Universidad de Costa Rica

Facultad de Ingeniería

Escuela de Ingeniería Eléctrica

IE-0623: Microprocesadores

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

Proyecto Final: RunMeter 623

Estudiante:
Moisés Campos Zepeda

Carné:
B31400

II semestre - 2020

Índice

1. Resumen

El presente trabajo consiste en un programa, ensamblado por medio de la herramienta AsmIDE, para un sistema de despliegue de información de un velódromo, denominado RunMeter623, el cual permite el despiegue de información en una pantalla de 4 dígitos de 7 segmentos y una pntalla LCD $2x16$ como se muestra en la figura (figura pendiente). Para esto cuenta con 4 modos:

- **Modo Configuración**: Se ingresa el numero máximo de vueltas *NumVueltas*, esto se hace por medio del uso del teclado matricial de la tarjeta y los displays de 7 segmentos para mostrar dicho valor.
- **Modo Competencia**: Modo en el que se calcula la velocidad de un ciclista con base en el tiempo transcurrido entre la detección de los sensores S1 (pulsador PH3) y S2 (pulsador PH2) así como las vueltas (numero de detecciones validas) realizadas hasta llegar a *NumVueltas*.
- **Modo Resumen**: Se despliega la velocidad promedio de las vueltas detectadas en el Modo Competencia, así como la cantidad de vueltas realizadas.
- **Modo Libre**: Muestra un mensaje en la pantalla LCD, se utiliza para poner al sistema en un estado ocioso si no se desea realizar ningún cálculo.

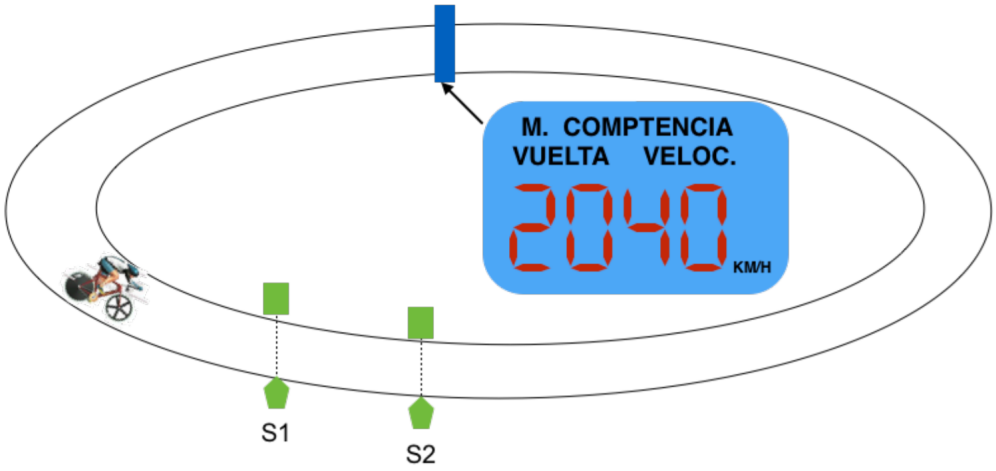

Figura 1: Imagen descriptiva de la operación de RunMeter 623. Tomada de [1].

Dentro de los resultados, cada modo presenta sus mensajes respectivos y se presenta la operación deseada de los sensores. Además del funcionamiento de las 22 subrutinas solicitadas (incluyendo el programa principal). El código de la solución presenta un peso de 2170 bytes, almacenados en EEPROM.

2. Diseño de la Aplicación

2.1. Delaración de estructuras de datos adicionales

Adicional a las Estructuras solicitadas conforme a la tabla de la figura 22 del Anexo. Se presentan las siguientes estructuras y su justificación.

- **BANDERAS_2:** variable de 2 bytes, la cual posee 2 banderas adicionales:
  - **SendData_flg:** permite controlar si se mandan comandos o datos en **SEND** para no tener que utilizar 2 subrutinas altamente redundantes.
  - **Calc_flag:** determina cuando mandar el mensaje de *"Calculando..."*
- **V_MIN, V_MAX:** Constantes de 1 byte con los valores de velocidad mínima y máxima iguales a 35 y 95 respectivamente, permiten hacer el programa más modular en torno a los límites superior e inferior de velocidad.
- **CONT_RTI**: variable que se compara con CONT_200 debido a que esta segunda es una constante conforme al enunciado. permite dar inicio a las conversiones ATD cada $200ms$.

2.2. Programa principal

Considerando que el microprocesador 9s12 no tiene un sistema operativo, la arquitectura consta de un programa que se ejecuta constantemente y permite 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.

2.3. Memoria de cálculos y 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 2 se puede apreciar el diagrama de flujos utilizado para la configuración e inicialización de variables conforme a [2].

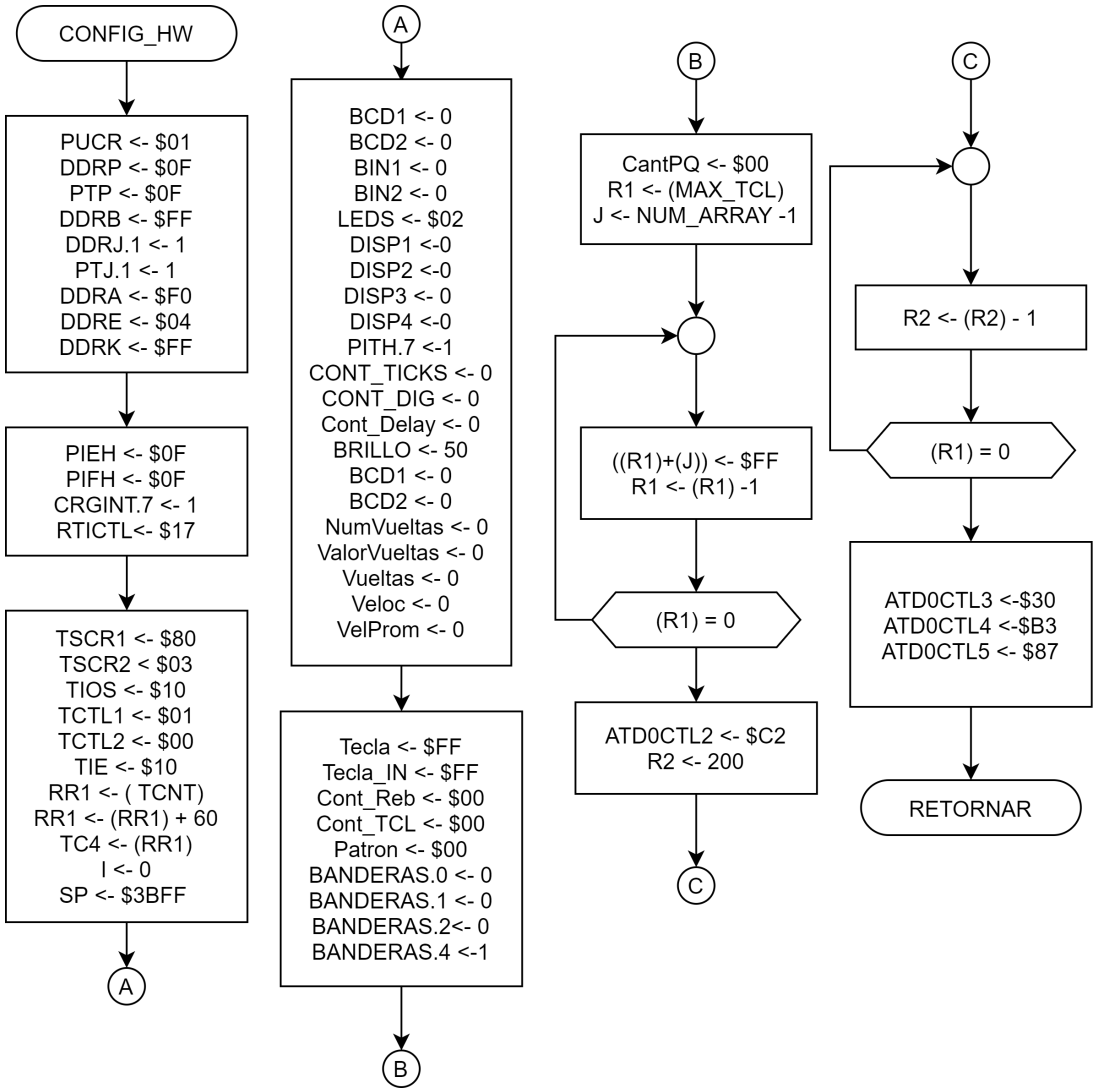

Figura 2: Diagrama de la configuración de hardware e inicialización de variables.

