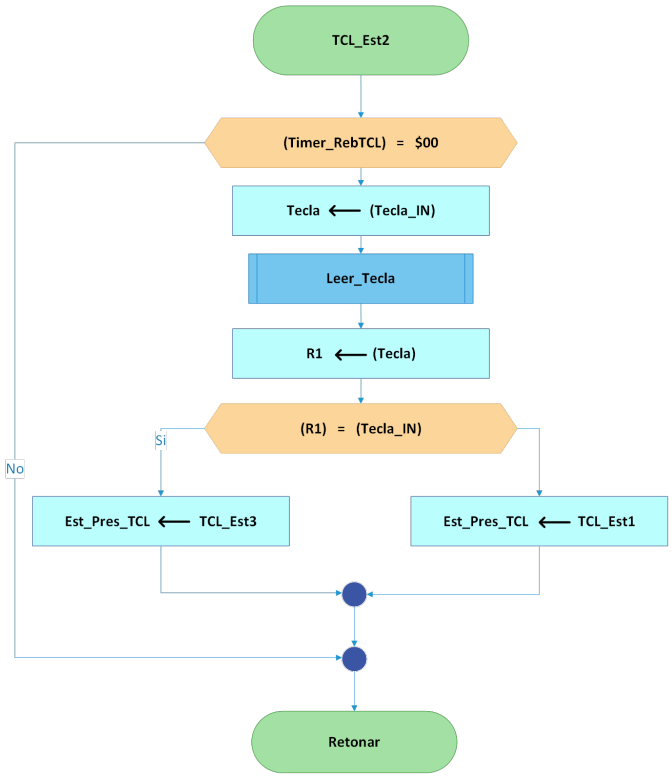

Figura 44: Diagrama de flujo del Estado 2 de la Tarea Teclado

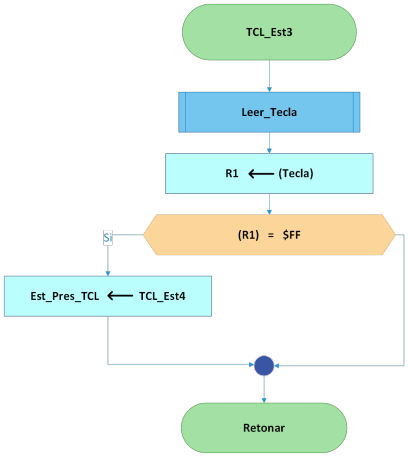

Figura 45: Diagrama de flujo del Estado 3 de la Tarea Teclado

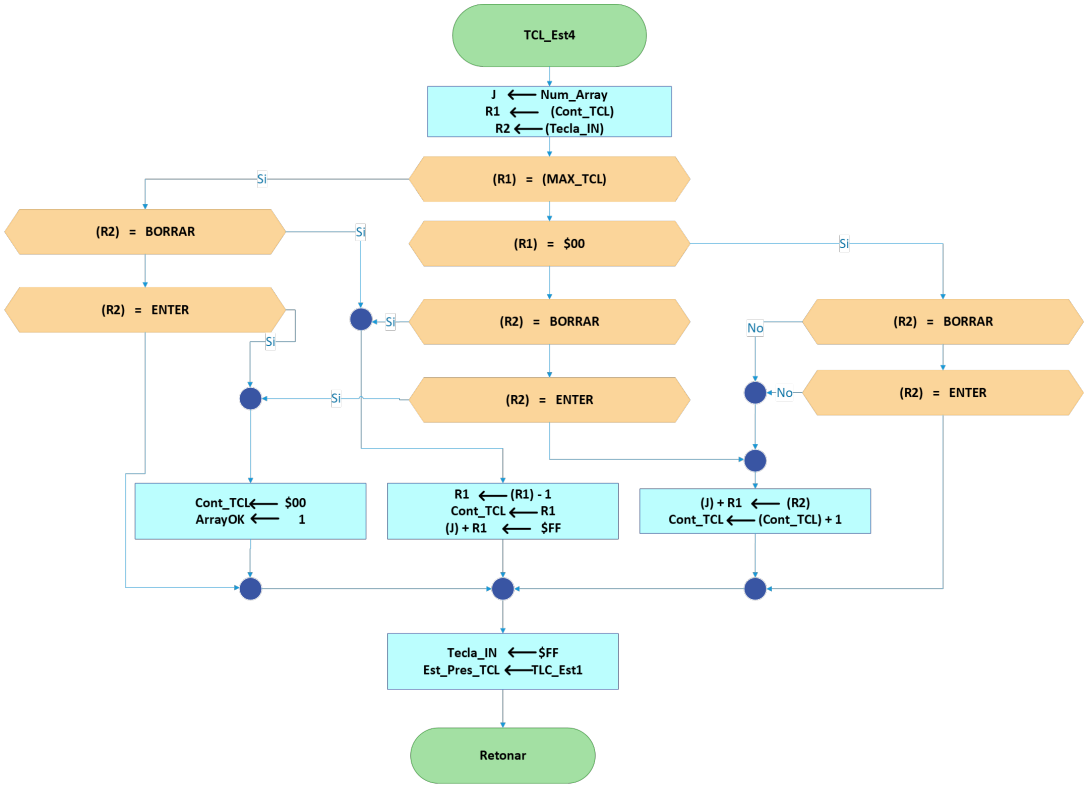

Figura 46: Diagrama de flujo del Estado 4 de la Tarea Teclado

3.11. Tarea Led Testigo

La **Tarea** `Tarea_Led_Testigo` tiene como objetivo implementar un mecanismo visual de parpadeo secuencial utilizando tres LEDs conectados al puerto `PTP` de la tarjeta Dragon 12+. Esta tarea opera como una **máquina de estados de tres pasos**, alternando entre los colores rojo, verde y azul en intervalos regulares. El parpadeo cíclico de los LEDs puede ser utilizado como indicador del estado activo del sistema o como señal de prueba del correcto funcionamiento de los temporizadores de la maquina de Tiempos.

