Universidad de Costa Rica

IE-0623

Microprocesadores

Trabajo Final

Grupo 01

Profesor: Geovanny Delgado M.Sc.E.E

Estudiante:
Eduardo Alfaro González

Carné:
B50203

9 de Diciembre de 2019

1. Resumen

El siguiente trabajo incluye el código para la tarjeta dragon 12 de NXP, el cual corresponde a un sensor de velocidad para vehículos, ensamblado por medio de la herramienta AsmIDE.

El sensor cuenta con 3 modos de operación, el primero, MODO Config es utilizado para configurar la velocidad limite de la zona donde se realiza la medición, esto se hace por medio del uso del teclado matricial de la tarjeta y las pantallas de 7 segmentos para mostrar la velocidad configurada. Es necesario notar que por especificaciones del proyecto la velocidad límite solo puede estár en el rango de 45 km/h a 90 km/h.

El segundo modo, MODO Libre, corresponde a un modo ocioso donde el sensor no realiza ninguna medición y tampoco se puede modificar ningún parámetro de memoria. Durante la ejecución de este modo las pantallas de 7 segmentos permanecen apagas.

Finalmente, el último modo, MODO Medición corresponde a al calculo de velocidades, la cual se realiza por medio de la simulación de dos sensores de proximidad por medio de los botones del puerto H. Una vez que se determina la velocidad se muestra en pantalla si es un valor valido y si se pasa de la velocidad limite configurada se enciende una alarma representada en los LEDs.

Cada modo cuenta con sus respectivos mensajes en la pantalla LCD de la tarjeta, además de indicar en que modo de operación se encuentra el sensor indican que significa el contenido de las pantallas de 7 segmentos.

Como resultados se obtiene la operación deseada para el sensor de velocidad por medio del uso de 22 subrutinas, incluyendo el programa principal, esto siguiendo el diseño de los requisitos del programa, realizando ciertas modificaciones para optimizar el código y la reducción de cálculos.

El código completo ensamblado tiene un peso de 1986 bytes. Todo el programa se almacena en la sección de memoria EEPROM.

Índice

Índice de figuras

2. Estructuras de datos

En la tabla 1 se muestran las estructuras de datos utilizadas para la realización del proyecto.

3. Diseño

3.1. Programa principal

Debido a que el microprocesador no tiene un sistema operativo se requiere un programa que se ejecute constantemente y permita el llamado a las otras subrutinas del resto del código. Este programa principal incluye la configuración de hardware de la tarjeta y la inicialización de las variables de memoria.

3.1.1. Configuración de hardware

Para poder utilizar los periféricos de la tarjeta es necesario configurar el hardware, esto se realiza por medio de la escritura en los registros de control de los correspondientes a cada periférico. En la figura 1 se puede apreciar el diagrama de flujos utilizado para la configuración e inicialización de variables. Información obtenida de [2].

Tabla 1: Estructuras de datos utilizadas a lo largo del proyecto

Posición	Etiqueta	Tamaño	Contenido
$1000	BANDERAS	2	15: MOD_H 14:MOD_L 11:Enable tick_vel 10:PRINT Calculando... 9:Data or Control LCD 8:Cambio_Modo 5:CALC_TICK 4:ALERTA 3:PANT_FLAG 2:ARRAY_OK 1:TCL_Leida 0: TCL_Lista
$1002	V_LIM	1	Velocidad limite
$1003	MAX_TCL	1	Maximo numero de teclas que se aceptan del teclado matricial, en este caso 2
$1004	TECLA	1	Tecla ingresada
$1005	TECLA_IN	1	Tecla anterior
$1006	CONT_REB	1	Contador de rebotes, utilizado para suprimirlos
$1007	CONT_TCL	1	Contador de teclas almacendas en memoria
$1008	PATRON	1	Patron a enviar para leer las teclas del teclado
$1009	NUM_ARRAY	2	Arreglo de números ingresados por medio del teclado
$100b	BRILLO	1	0-100 cotrola el brillo de 7 seg
$100c	POT	1	Variable que almacena el valor promedio del ATD
$100d	TICK_EN	2	Cantidad de ticks necesarios para cubrir 100m
$100f	TICK_DIS	2	Cantidad de ticks necesarios para cubrir 200m
$1011	VELOC	1	VELOCIDAD MEDIDA POR LOS SENSORES
$1012	TICK_VEL	1	Ticks utilizados para sensar la velocidad del vehiculo
$1013	BIN1	1	Corresponde al valor de DISP1 y DISP2 en binario
$1014	BIN2	1	Corresponde al valor de DISP4 y DISP3 en binario
$1015	BCD1	1	bin1 en bcd
$1016	BCD2	1	bin2 en bcd
$1017	BCD_L	1	Variable temporal para la conversión binaria a BCD
$1018	LOW	1	Variable temporal para la conversión binaria a BCD
$1019	DISP1	1	Valor a desplegar en el display derecho
$101a	DISP2	1	Valor a desplegar en el display centro-derecho
$101b	DISP3	1	Valor a desplegar en el display centro-izquierdo
$101c	DISP4	1	Valor a desplegar en el display izquierdo
$101d	LEDS	1	Valor a desplegar en los LEDs del puerto B
$101e	CONT_DIG	1	Digito actual de 7seg
$101f	CONT_TICKS	1	Contado de Ticks para el control de los displays
$1020	DT	1	100 - BRILLO, valor donde se resetea CONT_TICKS
$1021	CONT_7SEG	2	Contador para llamar la subrutina BCD_7seg, se reinicia en 5000
$1023	CONT_200	2	Contador para 200 ms para habilitar el atd
$1025	CONT_DELAY	1	Contador para generar retardos
$1026	Dm2S	1	Constante para generar un delay de 2ms
$1027	D40uS	1	Constante para generar un delay de 60 ms
$1028	D60uS	1	Constante para generar un delay de 60 ms
$1029	Clear_LCD	1	Constante igual a comando clear
$102a	ADD_L	1	Constante igual a Adress linea 1 lcd
$102b	ADD_L	1	Constante igual a Adress linea 2 lcd
$102c	TEMP	1	Variable temportal utilizada en BIN_BCD
$102d	LEDS37	1	Variable que define el desplazamiento de los le 37 en modo alerta
$1030	TECLAS	12	Tabla que contiene el valor de cada tecla
$1040	SEGMENT	12	Tabla para la conversión BCD a 7 segmentos
$1050	iniDsp	5	Tabla de comandos para iniciar la comunicación con la pantalla LCD
$1060	MESS1	13	MODO CONFIG
–	MESS2	14	VELOC. LIMITE
–	MESS3	13	RADAR 623
–	MESS4	13	MODO LIBRE
–	MESS5	14	MODO MEDICION
–	MESS6	15	SU VEL. VEL.LIM
–	MESS7	14	ESPERANDO...
–	MESS8	15	CALCULANDO...

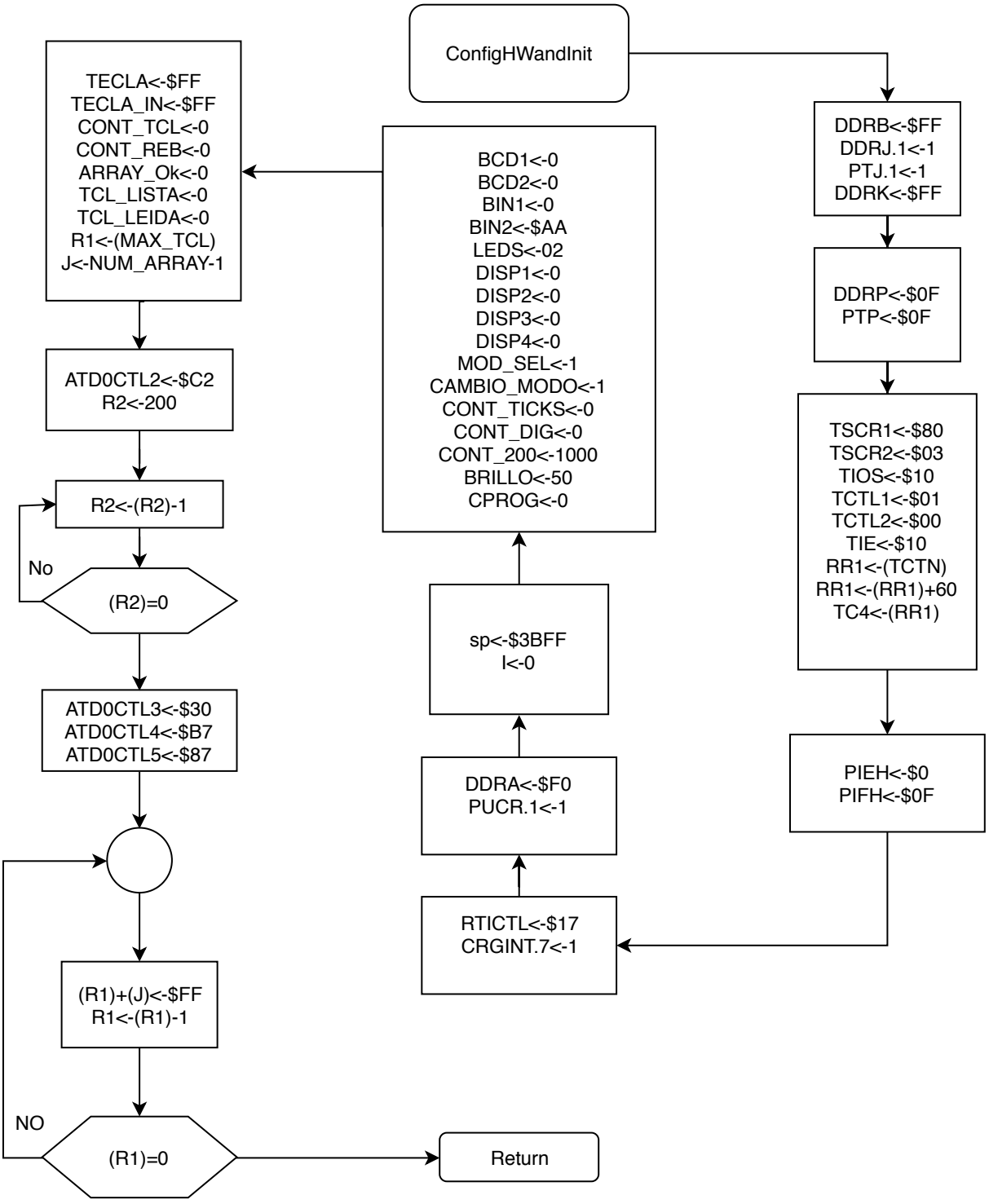

Figura 1: Diagrama de la configuración de hardware e inicialización de variables.

A continuación se presenta la explicación del contenido de cada uno de los registros de control, según la información expuesta en [3].

LEDs, pantalla de 7 segmentos y pantalla LCD

- DDRB: Registro que permite establecer si el puerto B es un puerto de escritura o lectura, en este caso todos los puertos son de escritura por lo que se escribe un 1 ($FF)

- DDRJ: Bit 1 del puerto J debe permitir escritura por lo que se escribe un 1 en el bit correspondiente.
- PTJ: Se escribe un 1 en el bit 0 para habilitar los LEDs inicialmente.
- DDRP: Habilita la escritura en el puerto P, el cuál controla cuál pantalla de 7 segmentos está encendida.
- PTP: Apaga todas las pantallas de 7 segmentos.
- DDRK: Se habilita la escritura en todos los bits del puerto K, esto para la comunicación con la pantalla LCD.

OC4 y TCNT

- TSCR1: Se escribe un 1 en el bit 7 para habilitar el módulo de timers.
- TSCR2: Se escribe un 3 en los primeros 3 bits debido a que se debe utilizar un preescalador de 8 (2^3), adicionalmente inicialmente la interrupción por rebase del contador está deshabilitada.
- TIOS: Se escribe un 1 en el bit 4 para habilitar la salida del output compare 4.
- TCTL1: Se escribe un 01 en los primeros dos bits para configurar la salida del puerto 4 como un toggle.
- TIE: Se habilita la interrupción del puerto 4 por medio del bit 4.

Adicionalmente se debe calcular el valor que se debe añadir al contador para generar una interrupción a 50 kHz por medio del output compare 4.

$$TOC = \frac{TCS \cdot PRS}{BUS\_CLK} \tag{1}$$

$$TC4 = \frac{24MHz}{8 \cdot 50kHz} = 60 \tag{2}$$

Puerto H

- PIEH: Inicialmente se deshabilitan las interrupciones del puerto H, por lo tanto se escribe un 0.
- PIFH: Se borran las banderas de interrupción del puerto H.

RTI Para la interrupción rti se opta por utilizar un periodo de interrupciones de 1 ms esto se logra con un valor de M de 1 y valor N de 7.

- CRGINT: Se escribe un 1 en el bit 7 para borrar la bandera de interrupción.
- RTICTL: Se escribe N en los primeros 4 bits y M en los siguientes 3.