A continuación se presenta el contenido de cada uno de los registros de control:

RTI

En este caso se considera un tiempo para la interrupción de $T_{RTI} = 1ms$ por lo cual tomamos la siguiente ecuación:

$$T_{RTI} = \frac{(N+1) \cdot 2^{(M+9)}}{OSC\_CLK}$$

Despejando la ecuación anterior con $OSC\_CLK = 8M\text{Hz}$ y $M = 1$ se tiene ahora:

$$N = \left\lceil \frac{T_{RTI} \cdot OSC\_CLK}{2^{(M+9)}} - 1 \right\rceil = 7 \tag{1}$$

Con $\lceil N \rceil$ siendo un redondeo a número entero superior de N. Se obtiene así $RTICTL = \$17$

TCNT

- En TSCR1 se activa el contador de tiempos TEN.
- En TSCR2 se pone el preescalador en 8.
- En TIOS se pone activa IOS4 para activar la salida por comparación en el pin 4 del **puerto T**, la cual se manejara con la rutina **OC4_ISR**.
- En TCTL1 se escribe un 1 en el primer bit para configurar la salida del puerto 4 como un *toggle*.
- En TIE se habilitan las interrupciones del puerto 4.

LEDS y pantallas (de 7 segmentos y LCD)

- Se maneja su parpadeo e intensidad mediante la interrupción OCR4. Inicialmente se reinician los valores a mostrar (**DISP1, DISP2, ..., DISP4**) y se ingresa un valor de BRILLO medio de 50 para el display y los LEDS.
- Para la pantalla LCD se reinicia la variable **Cont_Delay** y se habilita la pantalla con **DDRK**.

Output compare (OC4)

Además se calcula el valor a sumar al contador para generar una interrupción a 50 kHz en el output compare 4. Así podemos demostrar con la ecuación 3 el valor añadido a **TCNT**.

$$TOC = \frac{TCS \cdot PRS}{BUS\_CLK} \tag{2}$$

$$TC4 = \frac{24MHz}{8 \cdot 50kHz} = 60 \tag{3}$$

Configuración de ATD

- Para ATD0CTL2 se habilita ADPU para activar el módulo, el "Fast flag clear all"(AFFC), y las interrupciones con ASCIE. Además nos esperamos 10µs para continuar mientras iniciamos el ATD
- Para ATD0CTL3 definimos 6 conversiones.
- en ATD0CTL4 utilizamos 8 bits, 4 ciclos ATD, y la frecuencia a 600kHz para un PRS igual a:

$$PRS = \frac{Bus\_clk}{2 \cdot 600k\text{Hz}} - 1 = 19 \tag{4}$$

- Para ATD0CTL5 se activan DJM (justifica a la derecha), no se usa signo, y se lee del canal 7 ($CA = CB = CC = 1$).

2.3.1. Rutina Main

Rutina que lee el dipswitch y maneja los diferentes modos de la aplicación. En el se hace la detección de los modos por medio de los dipswitch en PTH7 y PTH6, además de activar/desactivar PTIH y TCNT.

Entradas

- **NumVueltas:** debe ser mayor a 0 para salir de Modo Configuración

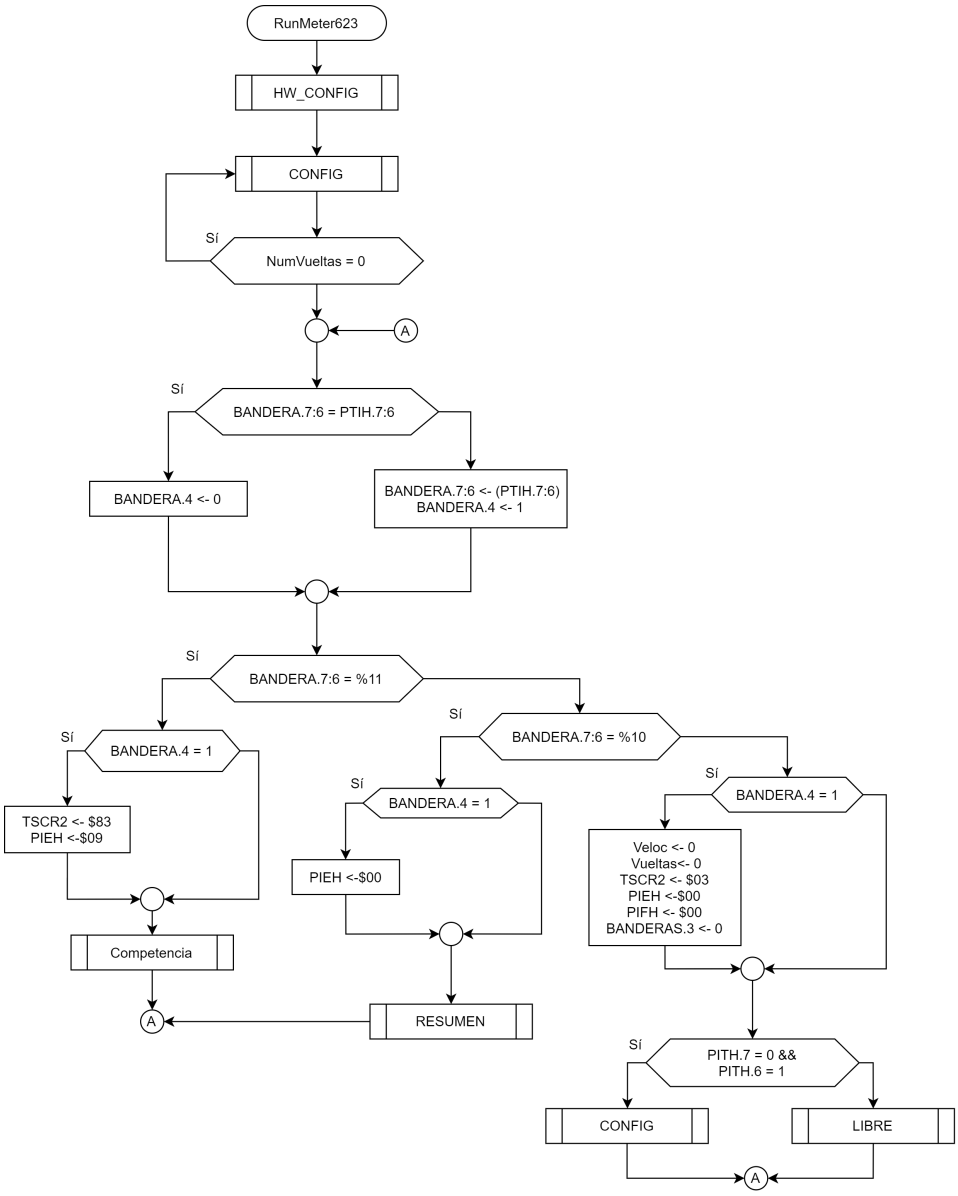

Figura 3: Diagrama de la configuración de Programa inicial con la lógica de selección de modos.

2.4. Subrutinas de atención a interrupciones (RTI)

2.4.1. ATD_ISR

Subrutina de convertidor analógico-digital, calcula **BRILLO** desde el el potenciómetro, con la variable **POT** con lo cual, al multiplicarse a plena escala en **DT** se controla el brillo del display de 7 segmentos en la rutina **TCNT_ISR** (ver Figura 8).

Entradas

- **ADR00H,ADR01H,...,ADR05H:** Registros de datos del convertidor analógico-digital.

Salidas

- **BRILLO:** variable correspondiente a k en el ciclo de trabajo del control visto en el algoritmo de los LEDS y pantalla de 7 segmentos.
- **DT:** Duty time, variable que determina cuanto tiempo deben permanecer los LEDS y pantalla de 7 segmentos.

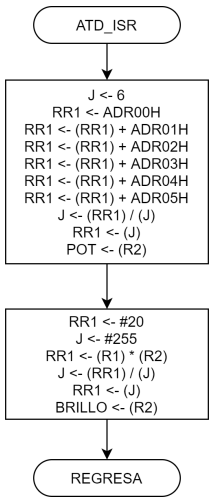

Figura 4: Diagrama de la subrutina **ATD_ISR** donde se hace el promedio de 6 mediciones del potenciómetro para controlar el brillo de la pantalla de 7 segmentos.

2.4.2. CALCULAR_ISR (Puerto H)

Rutina que lee e puerto H para los botones que simulan los sensores S1 y S2. Calcula **Veloc** y **VelProm** , así como los ticks necesarios para mostrar y borrar la pantalla.

Entradas

- **Cont_Reb:** cancela los rebotes de los botones SW2 y SW5.