Estado 1 – Encendido del LED Rojo

- Se mantiene este estado hasta que el temporizador asignado expire.
- Una vez transcurrido el tiempo de espera, se enciende el LED rojo y se apaga el LED azul.
- Se establece el siguiente estado de la máquina como el estado 2.
- Se reinicia el temporizador para iniciar un nuevo intervalo de espera.

Estado 2 – Encendido del LED Verde

- Este estado también se mantiene activo hasta la expiración del temporizador.
- Al finalizar el tiempo correspondiente, se activa el LED verde y se apaga el LED rojo.
- El estado siguiente se define como el estado 3.
- El temporizador es recargado para continuar con la secuencia.

Estado 3 – Encendido del LED Azul

- Se espera nuevamente la finalización del tiempo programado en el temporizador.
- Finalizado este periodo, se enciende el LED azul y se apaga el LED verde.
- El sistema vuelve al estado 1, reiniciando el ciclo de parpadeo.
- Se reinicia el temporizador con el valor predefinido.

3.11.1. Estructuras de Datos

Estructuras de Datos	
Variable	Descripción
Est_Pres_LDTst	Tipo Word, Puntero del estado Presente de la Tarea.
Timer_LED_Testigo	Tipo Byte, Timer en base de 100 ms
Valores y banderas	
Nombre	Descripción
LDTst_Estx	Dirección de la subrutina del estado X.
tTimerLDTst	Valor para cargar el timer de led testigo a 500 ms
LD_RED	Valor de $10, bit 4 del Puerto P para el Led Rojo
LD_BLUE	Valor de $20, bit 5 del Puerto P para el Led Azul
LD_GREEN	Valor de $40, bit 6 del Puerto P para el Led Verde

Tabla 18: Estructuras de Datos utilizados en la tarea Led Testigo

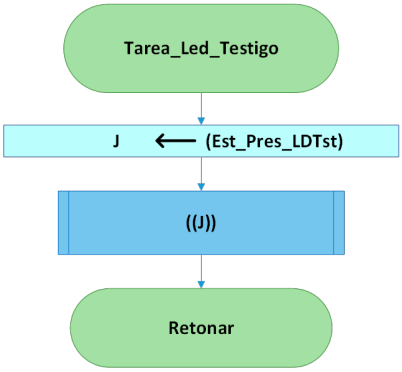

Figura 47: Diagrama de flujo de la Tarea Led Testigo

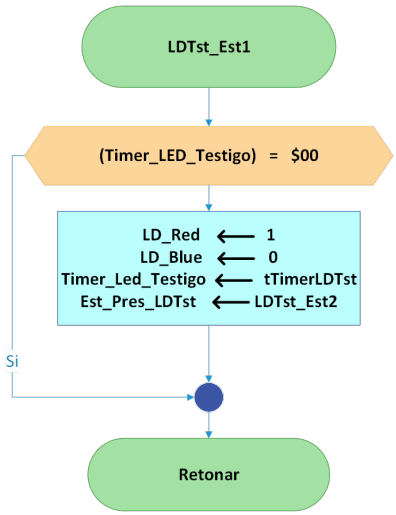

Figura 48: Diagrama de flujo del Estado 1 de la Tarea Led Testigo

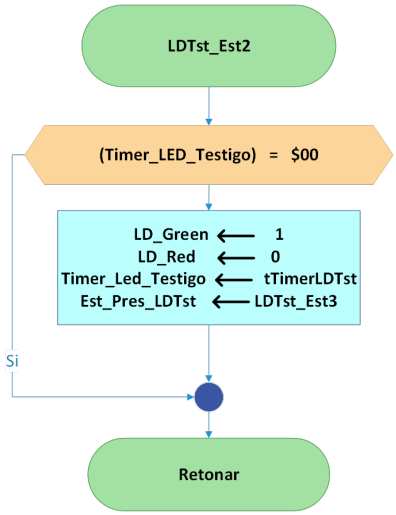

Figura 49: Diagrama de flujo del Estado 2 de la Tarea Led Testigo

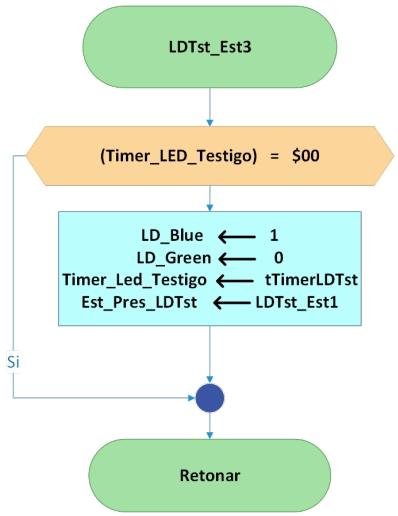

Figura 50: Diagrama de flujo del Estado 3 de la Tarea Led Testigo

4. Subrutinas

4.1. Bin_BCD_MUXP

Esta subrutina tiene como objetivo convertir un valor binario (contenido en el acumulador A) a su representación en BCD y almacenarlo en la variable `BCD`. Esta conversión es útil para desplegar el número en dispositivos como pantallas de 7 segmentos.

Funcionamiento General: El algoritmo implementado se basa en el método de X3. Este algoritmo convierte números binarios en BCD sin necesidad de realizar divisiones.