Puerta A (Teclado matricial)

- DDRA: Se habilita la escritura en los 4 bits más significativos del puerto A.
- PUCR: Se habilita la resistencia del Pull-up del puerto A por medio de la escritura de un 1 en el bit 1.

Convertidor analógico digital El convertidor analógico digital se debe encender primero, esperar a que se encienda completamente y luego se realiza la escritura de sus registros de control, es por esto que se carga un valor de 200 en el registro R2 y se continua hasta que este llegue a 0, con esto se cumple el tiempo de espera especificado por el fabricante de $10\mu s$

- ATD0CTL2: Se habilita por medio del bit 7, además se habilita la bandera de borrado rápido con la lectura de los registros de datos por medio del bit 6 y finalmente se habilitan las interrupciones por medio del bit 1.
- ATD0CTL3: En los bits del 3 al 6 se escribe un 6 para realizar 6 mediciones antes de generar una interrupción.
- ATD0CTL4: Los datos generados deben ser de 8 bits por lo que se escribe un 1 en el bit 7, adicionalmente se utilizan 4 ciclos del ATD para las conversiones y se utiliza un preescalador de 17 en los bits del 0 al 4.
- ATD0CTL5: Se utilizan conversiones con signo por lo que se escribe un 1 en el bit 7, además se utiliza la entrada 7 que corresponde al potenciómetro.

3.1.2. Inicialización de variables

Debido a que en el programa existen variables que se leen antes de que se escriban por primera, es necesario iniciarlas en un valor predeterminado. En el diagrama de la figura 1 se puede apreciar la inicialización de las variables que requieren de un valor por defecto.

3.1.3. Rutina Main

Es el programa principal del código es el encargado de detectar por medio de los dipswitch cual modo ha sido seleccionado y mostrar en pantalla el mensaje correspondiente.

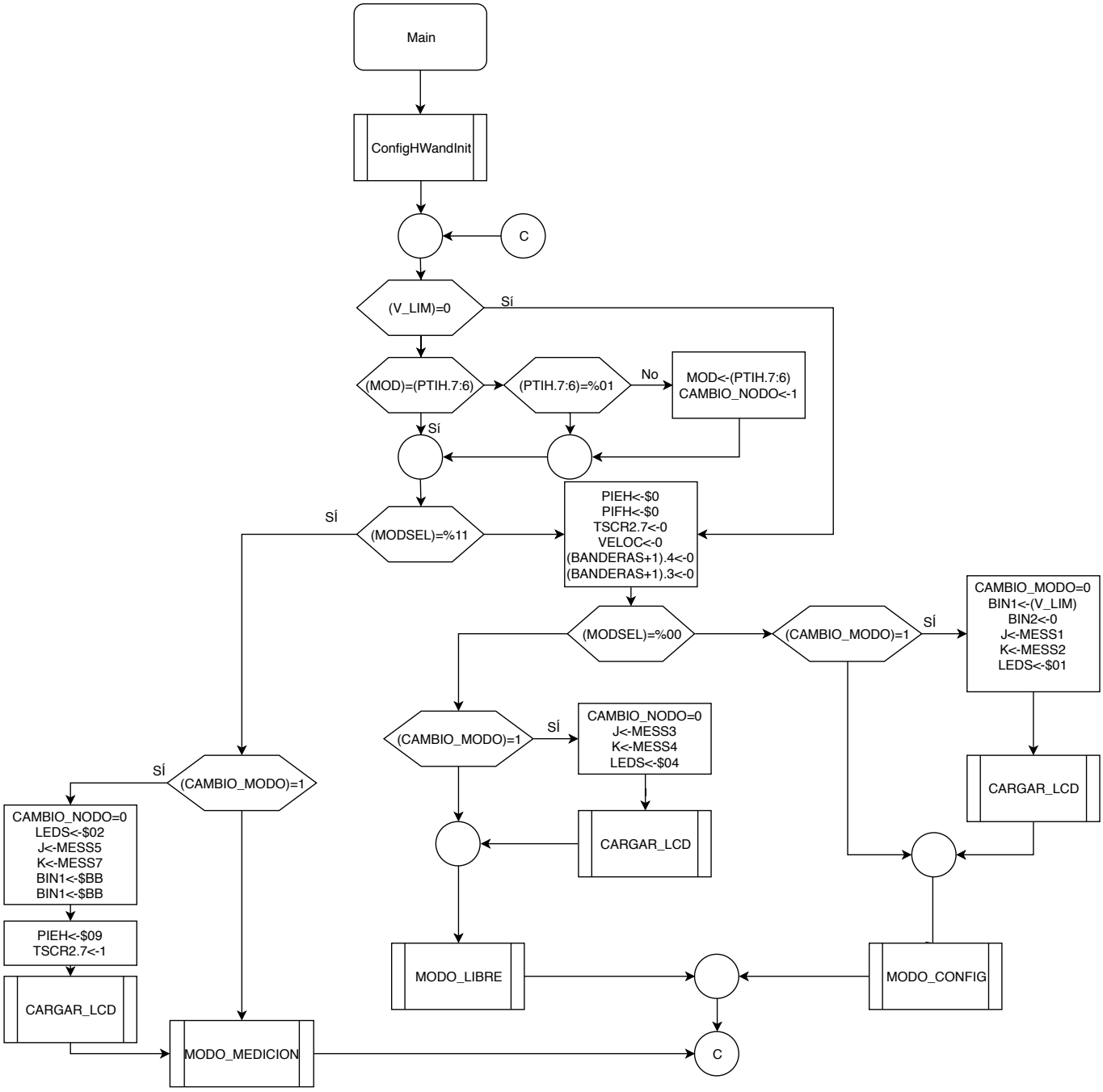

Figura 2: Rutina de programa principal del sensor de velocidad

En la figura 2 se puede apreciar el diagrama de flujos utilizado para el diseño del programa principal, este primero debe pasar por la subrutina de configuración de hardware para asegurar que los periféricos van a operar de la forma deseada, seguidamente se debe verificar que la velocidad límite sea diferente de 0 ya que no se puede operar el modo medición correctamente sin un valor para la misma.

Si la velocidad es distinta de 0 se lee el valor de los dipswitch y se verifica si es igual al valor guardado en memoria, si es distinto se levanta una bandera que indica que se debe cambiar el contenido de la pantalla LCD. Una vez que se realiza esta operación se selecciona el modo de operación correspondiente y se ejecuta la subrutina, una vez que esta finaliza se retorna a la comparación del

valor de la velocidad límite con 0 y la lectura de los dipswitch.

Paso de parámetros

- **Entrada**
  - V_LIM: Velocidad limite, si es 0 no puede cambiar de modo
  - MESS1, MESS2, MESS3, MESS4, MESS5, MESS7: Mensajes que se muestran en la pantalla LCD
- **Salida**
  - LEDs: Se enciende dependiendo del modo
  - BIN1: Dependiendo del modo se configura a $BB
  - BIN2: Dependiendo del modo se configura a $BB

3.2. Subrutinas de atención a interrupciones

3.2.1. Puerto H

Al puerto H se encuentran conectados los 4 botones de la tarjeta que permiten la atención por interrupciones, para este programa se utilizan para simular los sensores de velocidad, para el sensor 1 se utiliza el botón 3 y para el sensor 2 se usa el botón 0.

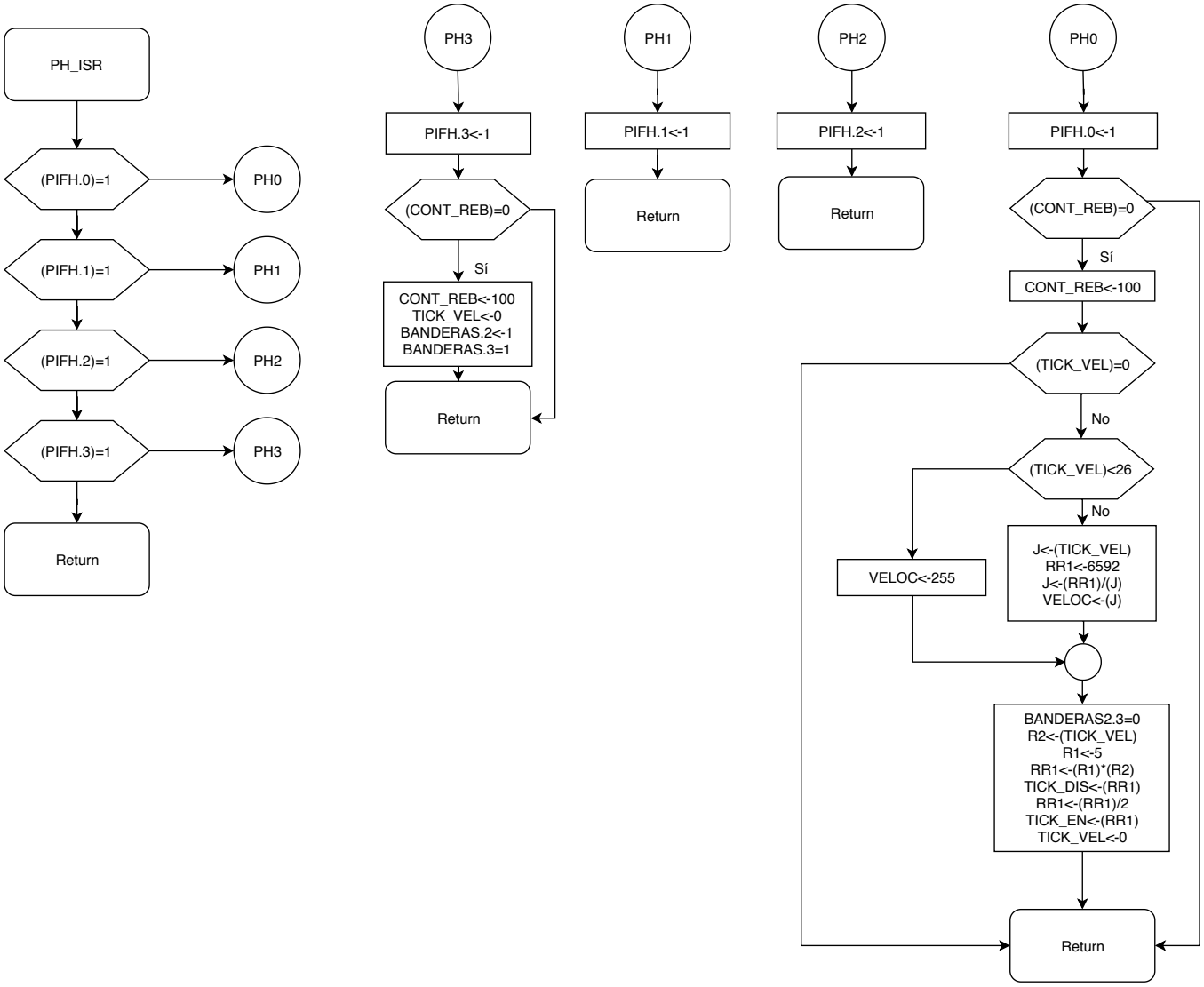

Figura 3: Subrutinas utilizadas para los botones del puerto H

3.2.2. PH3

Como se observa en la figura 3 el sensor 1 únicamente se encarga del inicio del conteo de ticks, para esto debe verificar que el contador de rebotes sea 0 para evitar lecturas asociadas al rebote físico del botón. Adicionalmente se levanta la bandera que indica al modo medición que se debe mostrar en pantalla el mensaje Çalculandoz la bandera que habilita el conteo de ticks.

Paso de parámetros

- **Entrada**
  - CONT_REB: Contador de rebotes, debe estar en 0 para que se pueda ejecutar el código.
- **Salida**
  - TICK_VEL: se borran en esta interrupción.

3.2.3. PH0

Simula el segundo sensor, aparte de eso incluye la subrutina calculo, es decir a partir de la cantidad de Ticks calcula la velocidad del vehículo así como los ticks necesarios para mostrar la velocidad y los necesarios para borrarla de pantalla. También debe borrar la bandera que habilita el conteo de ticks en la subrutina TCNT.

Para el calculo de la velocidad se usan las siguientes ecuaciones.

Como se menciono en la configuración del hardware para el modulo de tiempos se utiliza un pre escalador de 8 por lo tanto se tiene un tiempo entre ticks de:

$$T_{TICK} = \frac{8 \cdot 2^{16}}{24Mhz} = 0{,}02184533333 \tag{3}$$

Nota: si el tiempo entre sensores se pasa de 5.57056 s la medida se vuelve invalida debido a un rebase de TICK_VEL.

$$Velocidad = \frac{40m}{TICK\_VEL \cdot T_{TICK}} \cdot \frac{3600}{1000} = \frac{6591{,}796875}{TICK\_VEL} = \frac{421875}{64 * TICK\_VEL} \tag{4}$$

Debido a que no se requiere demasiada precisión y para mayor facilidad se utiliza la siguiente formula:

$$VELOC = \frac{40m}{TICK\_VEL \cdot T_{TICK}} \cdot \frac{3600}{1000} \approx \frac{6592}{TICK\_VEL} \tag{5}$$

Es necesario notar que se realizaron modificaciones a esta subrutina con respecto a las especificaciones establecidas en [4]. Inicialmente el cálculo del valor de TICK_EN y TICK_DIS se debía realizar en la subrutina PANT_CTRL. Los cambios se realizaron debido a que se puede utilizar el valor de TICK_VEL que corresponde a la cantidad de Ticks que el vehículo tarda en recorrer 40 m para calcular dichos valores como se muestra a seguidamente por medio de una regla de 3.