Salidas

- **Veloc:** Velocidad detectada en Modo Competencia para la ultima vuelta iniciada.
- **VelProm:** Promedio de las velocidades anteriores, es igual a Veloc en la primera vuelta.

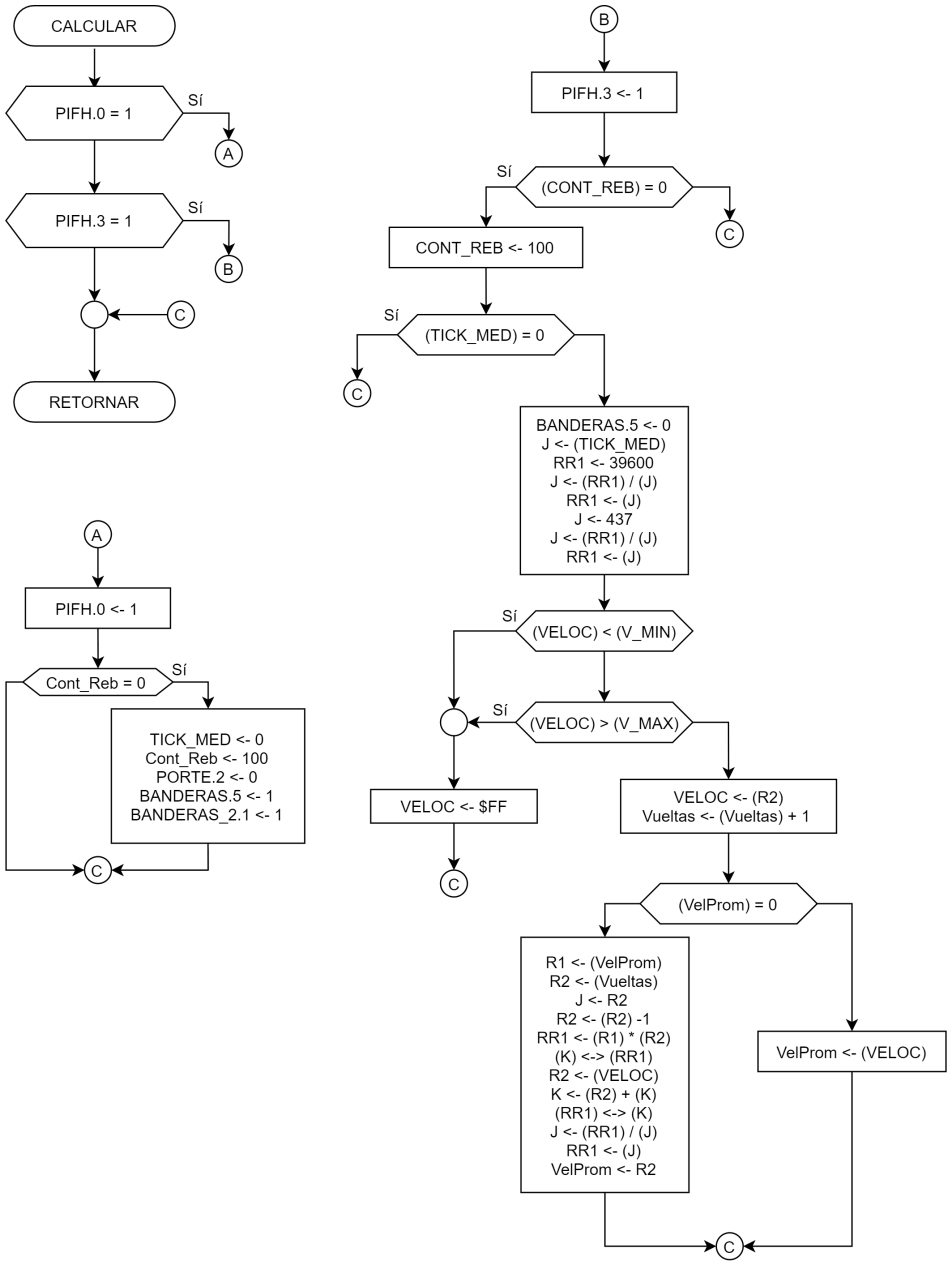

Figura 5: Diagrama para la subrutina de atención a interrupciones de PTIH, permite hacer los cálculos correspondientes a los sensores S1 y S2.

2.4.3. RTI_ISR

Subrutina encarga del manejo de los rebotes de los botones y activación del ATD para realizar mediciones del potenciómetro cada 200ms.

Entradas

- **Cont_Reb:** se verifica que no sea cero.
- **Cont_200:** : constante con la que se compara CONT_RTI.
- **Cont_RTI:** : variable con la que se define cuando escribir a ATD0CTL5.

Salidas

- **Veloc:** Velocidad detectada en Modo Competencia para la ultima vuelta iniciada.

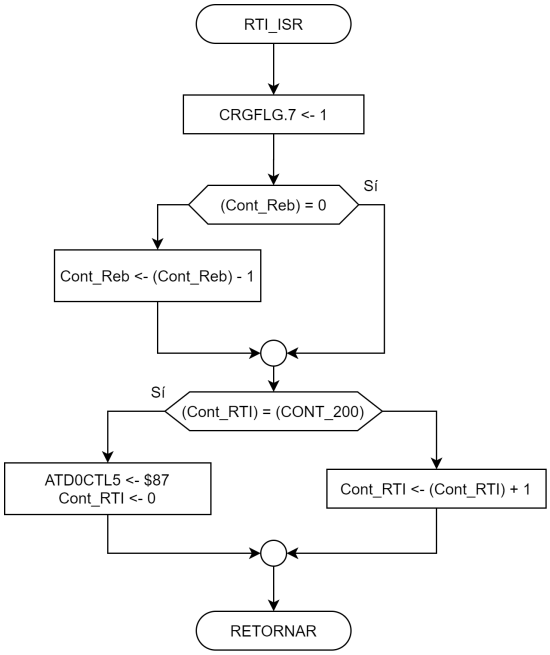

Figura 6: Diagrama para la subrutina de atención a interrupciones Real Time Interrupt, la cual activa el ATD para realizar medicines cada 200ms.

2.4.4. Output compare 4 (OC4_ISR)

Subrutina encargada del manejo de la pantalla de 7 segmentos, el contenido de los displays, el brillo y la subrutina bcd 7 segmentos. Adems decrementa CONT_DELAY, asistiendo al control de la pantalla LED. Consta de 3 partes:

- La primera maneja los ticks, buscando que lleguen al valor de la variable DT, permitiendo así apagarla pantalla apagar.
- La segunda parte corresponde al manejo de los displays de 7 segmentos mediante la variable **CONT_7SEG** para indicar cual dígito se muestra considerando una interrupción de $100ms$ la cual se calcula con la siguiente ecuación:

$$CONT\_7SEG = \frac{100ms \cdot BUS\_CLK}{PRS * TC4} \tag{5}$$

- Finalmente se maneja un contador llamado **Cont_Delay** que (TODO)

Entradas

- **DT:** determina si se apaga la pantalla, controlando así el brillo con el valor obtenido en **ATD_ISR**.
- **DISP1,DISP2,DISP3,DISP4:** contenido que se debe mostrar en el display de 7 segmentos.
- **LEDS:** Variable que determina el patrón de los leds a transferir a PORTB.

Salidas

- **CONT_DELAY:** contador de retraso para contador de pantalla de LEDS.
- **CONT_7SEG:** contador de retraso para contador de pantalla de 7 segmentos.