El procedimiento consiste en desplazar bit a bit el contenido del acumulador A hacia la izquierda, e insertar estos bits en la variable `BCD`, mientras se realizan ajustes condicionales sumando 3 a cada nibble si su valor es mayor o igual que 5. Este ajuste garantiza que la representación final en `BCD` cumpla con el formato decimal codificado en binario.

Parámetros de Entrada:

- Acumulador A: contiene el número binario (de 0 a 99) a convertir.

Parámetros de Salidas:

- Variable `BCD`: Se accede en memoria, contiene el número convertido a BCD empaquetado, donde el nibble alto representa las decenas y el nibble bajo las unidades.

4.1.1. Estructuras de Datos

Estructuras de Datos	
Variable	Descripción
Cont_BCD	Tipo Byte, Contiene el numero de desplazamientos a realizar
BCD	Tipo Byte, Variable donde se guarda el resultado de la subrutina.

Tabla 19: Estructuras de Datos utilizados en la Subrutina Bin_BCD_MUXP

4.1.2. Diagrama de flujo

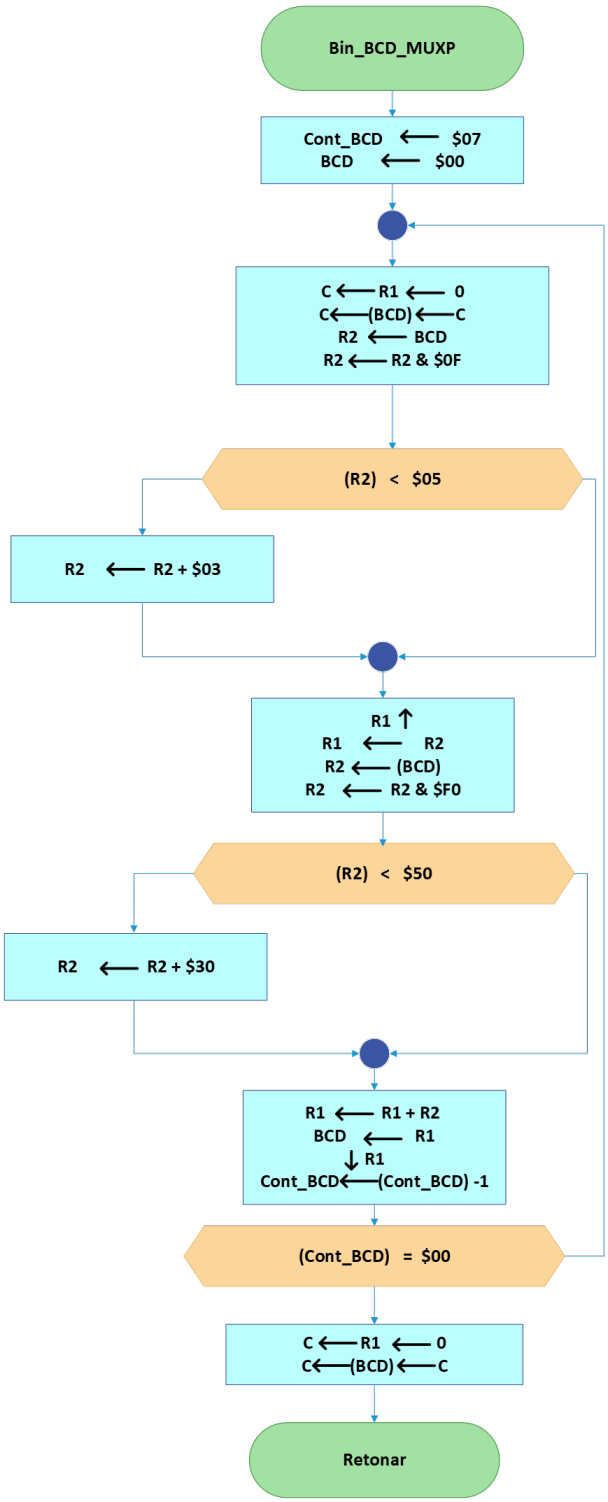

Figura 51: Diagrama de flujo de la subrutina Bin_BCD_BCD

4.2. BCD_7Seg

La subrutina `BCD_7Seg` toma dos bytes en formato BCD (almacenados en las variables `BCD1` y `BCD2`) y genera los patrones correspondientes para ser desplegados en una pantalla de 7 segmentos de 4 dígitos. Estos patrones se almacenan en las variables `Dsp1`, `Dsp2`, `Dsp3` y `Dsp4`, que corresponden a los cuatro dígitos del display. Cada byte en BCD contiene dos dígitos decimales: el nibble alto representa las decenas y el nibble bajo las unidades. La subrutina separa cada nibble utilizando desplazamientos y máscaras lógicas, y luego utiliza ese valor como índice en la tabla `Segment`, que contiene los patrones binarios de activación de segmentos para los números del 0 al 9 y caracteres adicionales como guiones o apagado.

Parámetros de Entrada:

- `BCD1`, `BCD2`: bytes en formato BCD que contienen los valores numéricos a mostrar, accedidos en memoria.
- `Segment`: tabla con los patrones de encendido para los dígitos del 0 al 9 y símbolos de apagado y guion.

Parámetros de Salida:

- `Dsp1`, `Dsp2`, `Dsp3`, `Dsp4`: variables que contienen los patrones binarios para mostrar en cada uno de los 4 dígitos del display, accedidos en memoria.

4.2.1. Estructuras de Datos

Estructuras de Datos	
Variable	Descripción
BCD1	Tipo Byte, Variable que contiene el numero BCD a mostrar.
BCD2	Tipo byte, Variable que contiene el numero BCD a mostrar.
Dsp1	Tipo Byte, Contenido a mostrar en el display 1.
Dsp2	Tipo Byte, Contenido a mostrar en el display 2.
Dsp3	Tipo Byte, Contenido a mostrar en el display 3.
Dsp4	Tipo Byte, Contenido a mostrar en el display 4.
Segment	Tabla de valores para los displays.