Inicialmente se calcula la cantidad de ticks para 100 m y se almacena en $TICK\_EN$

$$TICK\_EN = \frac{TICK\_VEL \cdot 100}{40} = \frac{TICK\_VEL \cdot 5}{2} \tag{6}$$

Luego esta variable se multiplica por dos y se almacena en $TICK\_DIS$

$$TICK\_DIS = TICK\_EN \cdot 2 \tag{7}$$

Para obtener 2 segundos con el contador de ticks se utiliza la siguiente formula:

$$TICK\_EN = \frac{2s}{0{,}02184533333} = 91{,}553 \approx 92 \tag{8}$$

Paso de parámetros

- **Entrada**
  - TICK_VEL: ticks que tarda el vehiculo en recorrer 40 m
  - CONT_REB: contador de rebotes
- **Salida**
  - VELOC: Velocidad del vehiculo
  - TICK_EN: Ticks para habilitar el mensaje en pantalla y velocidad en 7 seg
  - TICK_DIS: Ticks para deshabilitar el mensaje en pantalla y borrar velocidad en 7 seg

3.2.4. RTI

Subrutina encargada del manejo de los rebotes de los botones.

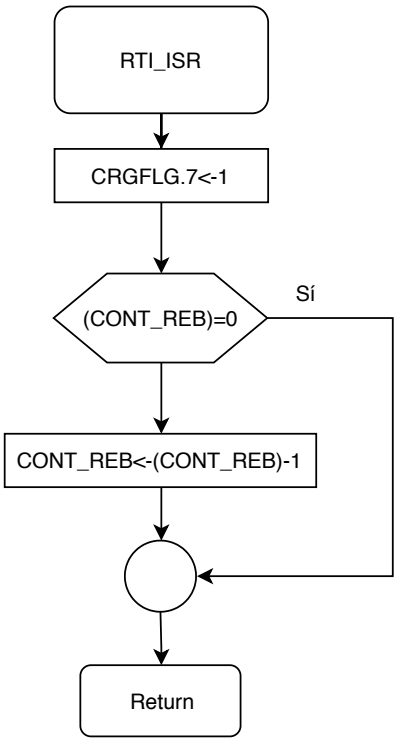

Figura 4: Diagrama de flujos de la subrutina de inteurrpción RTI

En la figura 4 se observa que solo es necesario verificar que el contador de rebotes no ha llegado a 0.

Paso de parámetros

- **Entrada**
  - CONT_REB: se debe verificar que no sea 0.
- **Salida**
  - CONT_REB: si no es cero se decrementa

3.3. Output Compare 4

Subrutina encargada del manejo de la pantalla de 7 segmentos, maneja el contenido de los displays y como se multiplexan, además cada cierto tiempo debe llamar al convertidor analógico digital y el control de los LEDs en el modo alarma, también llama a la subrutina bcd 7 segmentos. En la figura 5 se muestra el diagrama de flujos para el diseño a utilizar.

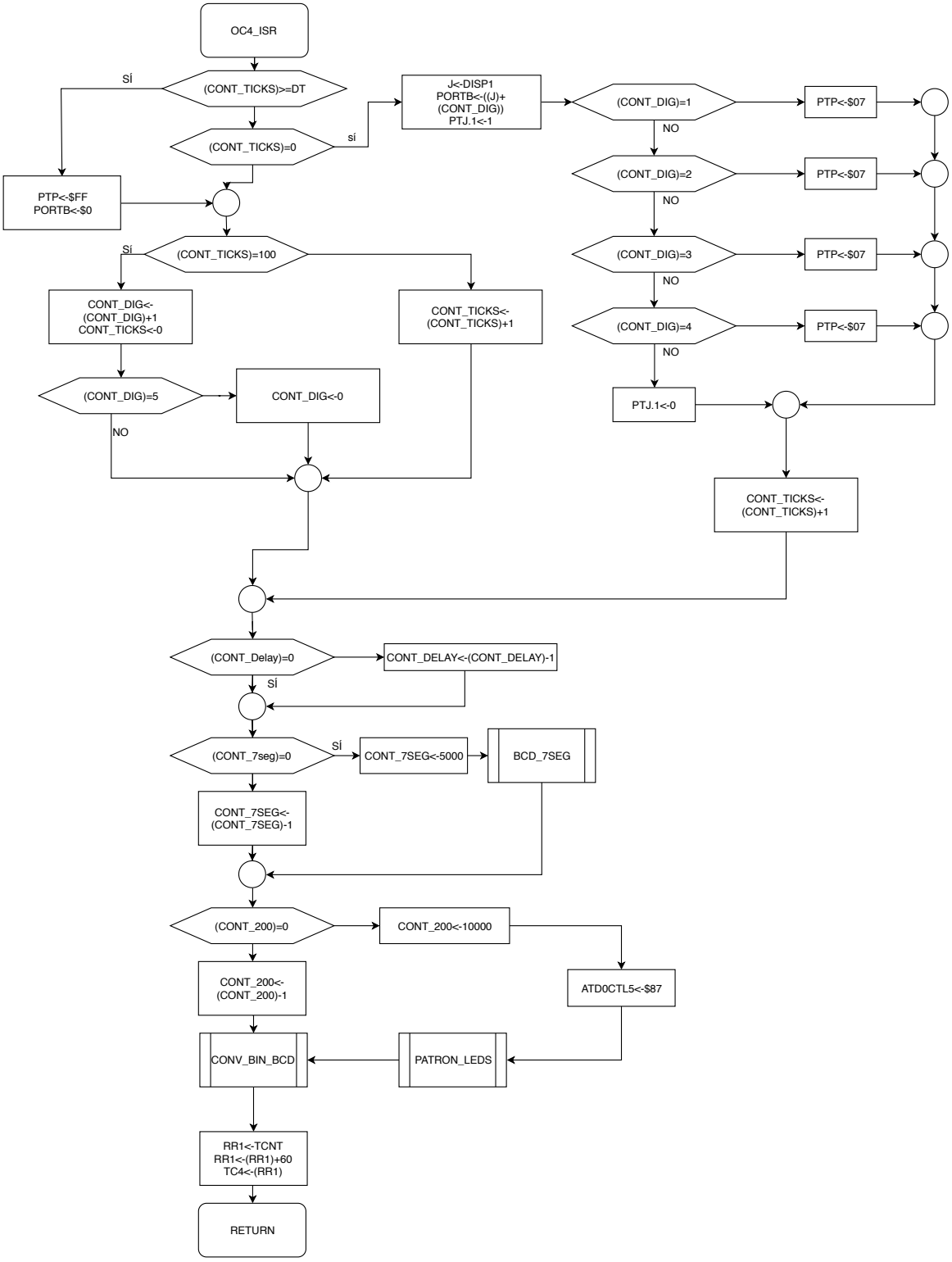

Figura 5: Diagrama de flujos de la subrutina OC4

La subrutina OC4 consta de 3 partes principales, la primera es la encargada del manejo de los ticks, cuando los ticks llegan al valor de la variable DT la pantalla se debe apagar. Cuando el valor de ticks es 0 se debe encender una pantalla de 7 segmentos o los LEDs. Finalmente se compara con el valor de 100, si se llega a este valor se debe reiniciar el contador en 0 y cambiar el dígito con el

que se está trabajando, si el valor es distinto de 100 se debe incrementar el contador.

La segunda parte corresponde al manejo de los displays de 7 segmentos, estos se controlan por la variable Cont_dig la cual indica sobre cual dígito se está trabajando, para manejar la multiplexación de las pantallas o la selección de los LEDs. Finalmente si el contador de dígitos no corresponde a ningún display se encienden los LEDs conectados al puerto B.

Finalmente la última sección se encarga de manejar otras subrutinas que se ejecutan periódicamente, la primera es el decremento del contador de retardo de la pantalla LCD, este se decrementa si el valor es distinto de 0. La segunda es la subrutina BCD_7Seg, la cual debe ser llamada cada 0.1 s. La tercera es la subrutina del convertidor analógico digital la cual se ejecuta cada 0,2 s al igual que la subrutina Patron LEDs.

Paso de parámetros

- **Entrada**
  - DT: determina si se debe apagar la pantalla actual determinado por atd
  - DISP1,DISP2,DISP3,DISP4: Contenido que se debe mostrar en los
  - LEDs:Variable que determina el patron de los LEDs
- **Salida**
  - CONT_DELAY: contador de retraso para contador de pantalla de 7 segmentos

3.3.1. TCNT

Interrupción causada por el overflow del contador de tiempo, esta encargada de incrementar el valor de los ticks que se usan para medir la velocidad del vehículo así como el decremento de las variables que determinan el tiempo que se muestran las velocidades en la pantalla.

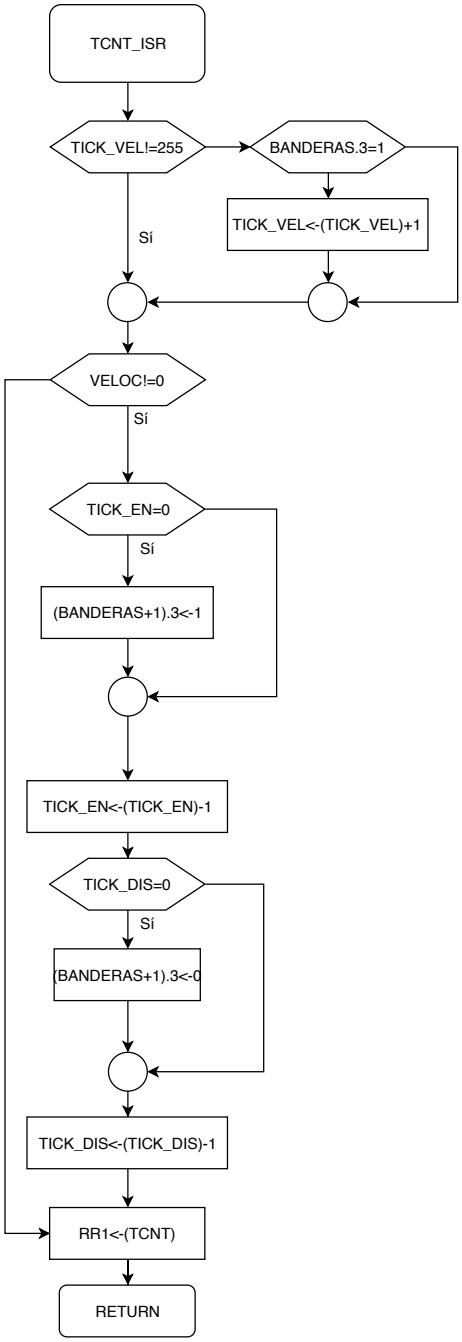

Figura 6: Subrutina a ejecutar durante la interrupción TCNT

En la figura 6 se observa el diagrama de flujos correspondiente, como se puede notar antes de incrementar o decrementar los valores de ticks en memoria se deben verificar ciertas condiciones. Para incrementar los ticks de velocidad es necesario revisar que el valor no sobrepase el valor máximo ya que de lo contrario el valor pasaria a ser incorrecto, y luego se debe verificar que el conteo se haya habilitado a en la interrupción PH3. Para decrementar los Ticks correspondientes al control de las pantallas se debe verificar que el valor de la velocidad sea distinto de 0 es decir que el vehículo haya pasado el segundo sensor.

Paso de parámetros

- **Salida**

- TICK_VEL: Ticks para medir la velocidad, solo si incrementan cuando se encuentra entre los dos sensores
- TICK_EN: Ticks necesarios para mostrar la velocidad el medidor, se decrementan
- TICK_DIS: Ticks necesarios para eliminar la velocidad el medidor, se decrementan

3.3.2. ATD

Subrutina utilizada para la conversión analógica digital del potenciómetro de la tarjeta dragón 12, utilizado para controlar el brillo de los LEDs y las pantallas de 7 segmentos. Se toman 6 mediciones y se calcula el promedio. Adicionalmente el valor de brillo se debe ajustar a una escala de 100. El diagrama de flujos se puede apreciar en la figura 7

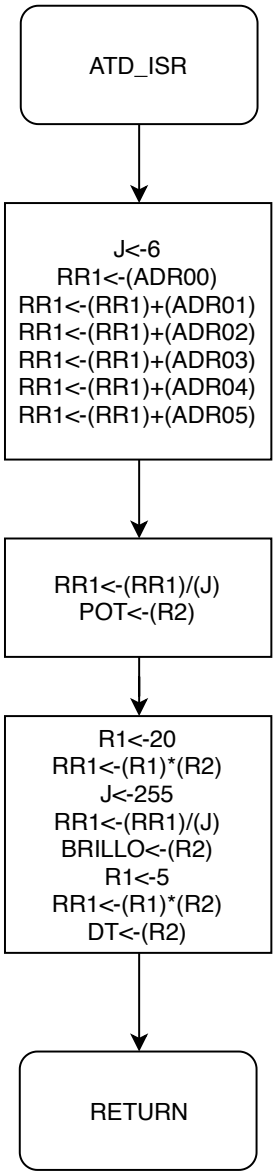

Figura 7: Diagrama de flujos de la subrutina encargada de la conversión analógica digital

Paso de parámetros

- **Entrada**

- ADR00H,ADR01H,ADR02H,ADR03H,ADR04H,ADR05H: Registros de datos del convertidor analógico digital.