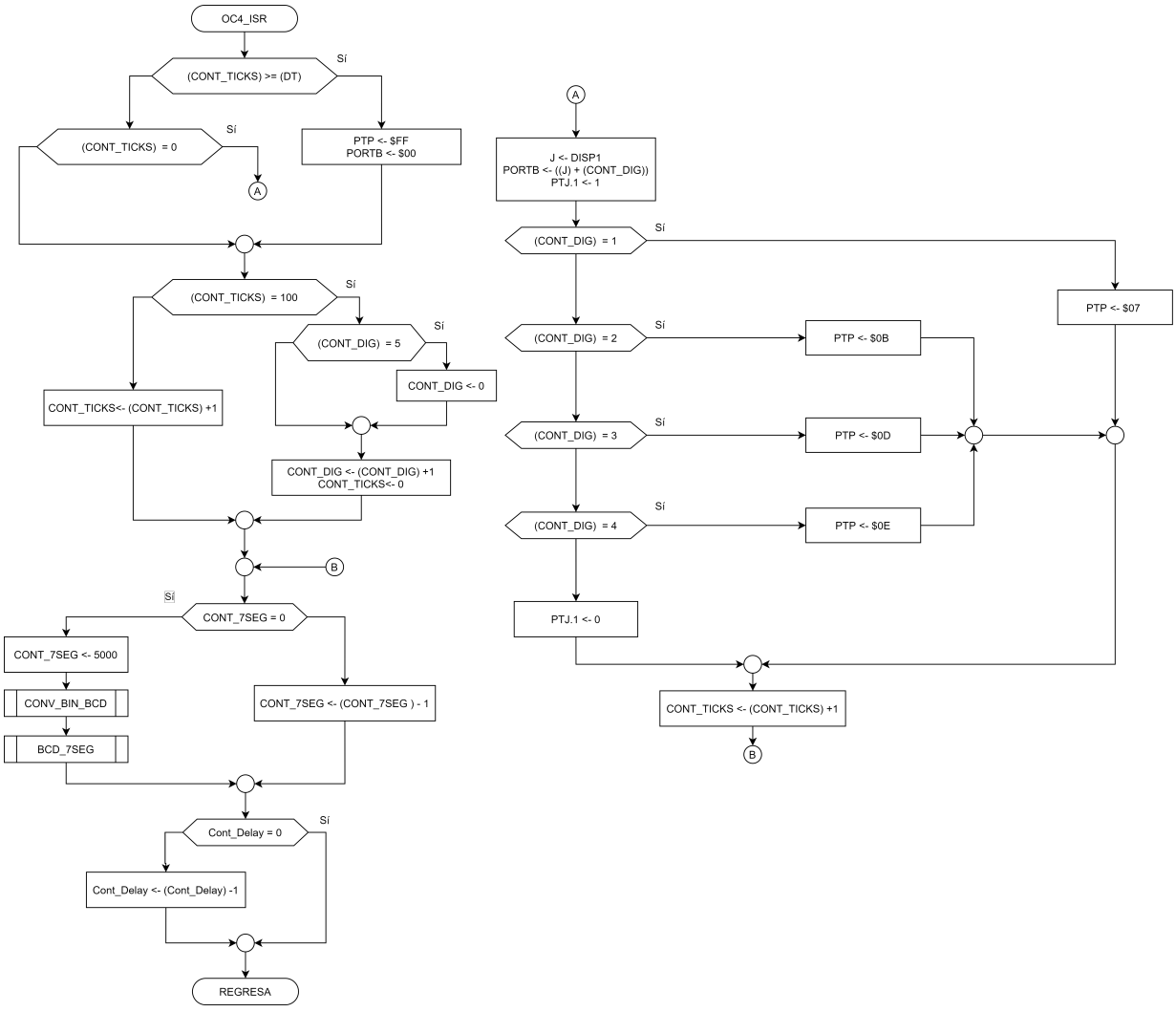

Figura 7: Diagrama de la subrutina **OC4_ISR**.

2.4.5. TCNT_ISR

Interrupción por overflow del contador de tiempo. Encargada de incrementarlos ticks que se usan para medir la velocidad del ciclista, y decrementando **TICK_EN** y **TICK_DIS** para agotar el tiempo que se muestran las velocidades y vueltas en la pantalla.

Entradas

- **TICK_EN:** Ticks recibidos de **PANT_CTRL** para activar la velocidad del medidor.
- **TICL_DIS:** Ticks recibidos de **PANT_CTRL** para desactivar la velocidad del medidor.

Salidas

- **TICK_VEL:** Ticks para medir la velocidad, se incrementa cuando el ciclista se encuentra entre los dos sensores
- **TICK_EN:** Ticks restantes para activar la velocidad del medidor.
- **TICL_DIS:** Ticks recibidos para desactivar la velocidad del medidor.

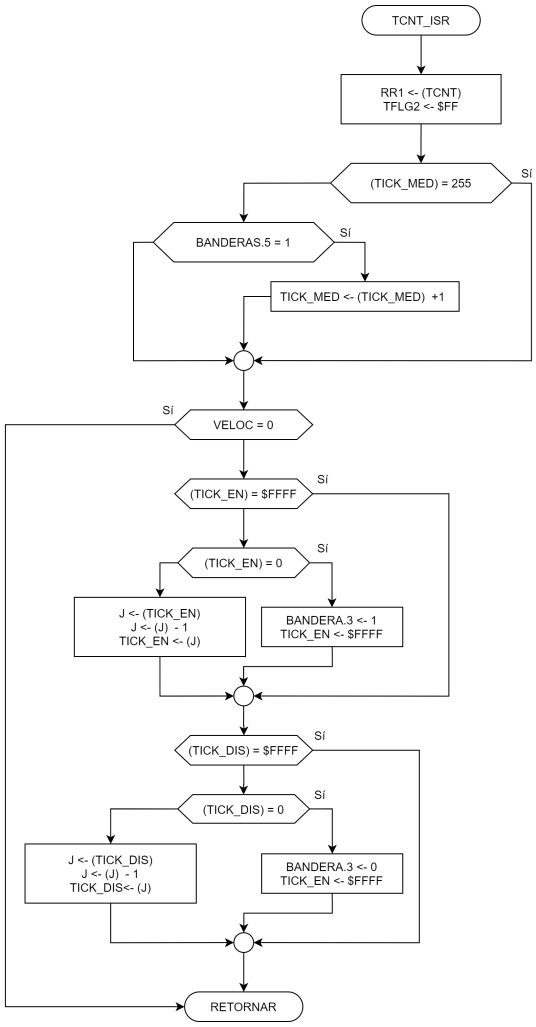

Figura 8: Diagrama de flujo de la subrutina de interrupción **TCNT_ISR** donde se utiliza el timer overflow para calcular los diferentes tiempos que cada pantalla debe durar en Modo Competencia.

2.5. Modo Configuración

A continuación se describen las subrutinas que componen el Modo Configuración, así como las consideraciones que se tuvo en su diseño.

2.5.1. MODO_CONFIGURACION

Modo de operación del sensor, en este modo se configura el número de vueltas del sensor. Para esto se debe verificar que el valor ingresado en el teclado sea valido es decir entre el rango de 5 y 25 vueltas. Si no es valido se borra, caso contrario se muestra en pantalla.

Entradas

- **ValorVueltas:** Número de vueltas a ingresar y validar.

Salidas

- **BIN1:** Valor en binario a mostrar en los displays de 7 segmentos 1 y 2. En este caso BB para que se apaguen.
- **BIN2:**Valor en binario a mostrar en los displays de 7 segmentos 3 y 4.

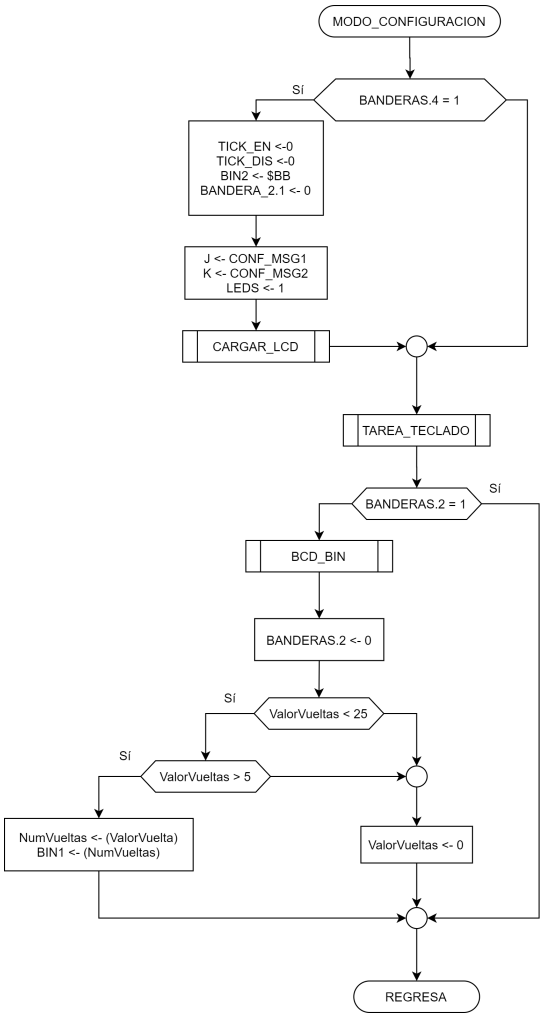

Figura 9: Diagrama del flujo para el modo configuración, este llama a Tarea_Teclado para recibir datos del teclado matricial.

2.5.2. BCD_BIN

Conversión de BCD a binario, utilizada para convertir el valor del arreglo del teclado.

Entradas

- **NUM_ARRAY:** Arreglo de numero en BCD digitados en el teclado.

Salidas

- **ValorVueltas:** Valor en binario resultante de la conversión, corresponde al limite de vueltas.

