



**PROYECTO FINAL.
RunMeter 623**

1. Descripción de la Aplicación.

Se desea diseñar e implementar un sistema para el despliegue de información en un velódromo. El sistema, denominado RunMeter 623, cuenta con dos sensores fotoreflectivos de presencia, una pantalla de 4 dígitos de 7 segmentos y una pantalla LCD 2 x16, como se muestra en la Figura #1. En esta figura se puede observar que el sistema dispone de dos sensores fotoreflectivos de presencia para detectar las bicicletas, separados **55 metros entre sí** y de una **pantalla, localizada a 300 metros del sensor S2**, para presentar la velocidad de la bicicleta calculada en Km/h y el número de vueltas recorridas.

Los sensores emiten una señal infrarroja en forma transversal a la pista, a un receptor que se encuentra al otro lado de la vía. Cuando pasa un ciclista, se interrumpe el haz en el primer sensor y se envía una señal al controlador. Cuando se pase el primer sensor se inicia el conteo del tiempo que requiere el ciclista para alcanzar el segundo sensor. Con este tiempo y con la distancia entre los sensores, se realiza un cálculo de la velocidad promedio del ciclista, en Km/h. Esta velocidad se presentará al ciclista en la pantalla.

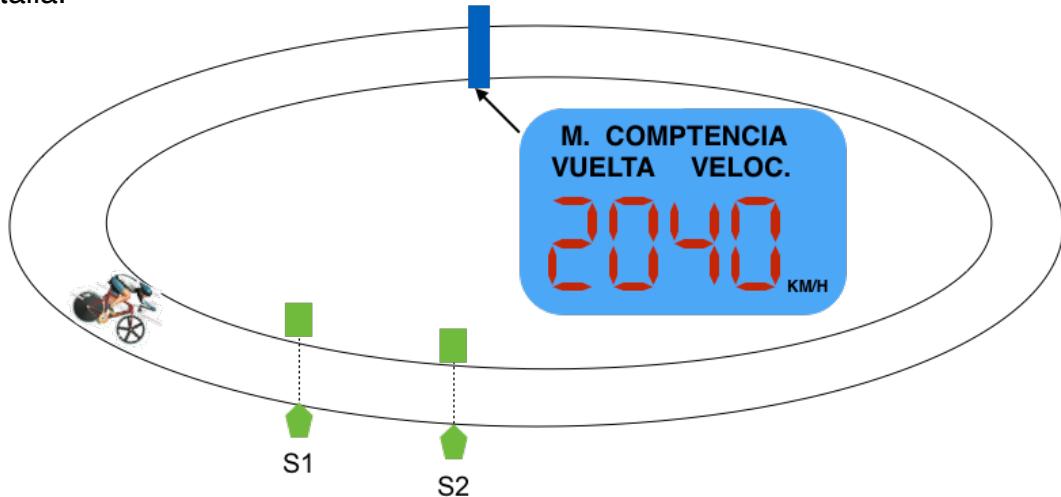


Figura #1
Imagen descriptiva de la operación del RunMeter 623

El RunMeter 623 tendrá cuatro modos de operación: Libre, Competencia, Configuración y Resumen. Estos modos se seleccionarán por medio de los **dos interruptores denominados Modo**. Los modos de operación del RunMeter 623 son excluyentes, es decir, solo se puede estar en un modo a la vez.



El RunMeter 623 tiene una Consola de Control, que incluye un display de 4 dígitos de 7 segmentos, una pantalla LCD 2 x16, un teclado de 12 botones, 4 LEDs de indicación de estado y dos interruptores para selección de estado. La distribución del panel de control se muestra en la Figura #2.



Figura #2
Distribución del panel de control del RunMeter 623

Las funciones de los modos de operación se describen a continuación.

i. **Modo LIBRE.**

En el modo libre el sistema está ocioso y no realizará ningún cálculo. En este modo el sistema desplegará el siguiente *Mensaje Modo Libre* en la pantalla LCD.



La pantalla de **7 segmentos** debe estar apagada y todas las interrupciones deben quedar deshabilitadas, luego de desplegar el mensaje.



ii. Modo CONFIGURACIÓN.

En el Modo Configuración es posible programar la cantidad de vueltas que será procesada por el RunMeter 623 para un ciclo de competencia. La cantidad de **vueltas solo podrá estar entre 5 y 25**. Dicha cantidad de vueltas deberá ser ingresada por medio del teclado del RunMeter 623 y se almacenará en la variable NumVueltas. Mientras el RunMeter 623 permanezca en el Modo Configuración, pueden ser realizados sucesivos cambios a **NumVueltas**. Cada cambio será reconocido y aceptado cuando se presiona la tecla E.