- **Salida**
  - Brillo: variable correspondiente a k en el ciclo de trabajo del control de los LEDs.
  - DT: Duty time, variable que determina cuanto tiempo deben permanecer los LEDs. encendidos.

3.4. Subrutinas de uso general

3.4.1. Tarea_Teclado

Subrutina encargada de manejar el teclado matricial de la tarjeta, suprime rebotes.

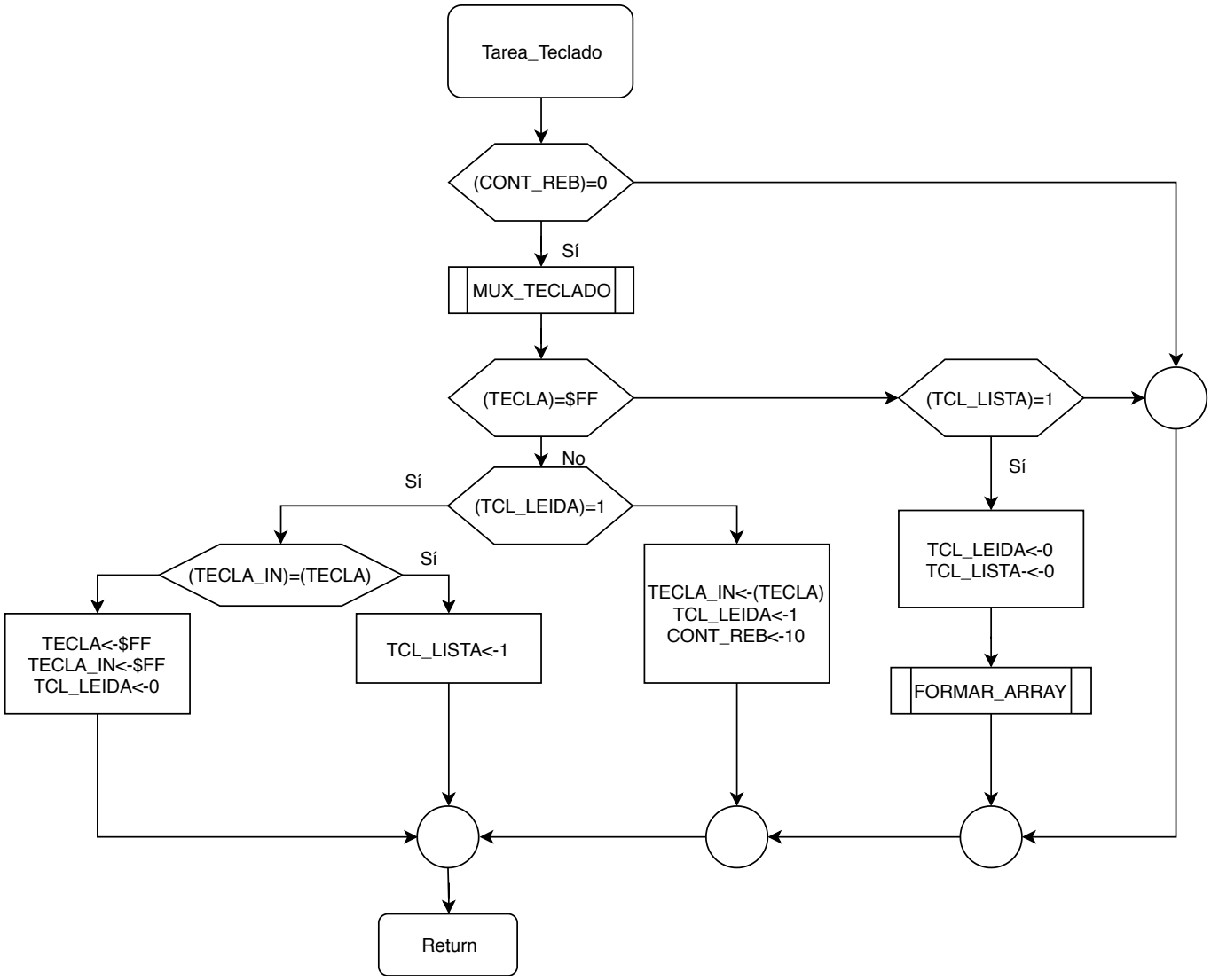

Figura 8: Diagrama de la subrutina tarea teclado

En la figura 8 se encuentra el diagrama de flujos de esta subrutina, inicialmente se debe verificar que el contador de rebotes sea igual a cero para evitar lecturas invalidas del teclado matricial, seguidamente se debe llamar a la subrutina MUX_TECLADO la cual lee la tecla del teclado.

Luego, es necesario realizar ciertas comparaciones para determinar si la tecla leída es válida. La primera, corresponde a verificar si el valor retornado por la subrutina MUX_TECLADO es $FF, si es distinto es porque una tecla ha sido presionada, si es igual se pueden dar dos casos, el primero corresponde a que no se ha presionado ninguna tecla recientemente, el segundo caso se da cuando se acaba de liberar la tecla, en este caso es necesario almacenar el valor en memoria. Si hay una tecla presionada se verifica que la tecla haya sido presionada por un tiempo mínimo para que la lectura sea valida, esto se hace por medio de la comparación con un valor anterior almacenado en memoria en TECLA_IN y con la ayuda de las bandera TECLA_LEIDA.

Paso de parámetros

- **Entrada**
  - TECLA: Tecla presionada en el teclado.
  - TECLA_IN: Tecla presionada antes de suprimir los rebotes, se debe verificar que sea igual a TECLA.

3.4.2. MUX_TECLADO

Subrutina encargada de capturar el valor presionado en el teclado matricial, para esto se envía un patrón al puerto A, y se detecta la señal de entrada correspondiente, para esto se tiene una tabla de valores correspondientes a cada tecla.

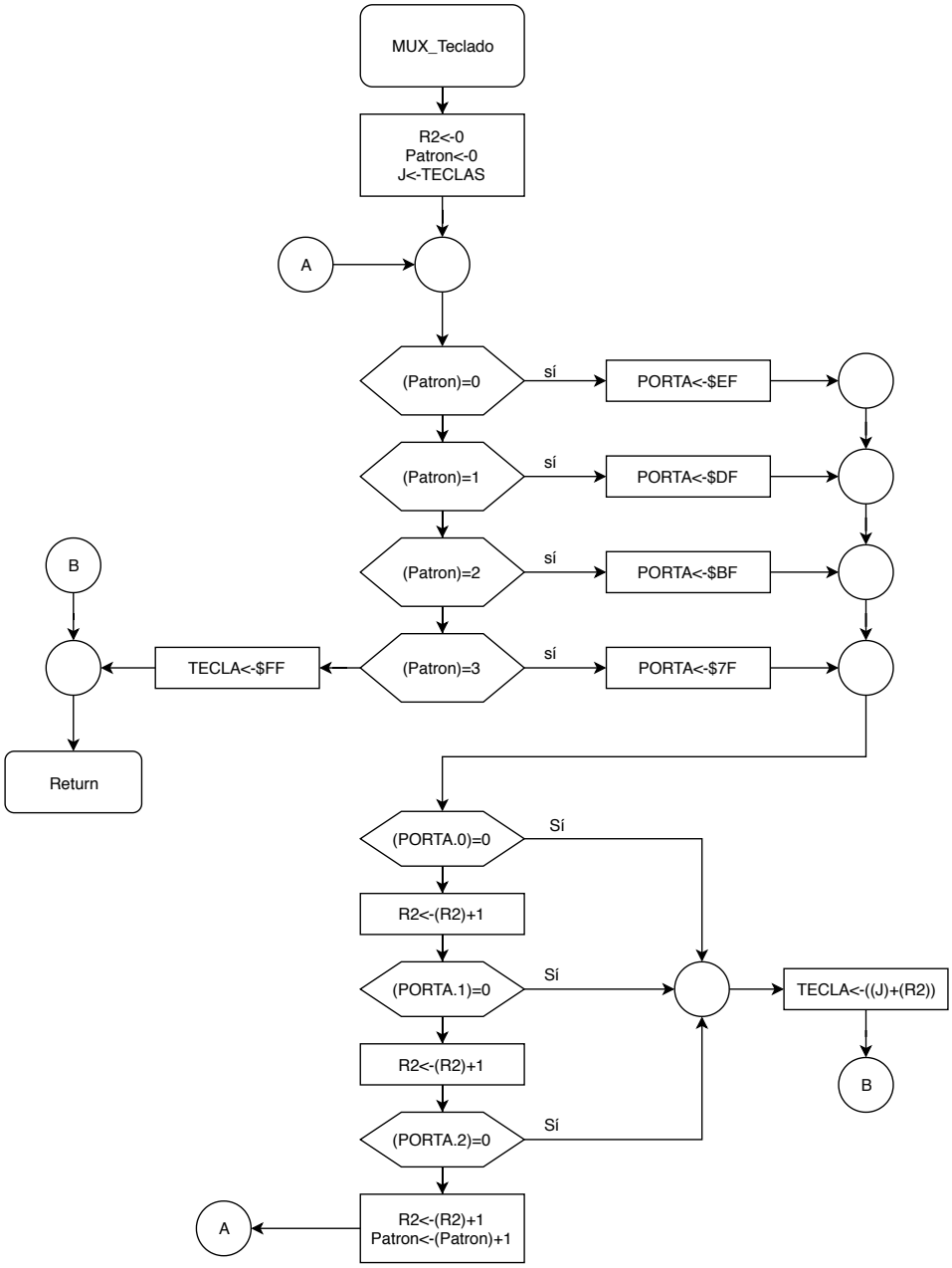

Figura 9: Diagrama de la subrutina mux teclado

En al figura 9 se muestra el diagrama de flujo de esta subrutina, como se mencionó primero se debe introducir un patrón predeterminado en el puerto A para poder detectar cuál es la tecla presionada, los patrones validos son $EF, $DF, $BF y $7F que corresponden a un conjunto de 1 s dejando un bit en 0. Seguidamente se detecta un 0 en alguno de los bits de lectura, en caso de no encontrarlo se prueba el siguiente patrón.

Paso de parámetros

- **Entrada**
  - TECLAS: Tabla que incluye los valores de cada una de las teclas del teclado.
- **Salida**
  - TECLA: Valor de la tecla presionada.

3.4.3. Formar_Array

Esta subrutina es la encargada de almacenar los valores del teclado en memoria, para esto los valores de TECLA_in y TECLA deben ser iguales se tiene una cantidad máxima de teclas determinada por MAX_TCL, además, al presionar la tecla Enter no se pueden introducir más teclas, también al presionar la tecla B se puede eliminar un valor de memoria. El diagrama correspondiente se presenta en la figura 10.