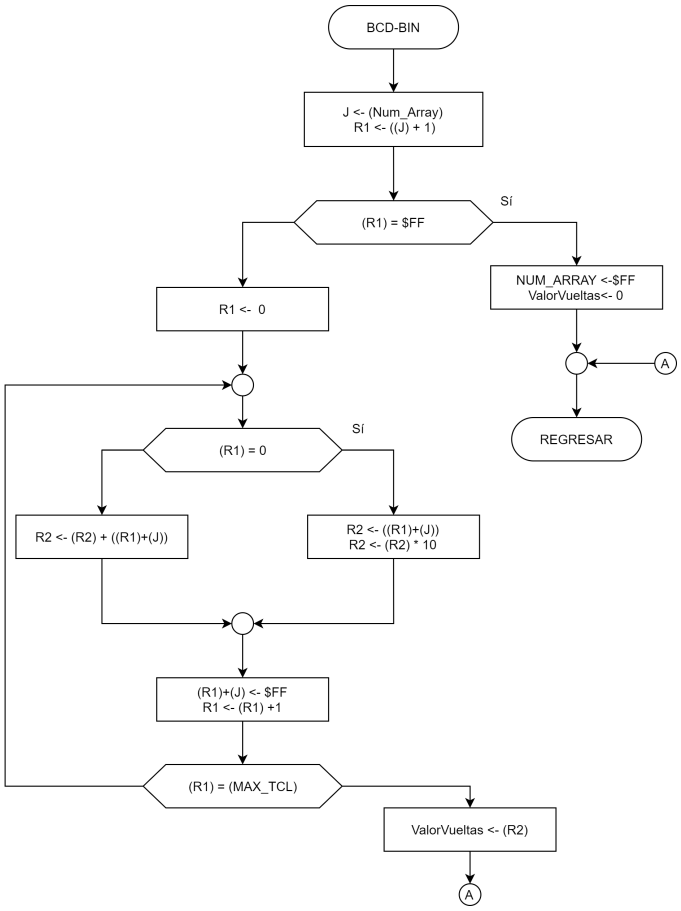

Figura 10: Diagrama del flujo del valor leído de **Num_Array** para transformarlo en **ValorVueltas**

2.5.3. Tarea Teclado

Subrutina encargada de manejar el teclado matricial. Llama y verifica los valores de **Mux_Teclado** para luego llamar a **Formar_Array**.

Entradas

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

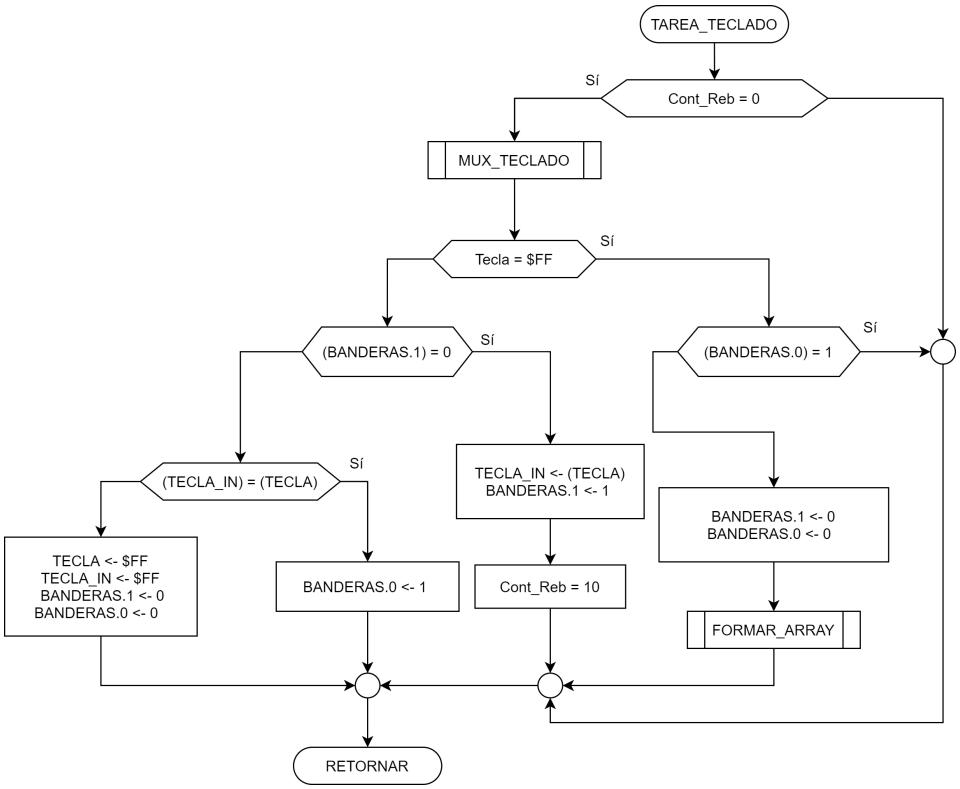

Figura 11: Diagrama del flujo de la subrutina **TAREA_TECLADO**.

2.5.4. Mux_Teclado

Subrutina encargada de capturar el valor presionado en el teclado matricial,mediante un patrón al puerto A, y se detectando la señal de entrada correspondiente al valor de una tecla. Se debe considerar que no se utiliza la columna 1 conforme a la Tarea 4.

Entradas

- **TECLAS:** Tabla que incluye los valores de cada una de las teclas validas.

Salidas

- **TECLA:** Valor de la tecla presionada.

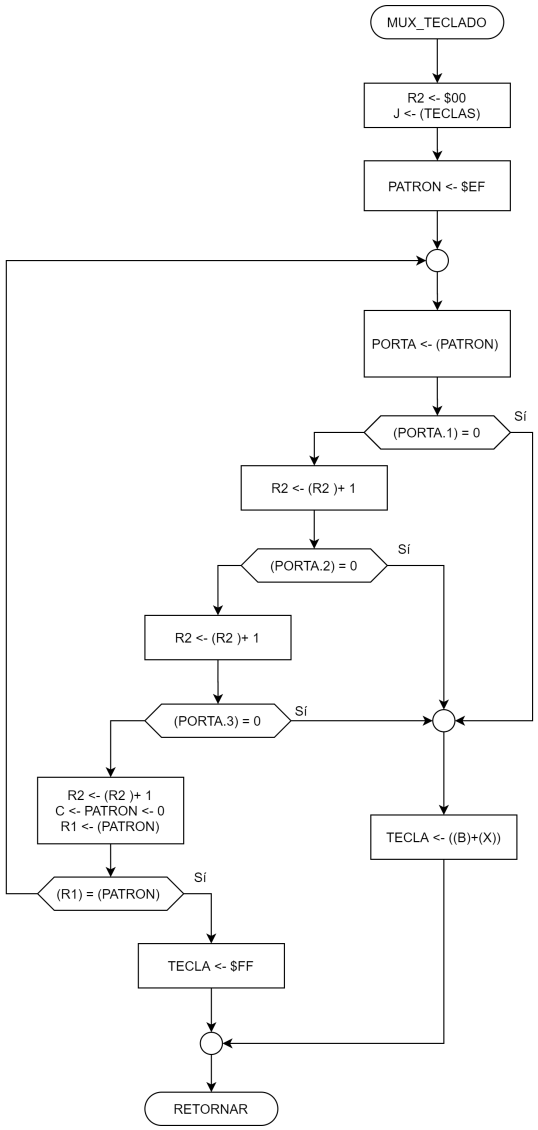

Figura 12: Diagrama de Mux_Teclado donde se muestra la lectura a partir del bit 1 en **PORTA**, omitiendo así la primera columna.

2.5.5. Formar_Array

Almacena los valores del teclado en el arreglo **Num_Array** cuando los valores de **TECLA_IN** y **TECLA** son iguales. La cantidad máxima de teclas se define por **MAX_TCL**, además al presionar la tecla enter (D en el teclado matricial) se guarda el valor. También al presionar la tecla B (0 en el teclado matricial) se elimina un valor antes de guardar.

Entradas

- **MAX_TCL:** Cantidad máxima de teclas en el arreglo.
- **TECLA_IN:** Valor de la tecla presionada.
- **CONT_TCL:** tecla actual, funciona como puntero.

Salidas

- **NUM_ARRAY:** Valor de la tecla presionada.