Tabla 20: Estructuras de Datos utilizados en la Subrutina BCD_7seg

4.2.2. Diagrama de flujo

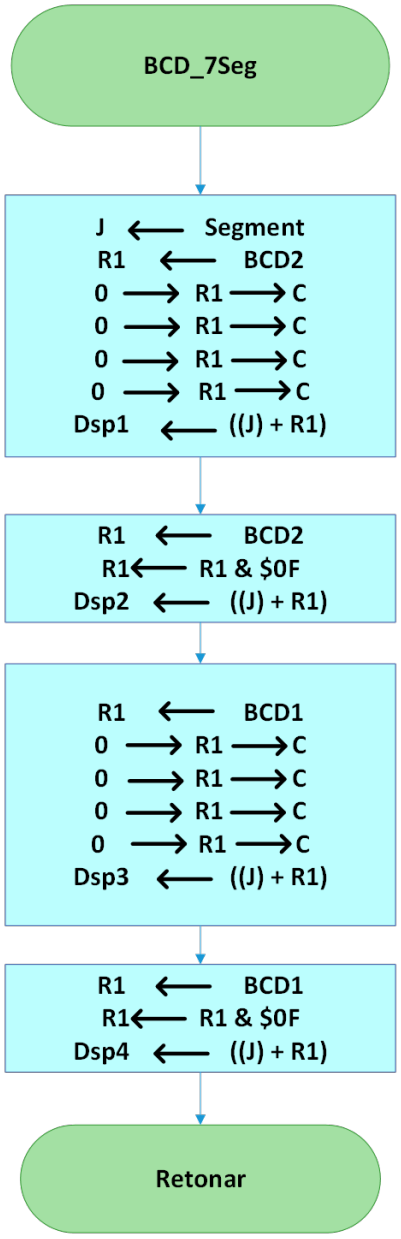

Figura 52: Diagrama de flujo de la subrutina BCD_7Seg

4.3. Calcula

La subrutina `Calcula` se encarga de calcular tres variables fundamentales para el proceso de separación de tuercas: `Velocidad`, `TimerIniPant` y `TimerFinPant`. Estos valores se derivan del tiempo que tarda una tuerca en viajar entre dos sensores (P1 y P2), así como de las distancias físicas del sistema.

Variables Calculadas:

- `DeltaT`: Tiempo transcurrido entre P1 y P2, en unidades de 100 ms.

- `Velocidad`: Velocidad de la tuerca, en cm/s.
- `TimerIniPant`: Tiempo que transcurre desde P2 hasta la posición de inicio del mensaje.
- `TimerFinPant`: Tiempo que transcurre desde P2 hasta la escobilla.

Ecuaciones Utilizadas:

- `DeltaT = tTimerCal - TimerCal`
- `Velocidad =` $\dfrac{\Delta S \cdot 10}{\Delta T}$, donde $\Delta S = 60$ cm (distancia entre sensores), y el factor 10 convierte unidades de 100 ms a segundos.
- `TimerIniPant =` $\dfrac{\Delta E - \Delta M - \Delta S}{Velocidad} \cdot 10$, donde $\Delta E = 160$ cm (distancia hasta la escobilla), $\Delta M = 60$ cm (distancia hasta el inicio de mensaje) y $\Delta S = 60$ cm (distancia de los sensores).
- `TimerFinPant =` $\dfrac{\Delta E - \Delta S}{Velocidad} \cdot 10$

Entradas:

- `TimerCal`: Tiempo actual del contador en base 100 ms.

Salidas:

- `DeltaT`: Intervalo de tiempo entre los sensores
- `Velocidad`: Magnitud de la velocidad de la tuerca, en cm/s.
- `TimerIniPant`: Tiempo hasta mostrar la velocidad y cantidad.
- `TimerFinPant`: Tiempo hasta ocultar los valores.

4.3.1. Estructuras de datos

Estructuras de Datos	
Variable	Descripción
TimerCal	Tipo Byte, Timer para calcular la velocidad.
DeltaT	Tipo Byte, Intervalo de tiempo entre los dos sensores.
Velocidad	Tipo Byte, Velocidad Calculada de la Tuerca.
TimerIniPant	Tipo Byte, Timer para el inicio de mensaje
TimerFinPant	Tipo Byte, Timer para el fin de mensaje
Valores y banderas	
Nombre	Descripción
DeltaS	Distancia entre los sensores 60 cm.
DeltaE	Distancia de la linea de empaque 160 cm.
DeltaM	Distancia para mostrar el mensaje 60 cm.
tTimerCal	Carga de TimerCal, es el valor maximo del timer.