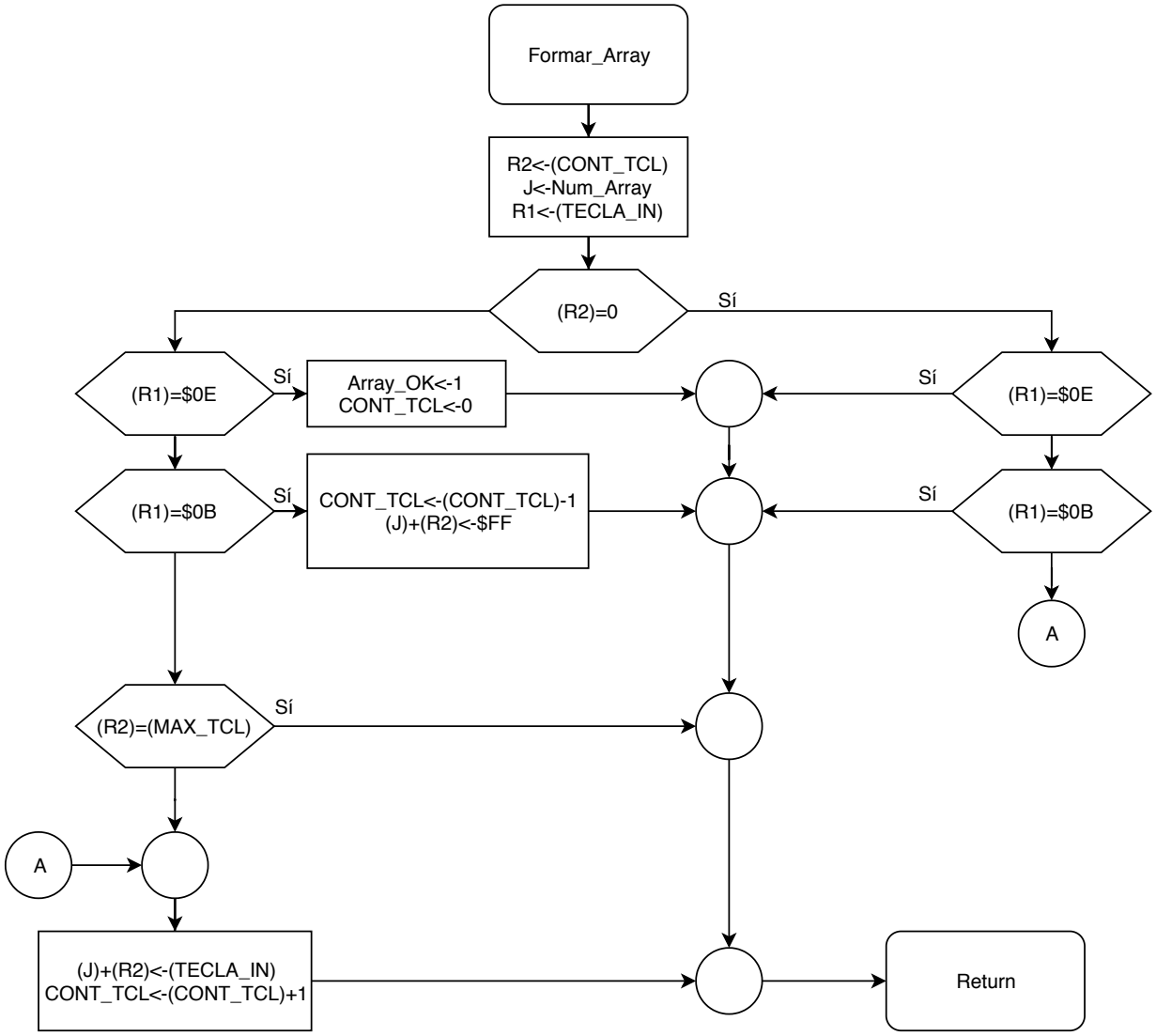

Figura 10: Subrutina asociada a Formar Array

Esta subrutina debe verifica la posición actual del último valor en el arreglo por medio de la variable CONT_TCL, además se debe revisar si hay espacio en memoria, ya que solo se admiten dos teclas. Si la tecla es $0B y el valor de la cantidad de teclas es distinto de 0 se debe borrar una tecla. Si la tecla es $0E se levanta la bandera Array_Ok que indica que ya la lectura de teclas se ha completado. Finalmente si no es ninguno de estos dos casos y no se ha llegado al máximo de teclas se almacena el valor recibido en memoria.

Paso de parámetros

- **Entrada**

- MAX_TCL: Cantidad máxima de teclas en el arreglo.
- TECLA_IN: Valor de la tecla presionada.
- CONT_TCL: Puntero a la tecla actual.

▪ **Salida**

- NUM_ARRAY: Arreglo donde se almacenan los valores en memoria.

3.4.4. BCD_7Seg

Subrutina encargada convertir valores en formato BCD a los valores necesarios para poder visualizarlos en la pantalla de 7 segmentos.

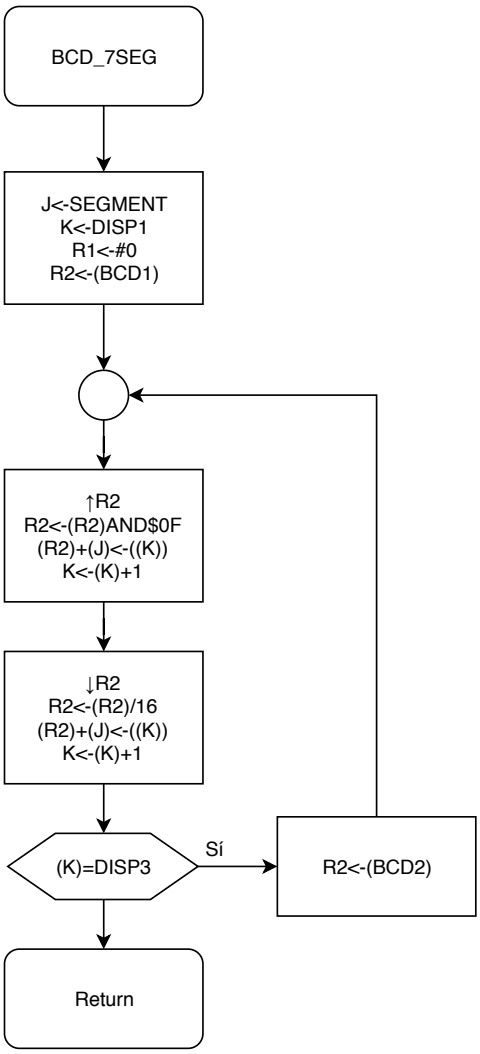

Figura 11: Subrutina utilizada para convertir valores en formato BCD a dígitos en 7 segmentos

Esta subrutina utiliza los datos contenidos en BCD1 y BCD2 para generar un offset y así direccionar la tabla Segment y encontrar el valor en 7 segmentos, una vez que se tiene el offset correspondiente (parte baja de BCD para DISP 1 y 3, parte alta para DISP 2 y 4), se mueve el valor de la tabla a la posición en memoria correspondiente para cada dígito.

Paso de parámetros

- **Entrada**
  - BCD1,BCD2: Valores en bcd a convertir.
  - SEGMENT: Tabla que contiene los valores para cada uno de los digitos.
- **Salida**
  - DISP1,DISP2,DISP3,DISP4: Valores en 7 segmentos para cada uno de los displays.

3.4.5. CARGAR_LCD

Subrutina encargada de enviar la información a desplegar en la pantalla LCD. En la figura 12 se observa el diagrama de flujos correspondiente.

Inicialmente es necesario enviar los comandos almacenados en la tabla IniDsp que corresponden a la inicialización de las comunicaciones con la pantalla LCD, para esto se utiliza la subrutina Send, indicando por medio del bit 6 de BANDERAS si es un comando o un dato.

Seguidamente, se envía el comando para borrar los contenidos de la pantalla (Clear_LCD) y se genera un tiempo de espera de 2 ms según las especificaciones del fabricante.

Luego, después de enviar el comando ADD_L1 se envían los datos de la primera línea del mensaje que se desea mostrar en pantalla. Una vez que se envía el último valor y se encuentra un carácter nulo se procede a enviar el comando de añadir la línea 2 y los datos correspondiente.

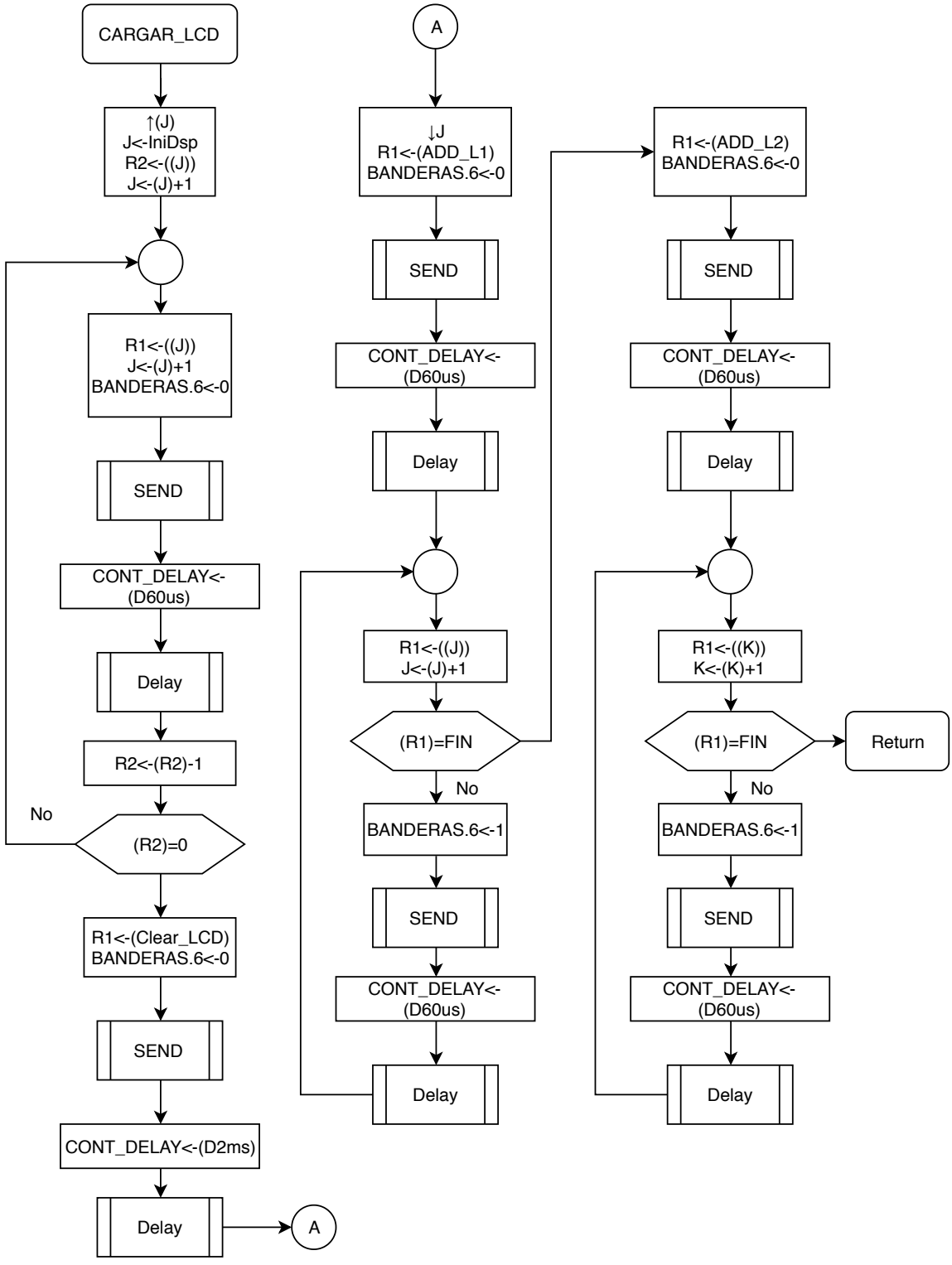

Figura 12: Subrutina utilizada para cargar mensajes en la pantalla LCD

Paso de parámetros

- **Entrada**
  - iniDsp: Tabla de comandos para iniciar la comunicación con la pantalla
  - ADD_L1,ADD_L2: Comandos para añadir lineas

- Registro X: Contiene el puntero para el contenido de la linea 1
- Registro Y: Contiene el puntero para el contenido de la linea 2
- D60us,D2ms: Constantes para el tiempo de espera necesario para la comunicación correcta con la pantalla.

- **Salida**
  - Registro A: Contiene los datos a enviar por medio de la subrutina Send

3.4.6. Send

Subrutina encarga de enviar los datos o comando a la pantalla LCD.