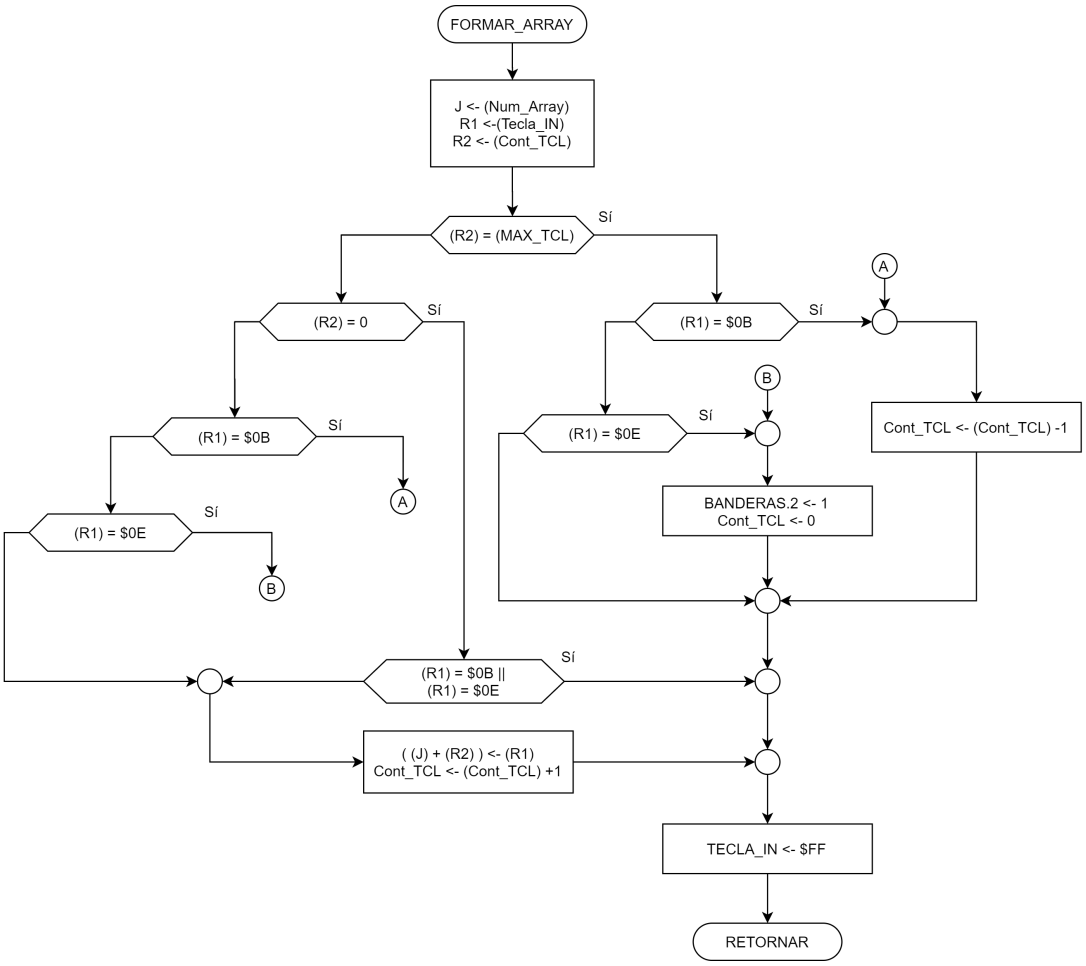

Figura 13: Diagrama de la subrutina **Formar_Array** donde se almacena el valor digitado en **NUM_ARRAY** antes de convertirse en **ValorVueltas**.

2.6. Modo Competencia

Seguido se describen las subrutinas que componen el Modo Competencia, así como las consideraciones que se tuvo en su diseño.

2.6.1. MODO_COMPETENCIA

Modo de operación del sensor, en donde se calcula la velocidad del ciclista mediante los botones SW2 y SW5, y llama a la subrutina de control **PANT_CTRL** para mostrar diferentes mensajes en la pantalla.

Entradas

- **BANDERAS_2.0:** Indica cuando imprimir el mensaje *"Calculando..."* .
- **Veloc:** Velocidad de la bicicleta, recibida de **CALCULAR_ISR**.

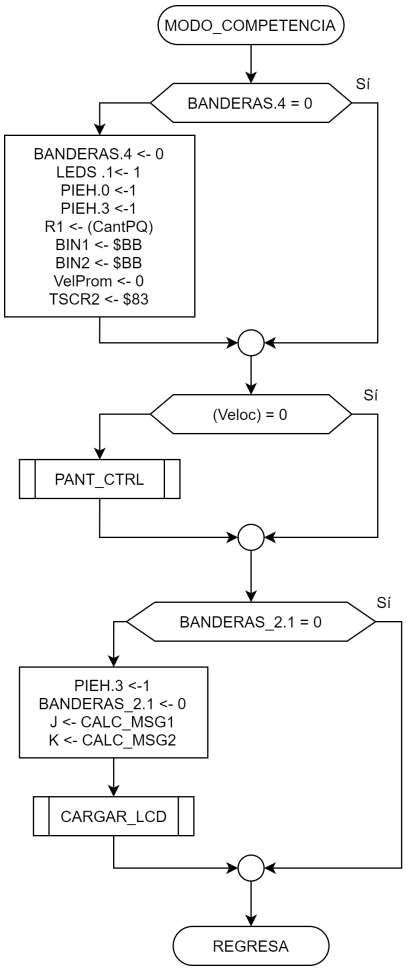

Figura 14: Diagrama del flujo del Modo Competencia. Este se encarga de mostrar el mensaje *Calculando...*, así como de llamar a **Pant_Ctrl**.

2.6.2. Pant_CTRL

Subrutina que modifica las pantallas cuando se detecta el paso de una bicicleta, si la velocidad se encuentra dentro de los limites. si se sale se muestran guiones acompañados de un mensaje de alerta. Si es un valor valido se muestra la velocidad cuando el vehículo pasa 100 m después del segundo sensor. Para calcular el tiempo de espera entre el mensaje *"Calculando..."* y el mostrar resultado se utiliza la siguiente ecuación, considerando $T_{tick} = 21,85ms$:

$$TICK\_DIS = \frac{200}{(VelProm \cdot T_{tick})} \cdot \frac{3600}{1000} \implies TICK\_DIS = \frac{32952}{VelProm} \tag{6}$$

Entradas

- **VelProm:** Promedio de velocidades de la bicicleta, calculata a traves de varias vueltas en **CALCULAR_ISR**.
- **Vueltas:** Numero de vueltas, con el limite configurado en Modo Configuración.

Salidas

- **BIN1:** Se se apaga ($BB) o muestra guiones ($AA) cuando la bici pasa los sensores.
- **BIN2:** Se envía el valor de la velocidad alcanzada, se apaga ($BB) o muestra guiones ($AA) cuando la bici pasa el sensor.