Tabla 21: Estructuras de Datos utilizados en la Subrutina Calcula

4.3.2. Diagrama de flujo

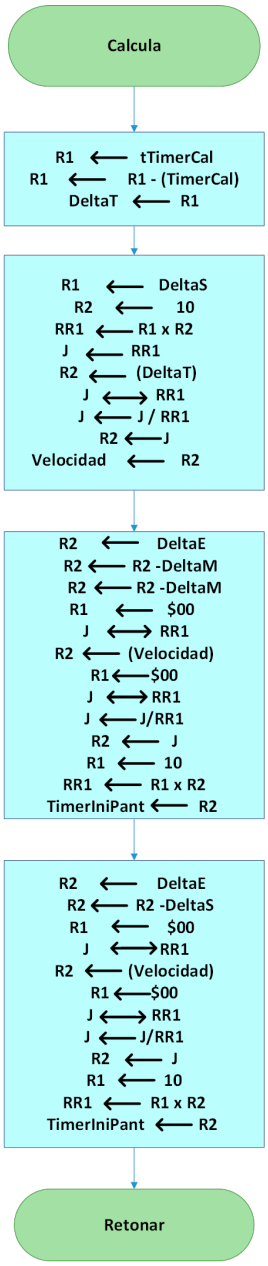

Figura 53: Diagrama de flujo de la subrutina Calcula

4.4. BCD_Bin

La subrutina `BCD_Bin` convierte un valor almacenado en formato BCD, a su equivalente en binario puro. El valor convertido es almacenado en la variable `ValorVelUmbral`, la cual se utiliza en el modo Separar para comparar velocidades y tomar decisiones de clasificación.

Funcionamiento General: El valor de entrada está dividido en dos registros: el acumulador A contiene las decenas y el acumulador B las unidades. El procedimiento sigue estos pasos:

- Si B contiene el valor `$FF`, se asume que solo hay unidades, y se copia el contenido de A

directamente a `ValorVelUmbral`.

- En caso contrario:
  - Se multiplica A (decenas) por 10.
  - Luego, se suma B (unidades).
  - El resultado total se almacena en `ValorVelUmbral` como un valor binario convencional.

Entradas:

- Acumulador D = (A:B): contiene el valor BCD a convertir, donde A son las decenas y B las unidades.

Salidas:

- `ValorVelUmbral`: contiene el valor entero equivalente en binario.

4.4.1. Estructuras de Datos

Estructuras de Datos	
Variable	Descripción
ValorVelUmbral	Tipo Byte, Variable que contiene el numero binario convertido.

Tabla 22: Estructuras de Datos utilizados en la Subrutina BCD_7seg

4.4.2. Diagrama de flujo

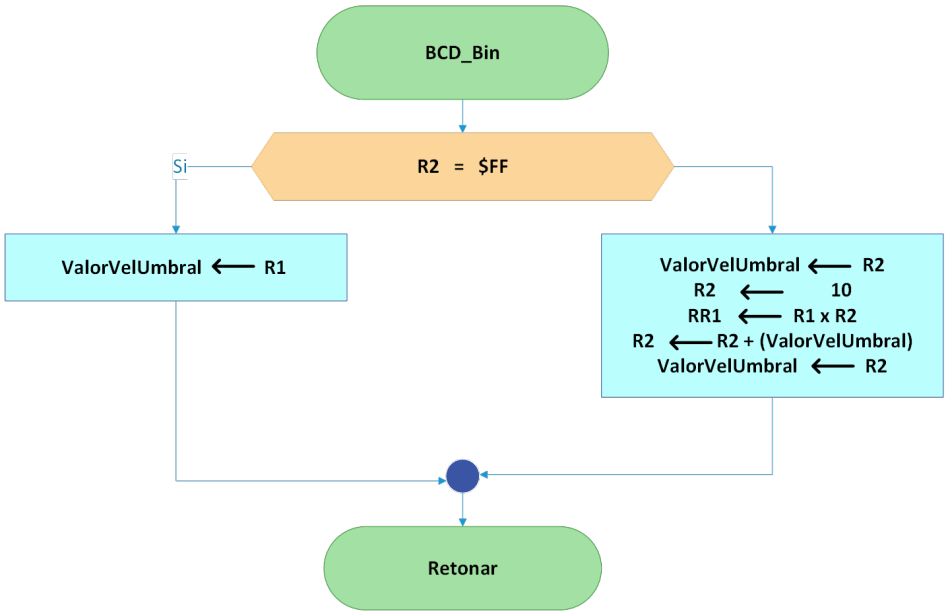

Figura 54: Diagrama de flujo de la subrutina BCD_Bin

4.5. Leer_Tecla

La subrutina `Leer_Tecla` se encarga de detectar cuál tecla ha sido presionada en un teclado matricial 4x3 conectado al puerto A del microcontrolador. El resultado de la lectura se almacena en la variable `Tecla`. Si ninguna tecla está siendo presionada, el valor almacenado será `#$FF`. La subrutina implementa un escaneo columna por columna mediante el uso de un patrón binario de exploración que va activando cada fila una a una. Para cada patrón aplicado al puerto A, se evalúan las líneas de entrada correspondientes a las columnas para verificar si alguna está en estado bajo (lo que indica una tecla presionada).

Una vez detectada una tecla, el desplazamiento acumulado en el registro A es usado como offset para acceder a la tabla `Teclas`, donde se almacena el valor lógico de cada tecla. Este valor se copia en la variable `Tecla`.

Entradas:

- `Teclas`: Tabla con los valores asignados a cada tecla del teclado matricial.

Salidas:

- `Tecla`: Contiene el valor correspondiente a la tecla presionada, según la tabla `Teclas`. Si no se detecta ninguna, se almacena el valor `$FF`.

4.5.1. Estructuras de Datos

Estructuras de Datos	
Variable	Descripción
Patron	Tipo Byte, Patrón para leer el Puerto A.
Tecla	Tipo byte, Variable que contiene la Tecla ingresada.
Teclas	Tabla de los valores asignado a las teclas .