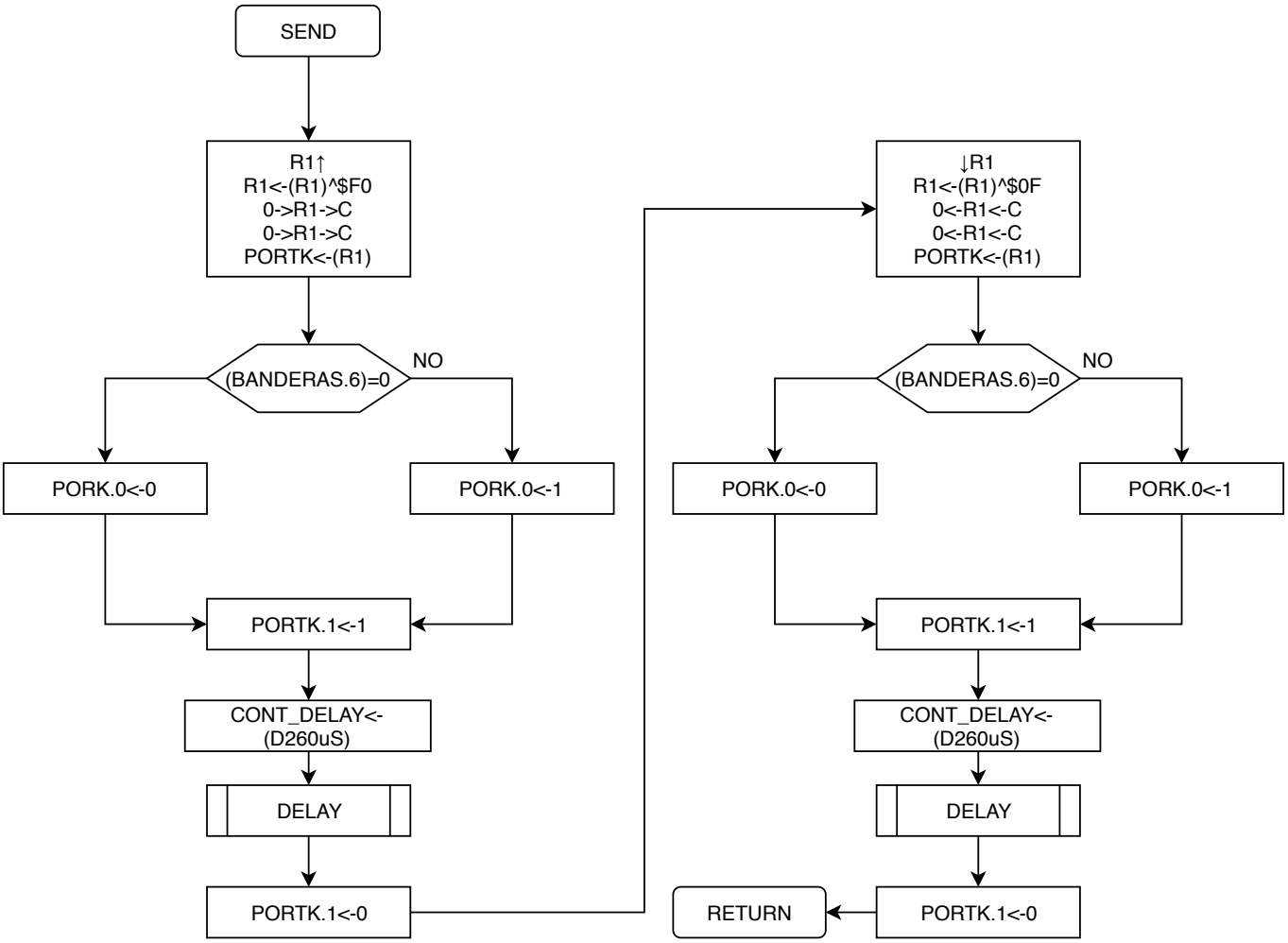

Figura 13: Diagrama de la subrutina usada para enviar comandos o datos a la pantalla LCD

En la figura 13 se muestra el diagrama de flujo, como se observa inicialmente se verifica si es un comando o un dato y dependiendo de esta condición se configura el bit 1 del puerto K (0 si es un comando 1 si es un dato), además se debe generar un corrimiento de los datos ya que estos envían en los bits del 2 al 6 del puerto K, también es necesario mencionar que los datos se envían en dos segmentos, primero la parte alta y luego la parte baja.

Paso de parámetros

- **Entrada**

- Banderas.1: Indica si es un comando o datos
- Registro A: Contenido del comando o dato a enviar a la pantalla.
- D240us: Constante con el valor necesario para generar el delay.

3.4.7. Delay

Subrutina de retraso, solo debe consumir el tiempo necesario para la correcta comunicación con la pantalla. Como se observa en la figura 14 esta subrutina se ejecuta haya que el valor CONT_Delay sea 0, este es modificado por la subrutina OC4.

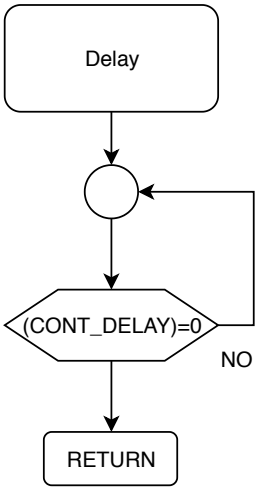

Figura 14: Diagrama de la subrutina usada para generar retardos

Paso de parámetros

- **Entrada**
  - CONT_DELAY: Contador que indica que tanto se debe esperar.

3.4.8. CONV_BIN_BCD

Subrutina de llamado a conversión Binaria a BCD, determina si el valor se en binario se puede convertir o si es el correspondiente a apagar la pantalla o mostrar guiones.

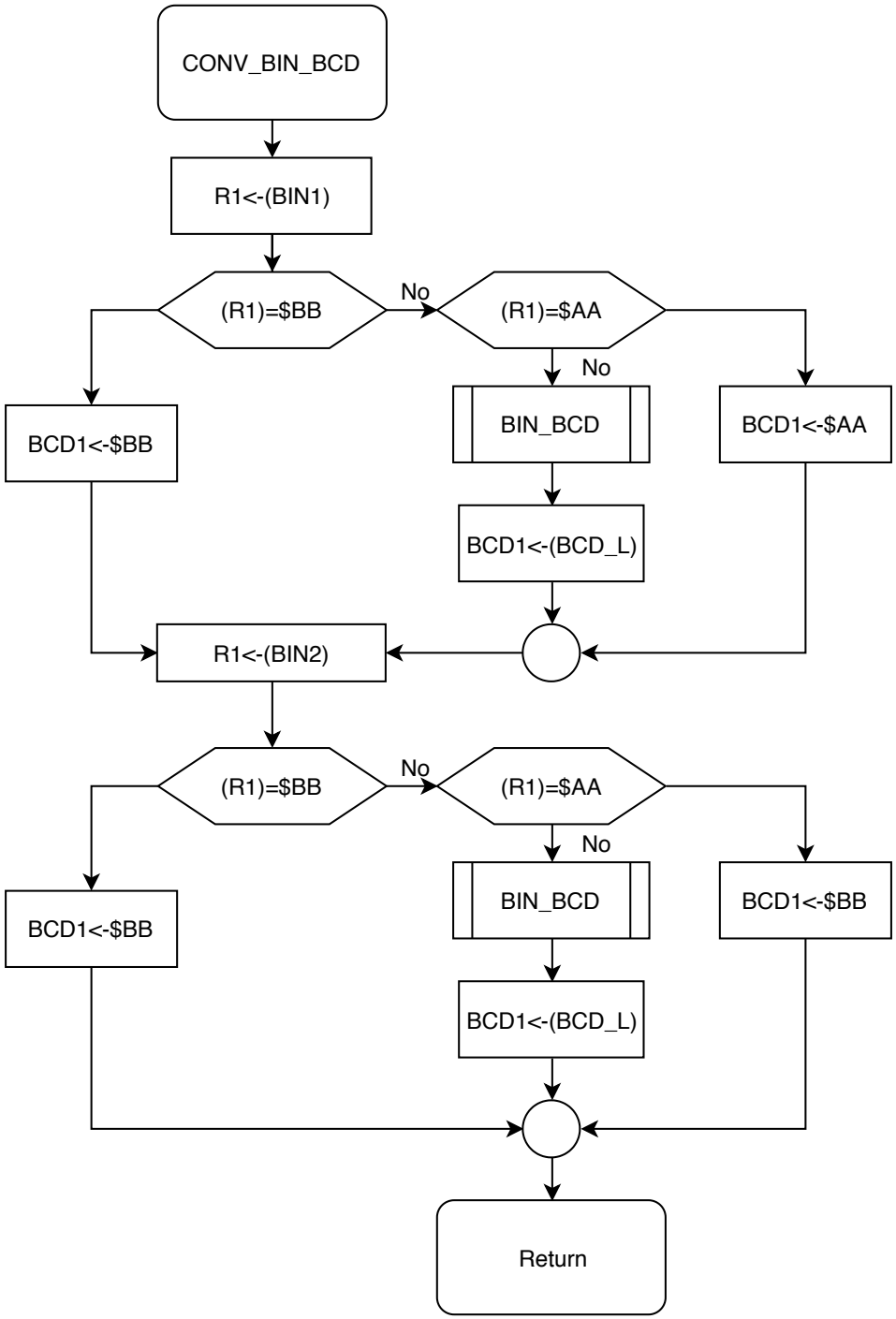

Figura 15: Subrutina usada para enviar convertir datos en binario al formato BCD

Como se puede observar en la figura 15 esta subrutina solo verifica que el valor no sea $AA o $BB antes de llamar a la subrutina BIN_BCD, en caso de que sea uno de estos valores solo se copian a la posición BCD correspondiente.

Paso de parámetros

- **Entrada**
  - BIN1,BIN2: Valores a convertir.
  - BCD_L: Valore convertido por subrutina BIN_BCD.

- **Salida**
  - Registro A: Valor que la subrutina BIN_BCD debe convertir.
  - BCD1,BCD2: Valores convertidos.

3.4.9. BIN_BCD

Conversión de 1 valor en binario de 8 bits a BCD.

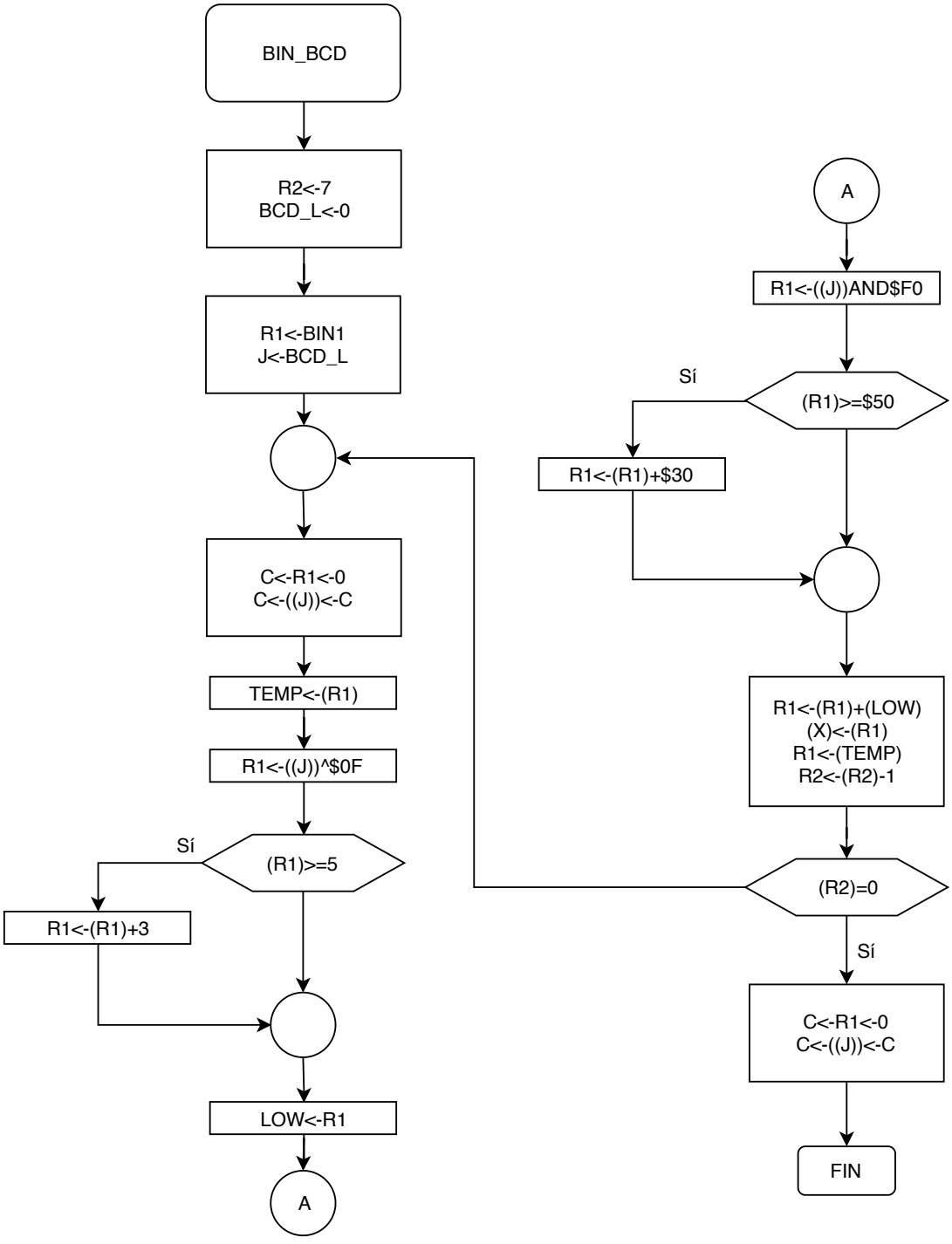

Figura 16: Diagrama de la subrutina utilizada para la conversión binario BCD

Como se puede notar en la figura 16 esta subrutina se realiza de la misma forma que se presenta en [5]. **Paso de parámetros**

- **Entrada**
  - Registro A: En este registro se encuentra el contenido a convertir.
- **Salida**
  - BCD_L: Posición de memoria donde se retorna el resultado en BCD.

3.4.10. BCD_BIN

Conversión de BCD a binario, utilizada para convertir el valor del arreglo del teclado a un valor numérico. El diagrama de la figura 17 corresponde a esta subrutina, en la cual resumidamente se multiplica la parte alta (primer valor de Num_Array) del valor en BCD por 10 y luego se le suma el valor de la parte baja.