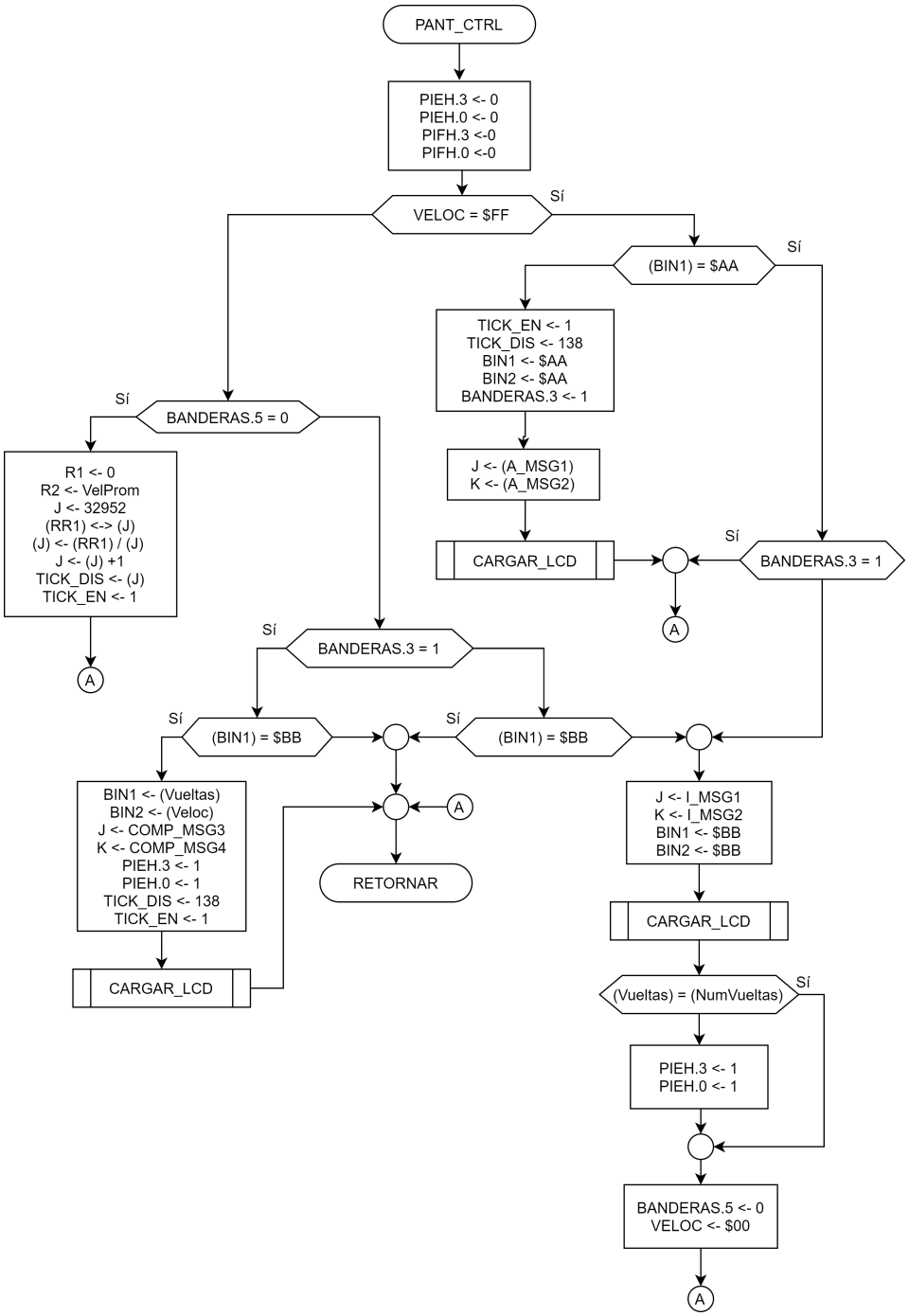

Figura 15: Diagrama de la subrutina **Pant_CTRL**, en la cual se manejan los mensajes vistos en la pantalla LCD del Modo Competencia.

2.7. Modo Resumen

El modo Resumen es una sola rutina, la cual se describe a continuación:

2.7.1. MODO_RESUMEN

Modo de operación del sensor, en donde se despliega la velocidad promedio de las vueltas detectadas con **VelProm** en el Modo Competencia, así como la cantidad de vueltas realizadas a través de la variaable **Vueltas**. Además se desactivan las interrupciones.

Entradas

- **VelProm:** Promedio de velocidades de la bicicleta, calculada a través de varias vueltas en **CALCULAR_ISR**.
- **Vueltas:** Numero de vueltas, con el limite configurado en Modo Configuración.

Salidas

- **BIN1, BIN2:** Se borran de la pantalla de 7 segmentos al escribirles $BB.

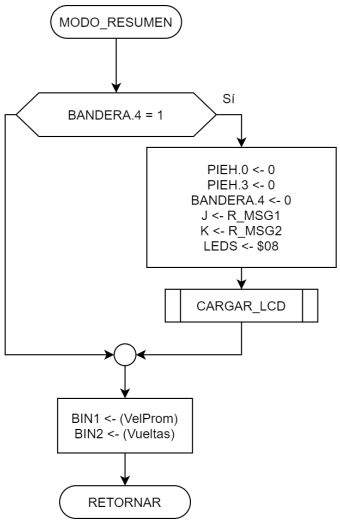

Figura 16: Diagrama del **MODO_RESUMEN** donde se despliega un resumen de los resultados logrados en el Modo Competencia.

2.8. Modo Libre

Igual que el Modo Resumen, el Modo Libre se compone de una sola subrutina, descrita abajo:

2.8.1. MODO_LIBRE

Modo de operación del sensor, en donde no se realiza calculo u operación. Además se borran los valores desplegados en la pantalla de 7 segmentos y deshabilitan interrupciones en PTIH.

Salidas

- **BIN1, BIN2:** Se borran de la pantalla de 7 segmentos al escribirles $BB.

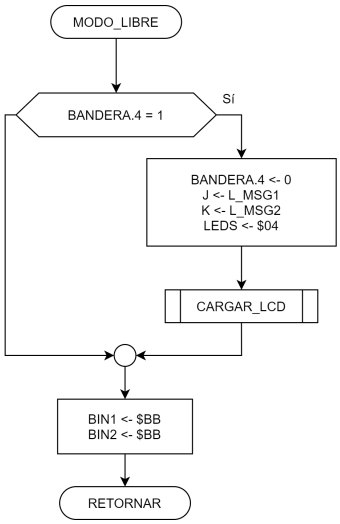

Figura 17: Diagrama del **MODO_LIBRE** donde se despliega un mensaje mientras el sistema se encuentra ocioso.

2.9. Otras Subrutinas

Estas son subrutinas de uso común a lo largo de todo el programa, independientemente del modo.

2.9.1. Cargar_LCD

Subrutina encargada de enviar la información a desplegar en la pantalla LCD

Entradas

- **ADD_L1, ADD_L2:** Constantes que representan comandos para añadir lineas.
- **D60us, D2ms:** Constantes para el tiempo de espera para comunicarse con la pantalla LED.
- **initDisp:** Tabla de comandos para iniciar la comunicación con la pantalla.
- **J, K:** Registros de índices e la arquitectura, Contienen los punteros de las lineas 1 y 2 a mostrar en las pantallas respectivamente.

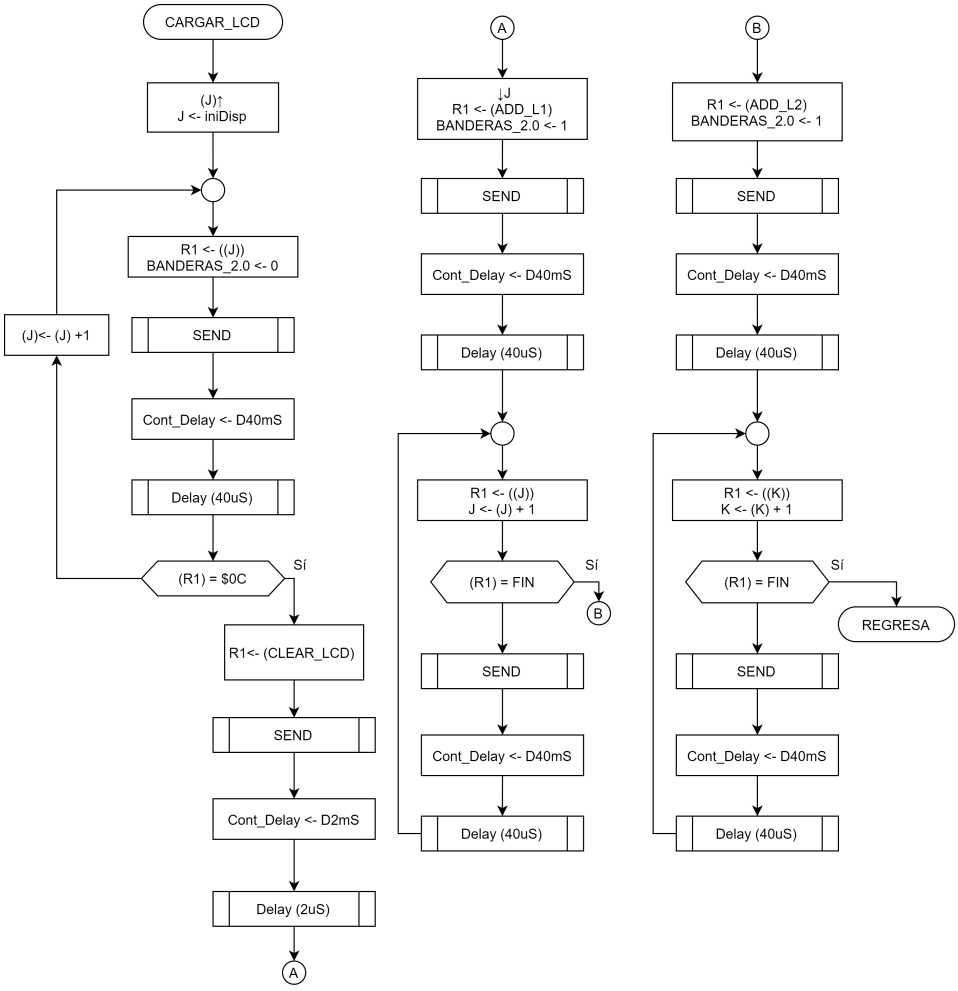

Figura 18: Diagrama de flujo para la subrutina que carga los mensajes a la pantalla LCD.

2.9.2. Send

Se encarga de enviar a la pantalla LCD el dato o comando que recibe conforme al estado de la bandera en **BANDERAS_2.0: SEND_DATA**.