Tabla 23: Estructuras de Datos utilizados en la Subrutina Leer_Tecla

4.5.2. Diagrama de flujo

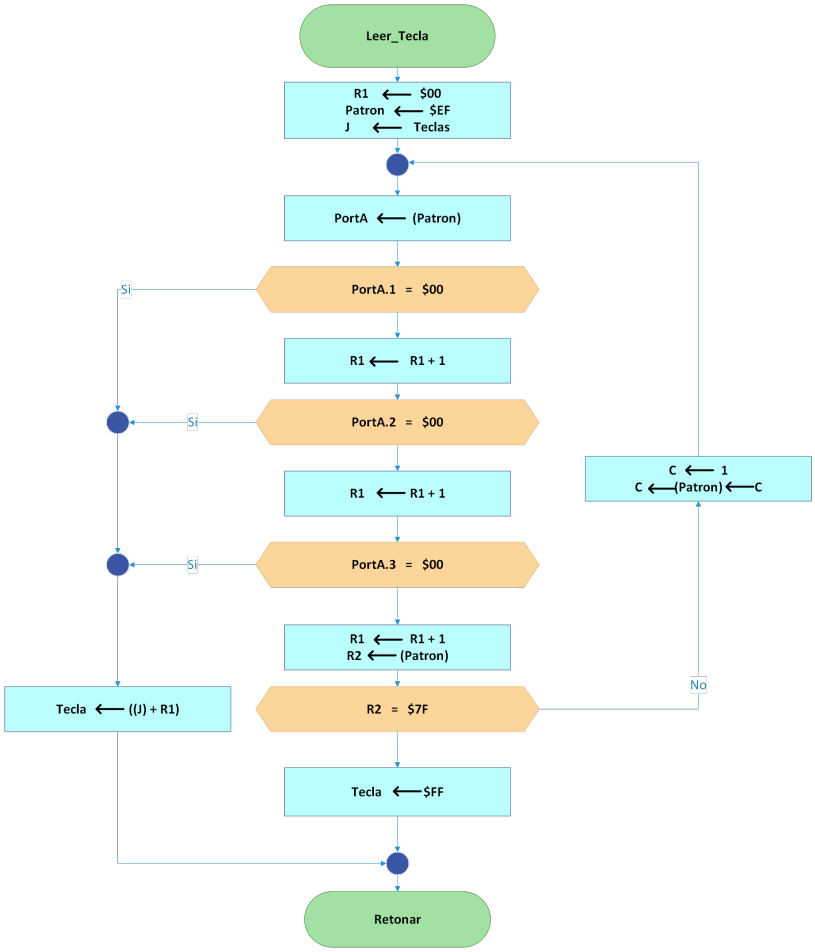

Figura 55: Diagrama de flujo de la subrutina Calcula

4.6. Borrar_Num_Array

La subrutina `Borrar_Num_Array` tiene como función limpiar completamente el arreglo `Num_Array`, utilizado para almacenar las teclas ingresadas por el usuario en operaciones como la configuración de la velocidad umbral. La subrutina recorre secuencialmente el arreglo `Num_Array` y escribe el valor `$FF` en cada una de sus posiciones. Este valor se utiliza como marcador para indicar que la posición está vacía o no contiene un dato válido.

El número de posiciones a limpiar está definido por la constante `MAX_TCL`, que representa la longitud máxima del arreglo. Se emplea un registro como contador decreciente y una instrucción de iteración para realizar el borrado.

Entradas:

- `MAX_TCL`: Tamaño del arreglo `Num_Array`.

Salidas:

- Todas las posiciones del arreglo `Num_Array` quedan con el valor `$FF`.

4.6.1. Estructuras de Datos

Estructuras de Datos	
Variable	Descripción
MAX_TCL	Tipo Byte, Longitud máxima de num_array.
num_array	Arreglo de Teclas ingresadas.

Tabla 24: Estructuras de Datos utilizados en la Subrutina Borrar_Num_Array

4.6.2. Diagrama de flujo

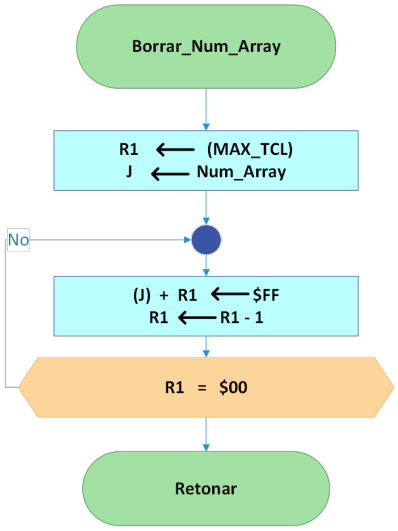

Figura 56: Diagrama de flujo de la subrutina Borrar_Num_Array

5. Subrutina de atención a interrupciones

5.1. Subrutina: Maquina_Tiempos

La subrutina `Maquina_Tiempos` es invocada mediante una interrupción por *Output Compare (OC)* del módulo de temporización del microcontrolador. Su función es administrar de forma centralizada todos los temporizadores de la aplicación, organizados por diferentes bases de tiempo. En cada ejecución:

- Se programa el siguiente evento de comparación temporal sumando el valor `Carga_TC4` al contador `TCNT`.
- Se ejecuta `Decre_Timers_BaseT`, que decrementa los timers de base mínima.
- Se verifican y actualizan las bases mayores.
- Cuando uno de estos alcanza cero, se recarga y se invoca `Decre_Timers` para procesar los timers de esa base específica.

Este mecanismo permite implementar múltiples temporizadores con diferentes resoluciones, todos sincronizados por una única interrupción periódica. Es fundamental para temporizar tareas como el parpadeo de mensajes, control de pulsos, retardos, refresco de pantalla, etc.