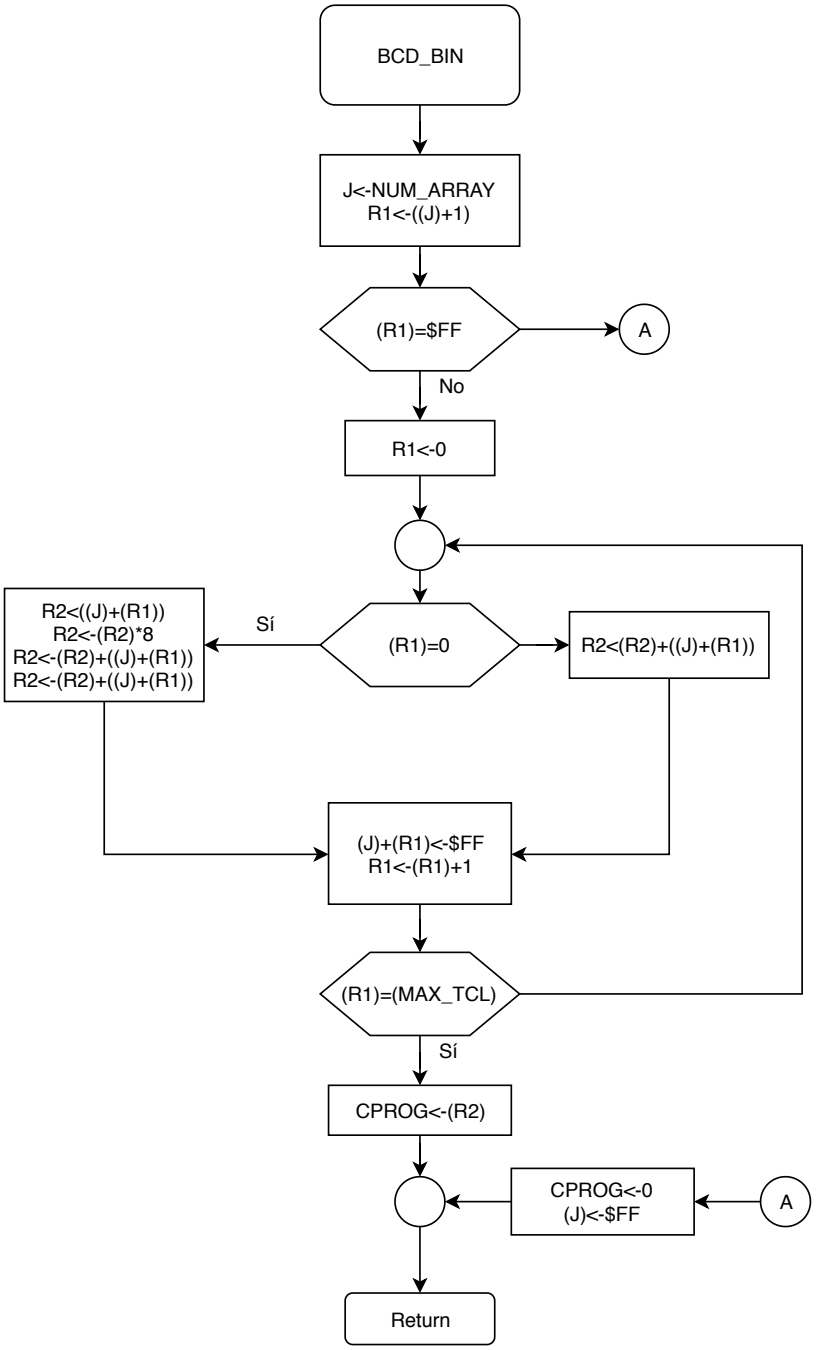

Figura 17: Subrutina utilizada para convertir datos en formato BCD a binario

Paso de parámetros

- **Entrada**
  - NUM_ARRAY: Arreglo de número en BCD producidos por el teclado.
- **Salida**
  - V_LIM: Valor en binario resultante de la conversión, corresponde a la velocidad limite.

3.4.11. MODO_Config

Modo de operación del sensor de velocidad, en este modo se configura la velocidad limite del sensor. Para esto se debe verificar que el valor ingresado en el teclado sea valido es decir entre el rango de 45 y 90 km/h, si no es valido se borra el valor ingresado, de ser valido se muestra en pantalla y se mantiene en memoria. Antes de realizar estas acciones se debe verificar que se haya ingresado al menos un valor y presionado la tecla Enter, esto se hace al revisar la bander ARRAY_OK. Todo esto se aprecia con mayor detalle en la figura 18

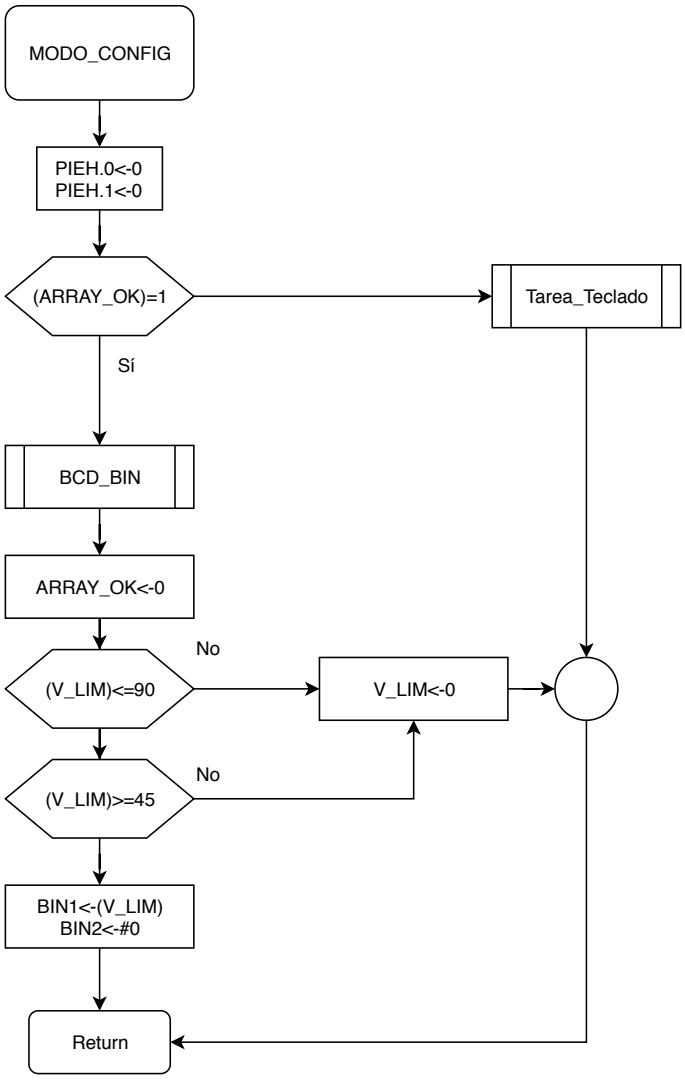

Figura 18: Diagrama de la subrutina asociada al modo configuración

Paso de parámetros

- **Entrada**
  - V_LIM: Velocidad limite ingresada.
- **Salida**
  - BIN1: Valor en binario que se debe mostrar en los displays de 7 segmentos 1 y 2
  - BIN2: Valor en binario que se debe mostrar en los displays de 7 segmentos 3 y 4, en este caso BB para que se apaguen

3.4.12. MODO_Medición

Modo de operación del sensor de velocidad, determinar si la velocidad es mayor a 0, si lo es llama a la subrutina de control de la pantalla, además, cuando el vehículo pasa por el primer sensor muestra el mensaje Çalculando.[en] pantalla, esto se indica por medio del tercer bit del registro de banderas. En el caso de que no se den las condiciones mencionadas la subrutina no realiza ninguna acción. Esto se puede ver descrito gráficamente en la figura 19

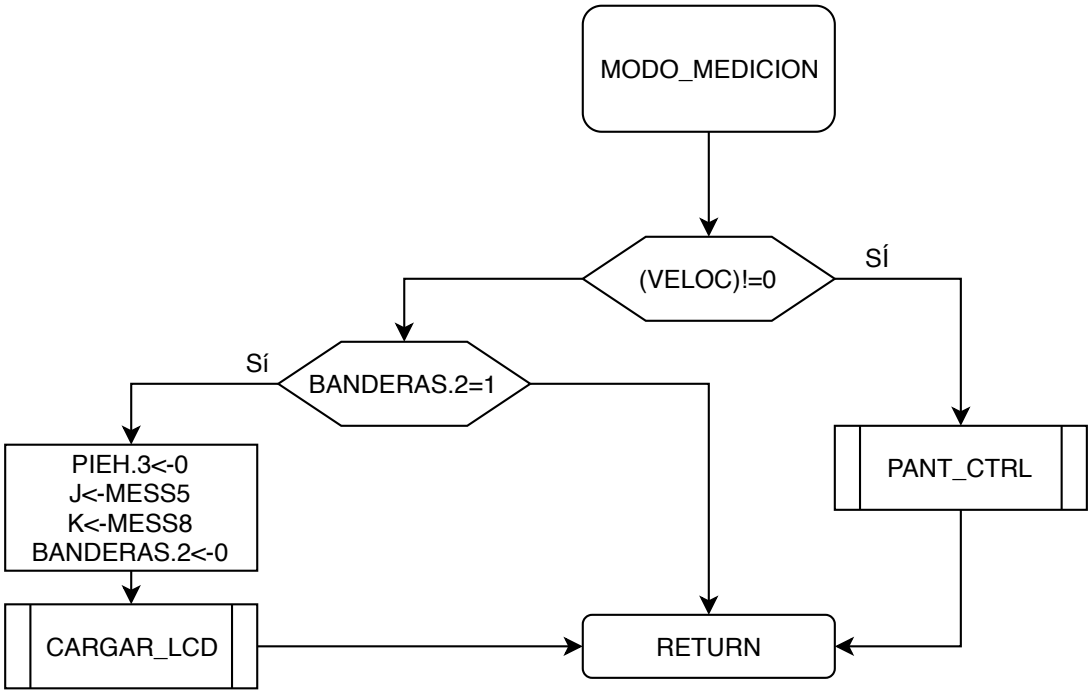

Figura 19: Diagrama de la subrutina del modo medición

Paso de parámetros

- **Entrada**
  - BANDERAS.4: Bandera que indica cuando imprimir el mensaje.
  - VELOC: Velocidad actual del vehículo.
- **Salida**
  -

3.4.13. MODO_Libre

Modo de operación del sensor, este es un modo donde no se realiza ninguna operación, solo se borran los valores desplegados en las pantallas de 7 segmentos por medio de la escritura de $BB en las posiciones de memoria correspondientes. El diagrama correspondiente a este modo se encuentra en la figura 20

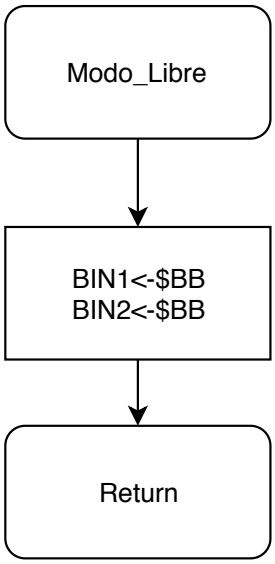

Figura 20: Diagrama de flujos del modo libre

Paso de parámetros

- **Salida**
  - BIN1,BIN2: Se debe borrar la pantalla de 7seg, para esto se envia BB a estas posiciones de memoria.

3.4.14. PANT_CTRL

Subrutina encargada de manipular las pantallas cuando se detecta el paso de un vehículo, si la velocidad se sale de los limites que establecidos para el sensor se muestran guiones acompañados de la velocidad limite configurada, si es un valor valido pero mayor que la velocidad limite se activa una alarma en los LEDs, luego se muestra la velocidad cuando el vehículo pasa 100 m después del segundo sensor, lo mismo ocurre para una velocidad adentro del limite pero no se activa la alarma de los LEDs.