Entradas

- **BANDERAS_2.0:** Indica si se envía un comando o datos.
- **D240ms:** Constante con el delay necesario para la pantalla.

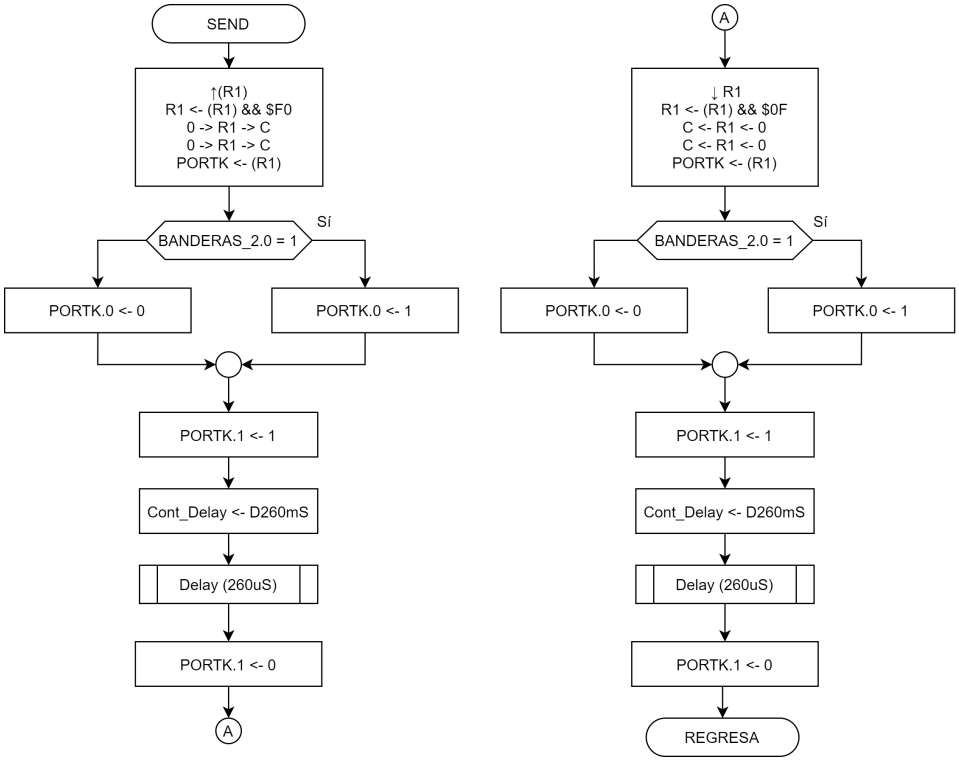

Figura 19: Diagrama de flujo para la subrutina que controla el envío de mensajes o comandos a la pantalla LED conforme a **BANDERAS_2.0**.

2.9.3. BCD_7SEG

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

Entradas

- **BC1,BCD2:** Valores en BCD a convertir.
- **SEGMENT:** Tabla que contiene los valores para cada uno de los dígitos, guiones y valores en blanco.

Salidas

- **DISP1,DISP2,DISP3,DISP4:** Valores en 7 segmentos para cada uno de los display.

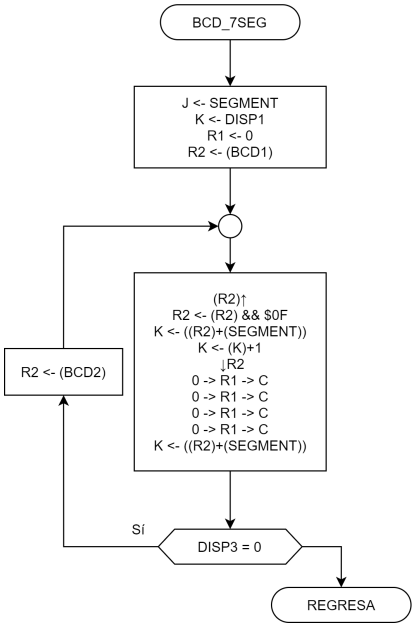

Figura 20: Diagrama de la subrutina **BCD_7SEG** donde se distribuyen los valores a desplegar en la pantalla de 7 segmentos.

2.9.4. Delay

Subrutina de atraso, consume el tiempo necesario para la comunicación con la pantalla LCD.

Entradas

- **CONT_DELAY:** Contador que indica cuanto esperar (si Cont_Delay es 1 = 230µs).

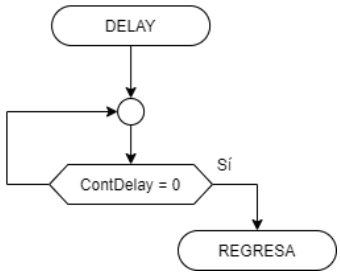

Figura 21: Diagrama de la subrutina **Delay**, donde se consume el tiempo requerido en la variable **Cont_Delay**.

3. Conclusiones y Recomendaciones

Considerando los resultados obtenidos en el proyecto, se toman las siguientes conclusiones:

3.1. Conclusiones

- Al correr el presente diseño en la tarjeta Dragon 12+ se logra cumplir todos los requisitos establecidos previamente.
- Los diseños presentan una solución viable al problema de los sensores requeridos en el velódromo presentado en la figura 1.
- La modificación en el diseño de la subrutina en **MODO_CONFIG** conforme a la figura 14.

3.2. Recomendaciones

- Antes de codificar es sumamente útil desarrollar los diseños, aún si estos se modifican, pues estos ofrecen una lógica que guía y simplifica el proceso de codificación.
- Es sumamente útil anotar el nombre y funcionamiento de cada bandera, para evitar errores, así como para referirse a su estado a la hora de utilizar el Debug12 para depurar.
- Se recomienda trabajar con control de versiones (por ejemplo git), y depurar un modo a la vez para simplificar la búsqueda de errores.
- Finalmente se recomienda probar el modo competencia en un intervalo entre 2 y 5 segundos, pues corresponde aproximadamente a los valores de las velocidades limite ($V_{MAX} = 95\frac{km}{h}$ y $V_{MIN} = 35\frac{km}{h}$)

Referencias

[1] Geovanny Delgado. Proyecto final, runmeter623. *Microprocesadores IE-623*, 2020.

[2] Freescale HCS12 microcontroller family. User's manual. *Dragon12-Plus-USB Trainer*, 2005.

4. Anexo: Estructuras de datos solicitadas

ESTRUCTURAS DE DATOS - RunMeter 623

VARIABLES				BANDERAS ($1000)		TABLAS
SUBRUTINAS	NOMBRE	TAMAÑO (BYTES)	POSICION	BANDERA	BIT	NOMBRE
MODO CONFIG	NumVueltas	1	$1001	TCL_LISTA	0	TECLAS ($1040)
	ValorVueltas	1	$1002	TCL_LEIDA	1	SEGMENT ($1050)
TAREA TECLADO:	MAX_TCL	1	$1003	ARRAY_OK	2	InitDisp ($1060)
	Tecla	1	$1004	PANT_FLAG	3	Inicio Mensajes ($1070)
	Tecla_IN	1	$1005	CALC_TICKS	5	
	Cont_Reb	1	$1006			
	Cont_TCL	1	$1007			
	Patron	1	$1008			
	Num_Array	2	$1009			
ATD_ISR	BRILLO	1	$100B			
	POT	1	$100C			
PANT_CTRL	TICK_EN	2	$100D			
	TICK_DIS	2	$100F			
CALCULAR	Veloc	1	$1011			
	Vueltas	1	$1012			
	VelProm	1	$1013			
TCNT_ISR	TICK_MED	2	$1014			
CONV_BIN_BCD	BIN1	1	$1016			
	BIN2	1	$1017			
	BCD1	1	$1018			
	BCD2	1	$1019			
BIN_BCD	BCD_L	1	$101A			
	BCD_H	1	$101B			
	TEMP	1	$101C			
	LOW	1	$101D			
BCD_7SEG	DISP1	1	$101E			
	DISP2	1	$101F			
	DISP3	1	$1020			
	DISP4	1	$1021			
OC4_ISR	LEDS	1	$1022			
	CONT_DIG	1	$1023			
	CONT_TICKS	1	$1024			
	DT	1	$1025			
	CONT_7SEG	2	$1026			
RTI_ISR	CONT_200	1	$1028			
SUBRUTINAS LCD	Cont_Delay	1	$1029			
	D2mS	1	$102A			
	D240uS	1	$102B			
	D60uS	1	$102C			
	Clear_LCD	1	$102D			
	ADD_L1	1	$102E			
	ADD_L2	1	$ 102F			
DISPONIBLES	A DEFINIR	4	$1030-$1034			

Figura 22: Tabla con las estructuras de datos solicitadas. obtenido de [1]