5.1.1. Estructuras de datos

Estructuras de Datos	
Variable	Descripción
Timer1mS	Tipo Byte, Timer para la base de 1 mS.
Timer10mS	Tipo Byte, Timer para la base de 10 mS.
Timer100mS	Tipo Byte, Timer para la base de 100 mS
Timer1S	Tipo Byte, Timer para la base de 1 S
Valores y banderas	
Nombre	Descripción
Tabla_Timers_Base_T	Tabla de los Timers con una Base T.
Tabla_Timers_Base1mS	Tabla de los Timers con una Base 1 mS.
Tabla_Timers_Base10mS	Tabla de los Timers con una Base 10 mS.
Tabla_Timers_Base100mS	Tabla de los Timers con una Base 100 mS.
Tabla_Timers_Base1S	Tabla de los Timers con una Base 1 S.
tTimer1mS	Carga para el Timer1mS
tTimer10mS	Carga para el Timer10mS
tTimer100mS	Carga para el Timer100mS
tTimer1S	Carga para el Timer1S

Tabla 25: Estructuras de Datos utilizados en la Subrutina Calcula

5.1.2. Diagrama de flujo

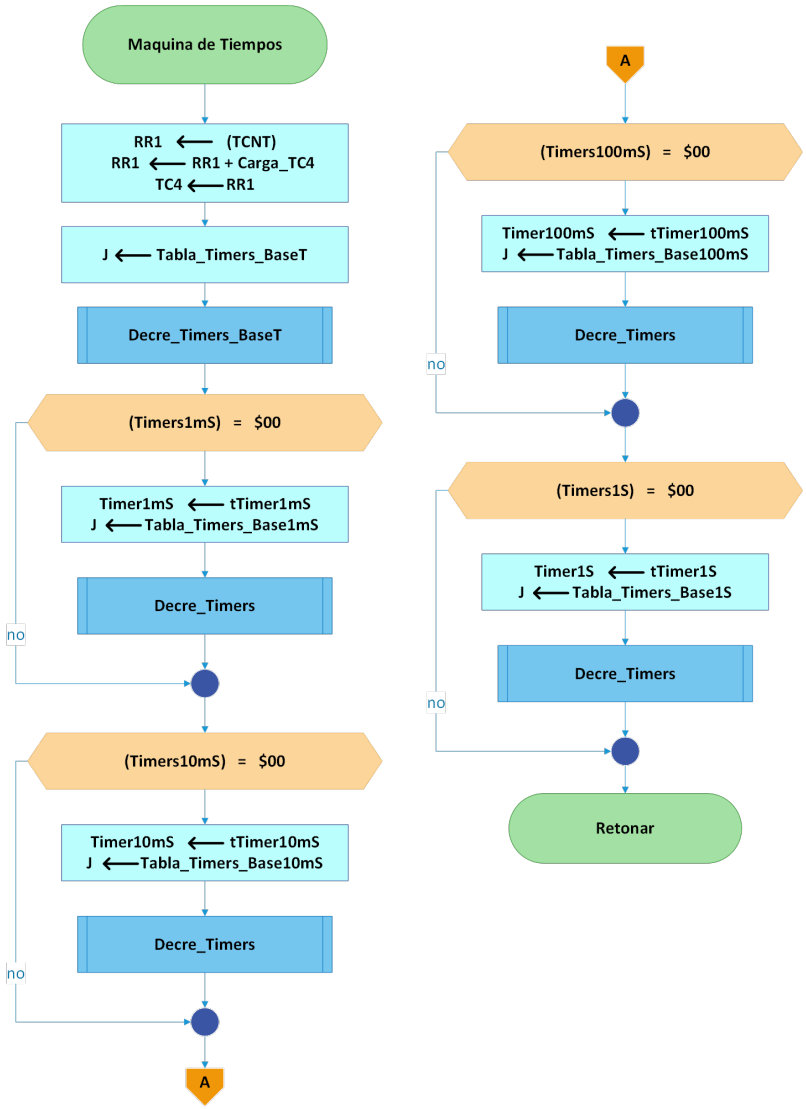

Figura 57: Diagrama de flujo de la subrutina Maquina_Tiempos

5.2. Subrutina: Decre_Timers_BaseT

Decrementa todos los timers contenidos en una tabla apuntada por `Tabla_Timers_BaseT`. Los valores `$0000` son ignorados. Si se encuentra un valor `$FFFF`, se considera fin de la tabla.

La subrutina recorre la tabla de dos en dos bytes. Si el valor leído es distinto de cero y distinto de `$FFFF`, lo decrementa en uno. Esta tabla suele usarse para temporizadores de 16 bits en tareas que requieren mayor resolución o duración.

5.2.1. Diagrama de flujo

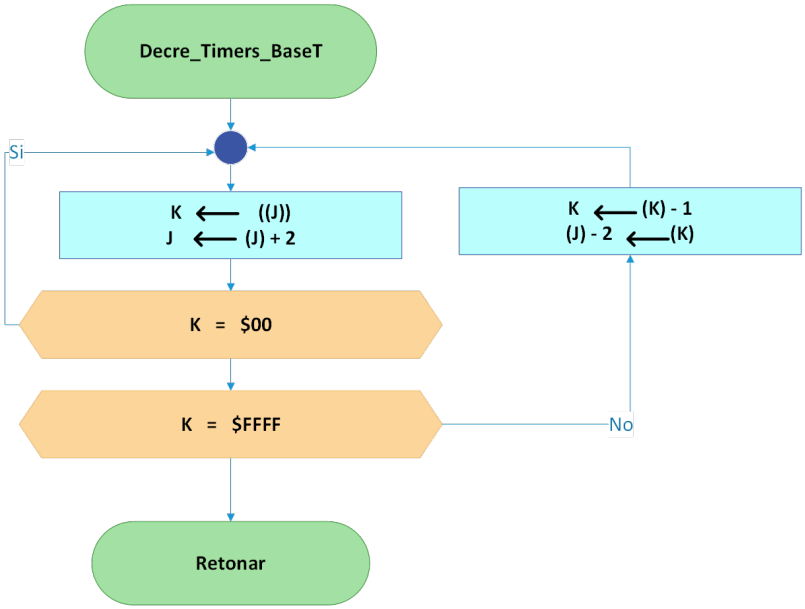

Figura 58: Diagrama de flujo de la subrutina Decre_Timers_BaseT

5.3. Subrutina: Decre_Timers

Decrementa todos los temporizadores ubicados en una tabla de 8 bits, con fin marcado por el valor `$FF`. La subrutina recorre la tabla byte a byte. Si el valor en la posición actual es distinto de `$FF`, lo decrementa. Si es `$00`, simplemente avanza a la siguiente posición.