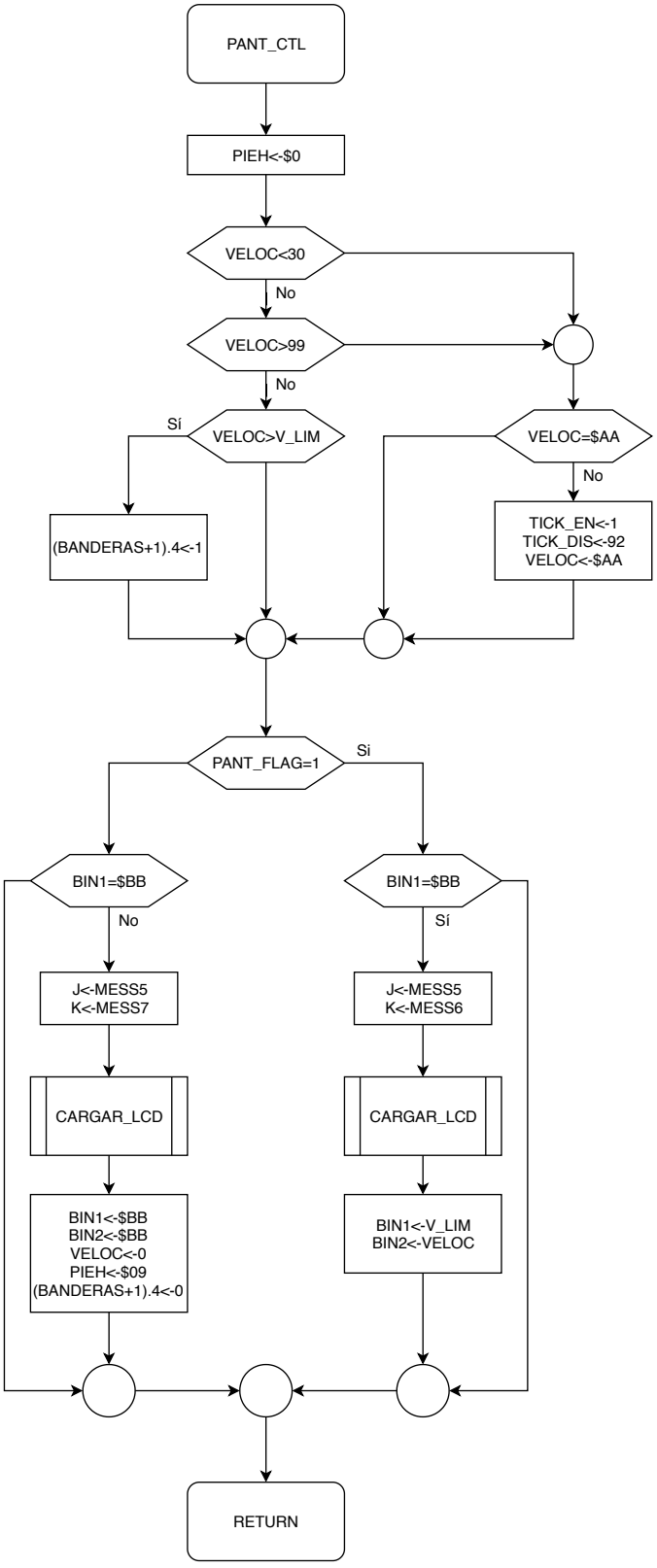

Figura 21: Diagrama de flujos del control de las pantallas en el modo medición

Como se mencionó anteriormente está subrutina fué modificada con respecto a las especificaciones, debido a que al ingresar a esta subrutina no se cuenta con el valor de TICK_VEL ya que es borrado

al pasar por el segundo sensor.

Paso de parámetros

- **Entrada**
  - VELOC: Velocidad del vehiculo
  - V_LIM: Velocidad limite, configurada en modo config.
- **Salida**
  - BIN1: Se configura según la velocidad del vehículo, se apaga (BB) cuando ya el vehículo pasa el sensor
  - BIN2: Se envía el valor de la velocidad limite, se apaga (BB) cuando ya el vehículo pasa el sensor
  - (BANDERAS+1).4: Bandera de alarma.

3.4.15. Patrón_LEDS

Controla lo que se muestra en los LEDs en el modo medición, si la alarma esta activada se encienden en forma secuencial los LEDs de 3 a 7.

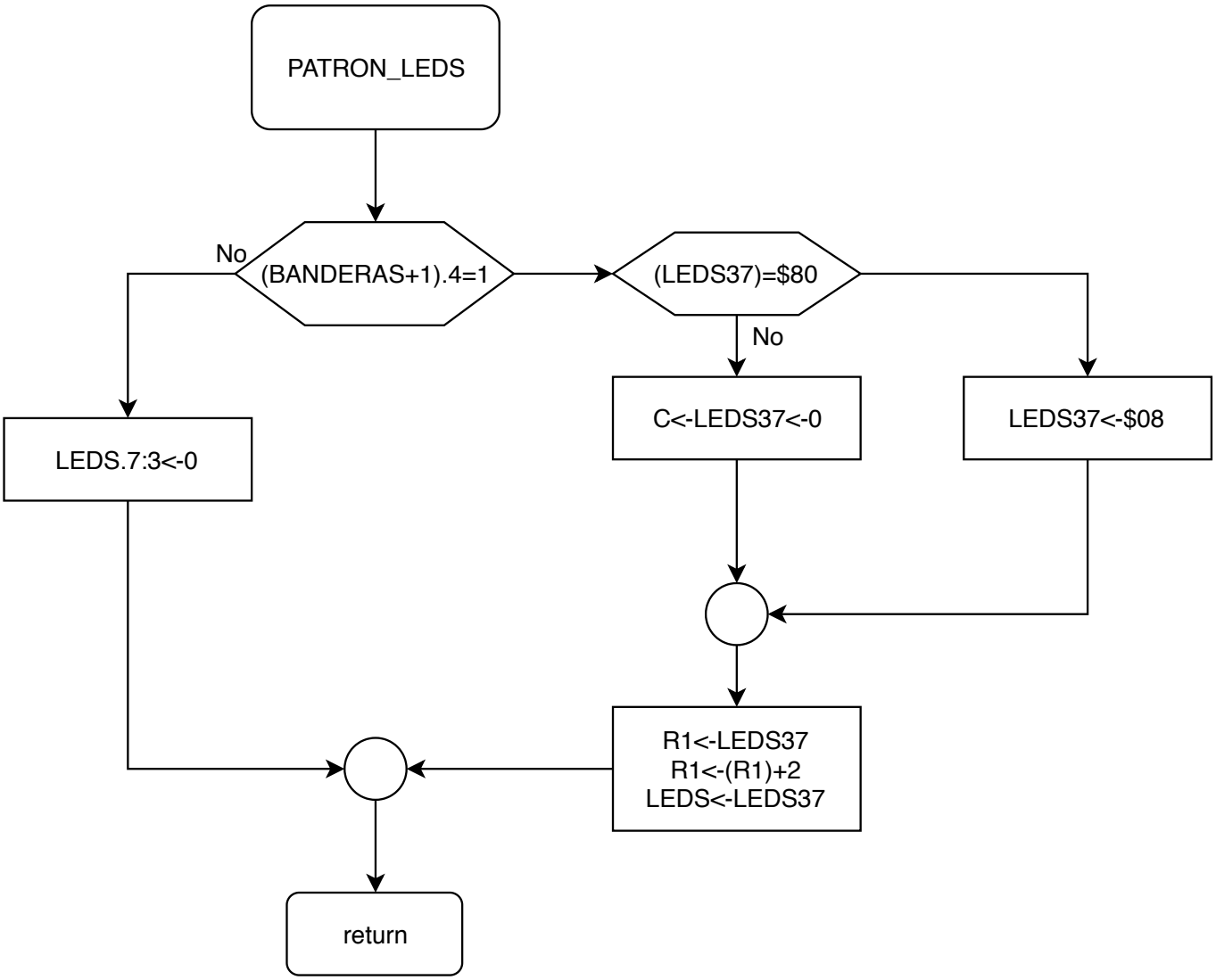

Figura 22: Diagrama de la subrutina usada para los LEDs cuando hay una alerta

Cómo se observa en la figura 22 inicialmente se debe revisar el registro de banderas para verificar si la alerta está activa, de ser así se realiza el corrimiento del led que debe estar encendido, si es el último led se retorna al primer led de la alarma (LED 4).

Paso de parámetros

- **Entrada**
  - (BANDERAS+1).4: Bandera de alarma.
- **Salida**
  - LEDs: Variable que controla el patron de los LEDs.

4. Conclusiones y Recomendaciones

A partir de los resultados se pueden realizar las siguientes conclusiones y recomendaciones.

- Al analizar el funcionamiento del programa en la tarjeta dragon 12 se puede concluir que cumple con todos los requisitos establecidos previamente.
- Se puede concluir que a partir de los cambios realizados a las subrutinas ph0 y Pant_CTRL se simplifican los cálculos a realizar.
- También se puede concluir que es necesario establecer un buen diseño de programa antes de programar para evitar dificultades en el desarrollo.
- Se recomienda trabajar de forma ordenada y modular para poder reducir la cantidad de código o utilizar múltiples veces el mismo código.
- Es recomendable verificar la correcta escritura de los registros de control para que estos operen de la forma deseada.
- Finalmente se recomienda revisar cuidadosamente el borrado de las banderas de interrupción para asegurar el correcto funcionamiento del programa

5. Anexos

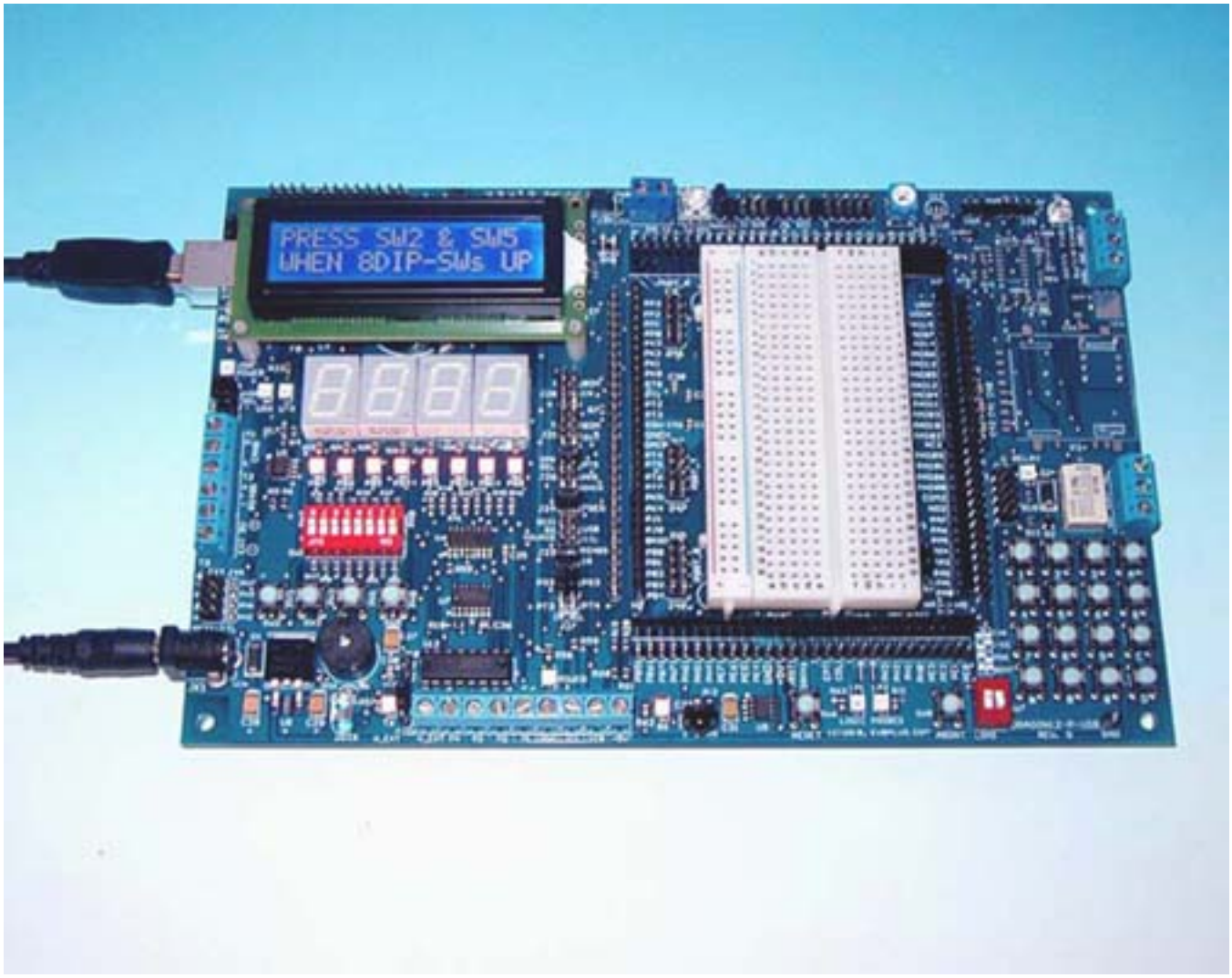

Figura 23: Fotografía de la tarjeta Dragon 12 plus, obtenida de [1]

Referencias

[1] EVB, "Dragon12-plus-usb features:." [Online]. Available: http://www.evbplus.com/9s12/9s12_hcs12.html

[2] F. H. microcontroller family, "User's manual," *Dragon12-Plus-USB Trainer*, 2005. [Online]. Available: http://www.evbplus.com/download_hcs12/dragon12_plus_usb_9s12_manual.pdf

[3] G. Delgado, "Ii parte dispositivos perifericos," *Microprocesadores IE-623*, 2019.

[4] ——, "Proyecto final, radar 623," *Microprocesadores IE-623*, 2019.

[5] ——, "I parte," *Microprocesadores IE-623*, 2019.