5.3.1. Diagrama de flujo

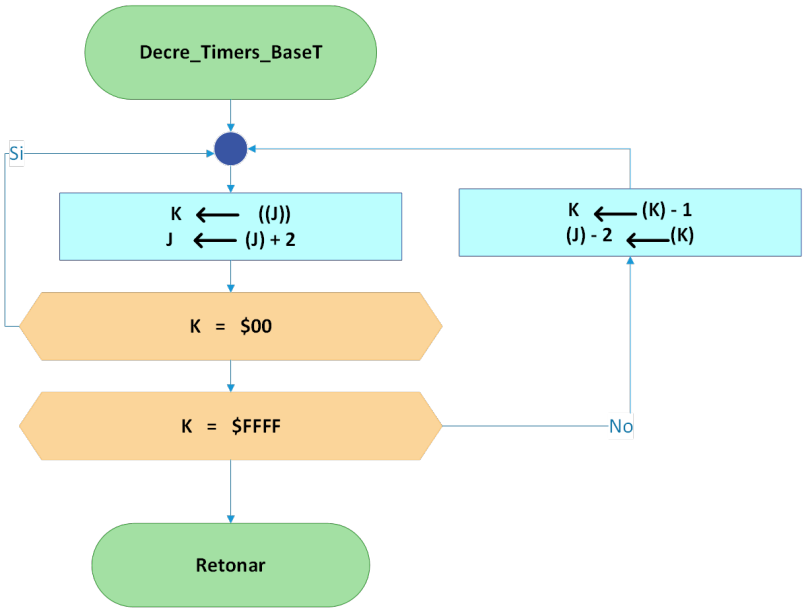

Figura 59: Diagrama de flujo de la subrutina Maquina_Tiempos

6. Conclusiones

El desarrollo del proyecto Separador 623 permitió integrar y aplicar de manera práctica los conocimientos adquiridos a lo largo del curso IE0623 Microprocesadores. A través del diseño modular del sistema, basado en máquinas de estado temporizadas, se logró implementar un dispositivo funcional y eficiente para la clasificación de tuercas según su velocidad lineal. Entre los principales logros del proyecto destacan:

- **Integración efectiva de periféricos:** Se configuraron y utilizaron múltiples periféricos del microcontrolador, incluyendo el convertidor analógico-digital, pantalla LCD, displays de 7 segmentos, teclado matricial, sensores de proximidad y relés, demostrando un manejo completo del entorno de desarrollo.
- **Diseño estructurado con máquinas de estado:** La implementación de tareas como máquinas de estado independientes favoreció la modularidad del código y facilitó el mantenimiento, depuración y escalabilidad del sistema.
- **Gestión precisa del tiempo:** La subrutina de atención a interrupciones y el uso del módulo *Output Compare* permitieron una temporización precisa, esencial para el correcto funcionamiento del sistema, especialmente en el cálculo de velocidad y control del brillo.
- **Interacción intuitiva con el usuario:** El sistema responde adecuadamente a entradas del usuario mediante el teclado, dipswitches y botones, y proporciona información clara mediante la pantalla LCD y los displays, logrando una interfaz eficiente.
- **Robustez y validación de datos:** Se incorporaron mecanismos de validación para garantizar que los valores ingresados se encuentren dentro de rangos válidos, evitando errores durante la operación. Además, el sistema contempla condiciones de error como velocidades fuera de rango o rebases.

La realización del proyecto Separador 623 representó un verdadero reto y, al mismo tiempo, una oportunidad para poner en práctica los conocimientos adquiridos durante el curso de Microprocesadores. A lo largo del desarrollo del sistema.

Uno de los principales aprendizajes fue comprender la importancia del control del tiempo dentro del sistema. Implementar tareas que dependen de interrupciones precisas y sincronización entre periféricos me obligó a ser más riguroso y ordenado en mi forma de programar.

Además, trabajar con máquinas de estado ayudó a ver la programación desde una perspectiva estructurada. Dividir el sistema en tareas independientes y coordinarlas mediante un despachador facilitó tanto el diseño como la depuración. Fue interesante notar cómo cada pequeño bloque del sistema (ya fuera una rutina de conversión, una lectura de sensor o un mensaje en la pantalla LCD) juega un papel fundamental en el funcionamiento general.

7. Recomendaciones

Durante el desarrollo del proyecto **Separador 623** se enfrentaron varios desafíos técnicos y de diseño que representaron oportunidades de aprendizaje. Uno de ellos fue la comprensión clara de las máquinas de estado antes de programar,una de las principales dificultades fue organizar correctamente la lógica de cada tarea mediante máquinas de estado. Al principio, puede ser confuso manejar

múltiples tareas con diferentes transiciones. Por eso ayuda antes de codificar, definir cada máquina de estado en papel (diagramas de flujo) y validar su lógica con pruebas unitarias a cada tarea, esto ayuda a evitar errores difíciles de depurar posteriormente e integrarlos al sistema de una manera mas sencilla.