En el Modo Configuración el valor **NumVueltas** programado deberá aparecer en la pantalla de 7 segmentos en **DSP3-DSP4**, en todo momento, hasta que se salga del Modo Configuración. Cuando se vaya a **cambiar esta variable se debe sobreescribir sobre el valor programado**. Los **DISP1-DISP2** deben permanecer apagados.

Además en la pantalla LCD se debe mostrar el *Mensaje Modo Configuración*:



iii. Modo COMPETENCIA.

Al ingresar al modo competencia se debe desplegar en el LCD el *Mensaje Inicial*:



Los display de **7 segmentos** debe iniciar apagados.

En el Modo Competencia se debe **calcular** la velocidad de un ciclista con base en el tiempo transcurrido entre la detección del ciclista por los sensores y se debe desplegar su valor en la pantalla donde el competidor pueda leerla. Para esto la **velocidad deberá ser desplegada en la pantalla 100 metros antes de que el ciclista esté debajo de la misma** y **hasta que este la haya superado**. La estimación de cuando el ciclista se



encuentra a los 100 metros y de cuando superó la pantalla, se hará basándose en la velocidad promedio calculada.

Junto con la velocidad se deberá desplegar el **acumulado de vueltas** recorridas por el ciclista, basado en la detección de su presencia por medio del **sensor fotoreflectivo S1**. La velocidad y el acumulado de vueltas se debe mostrar en la pantalla de 7 segmentos. El **acumulado a la izquierda (DSP1:DSP2)** y la **velocidad promedio a la derecha (DSP3:DSP4)**. Mientras tanto en la pantalla LCD se debe mostrar **Mensaje de Competencia**:



Se va a considerar que el velódromo será utilizado por solo un ciclista a la vez, de tal manera que la lectura de los **sensores debe estar inactiva desde que el ciclista pasa enfrente de ellos y hasta que se termina de desplegar la velocidad en la pantalla**. La pantalla de 7 segmentos deberá apagarse luego de que se termina de presentar la velocidad al ciclista y mientras se le calcula su velocidad en la siguiente vuelta.

Superada la pantalla por el ciclista y mientras no se le detecte por S1 el LCD debe mostrar el **Mensaje Inicial**.

Cuando se **detecte el ciclista en S1 y hasta que se deba mostrar el Mensaje de Competencia** en la Pantalla LCD se deberá mostrar el **Mensaje Calculando**:



No se tomará en cuenta, el paso de ciclistas en el sentido opuesto. La velocidad mínima leída será **V_MIN=35 Km/h** y la velocidad mayor será de **V_MAX= 95 Km/h**. Si algún ciclista pasa a una velocidad menor a V_MIN o mayor a V_MAX se presentará en la pantalla de **7 segmentos 4 guiones - - -** y el **Mensaje de Alerta** en el LCD, durante **tres segundos**, al momento en que se determine que la velocidad está fuera de rango.



El *Mensaje de Alerta* es el siguiente:



Para velocidades fuera de rango no se debe incrementar el acumulado de vueltas. Inmediatamente después de desplegar el *Mensaje de Alerta*, el RunMeter 623 deberá quedar habilitado para leer los sensores, calcular la velocidad e incrementar el acumulado de vueltas a desplegar.

El RunMeter 623 calculará la velocidad del ciclista una cantidad definida de veces denominada NumVueltas que es programable. Al cabo de esta cantidad de vueltas el RunMeter 623 debe deshabilitar los sensores y no calcular más veces la velocidad promedio. La última vuelta se completará cuando se haya terminado de desplegar el *Mensaje de Competencia* en el LCD, dicho mensaje deberá ser cambiado por el *Mensaje Inicial*. El operador deberá cambiar de modo y luego regresar a Modo Competencia si desea iniciar un nuevo ciclo de mediciones.

Justo antes de iniciar el cálculo de la velocidad (con la detección del ciclista por S1), incluso antes de que el acumulador de vueltas alcance el valor programado en NumVueltas, es posible suspender el Modo Competencia pasando con los interruptores a cualquier otro modo. En este caso todos los valores calculados deben mantenerse hasta que se reingrese al Modo Competencia.

Cada vez que el RunMeter 623 pase de Modo Competencia a cualquier otro modo se debe suspender cualquier cálculo en curso. Además al **ingresar a Modo Competencia se deben borrar todas las variables de cálculo que se utilizan en este modo.**

iv. Modo RESUMEN.

En el Modo Resumen el RunMeter 623 deberá desplegar la velocidad **promedio** del ciclo de competencia, calculada como el valor promedio de las velocidades de todas las vueltas en la cantidad de **vueltas** realizadas. Estos valores se desplegarán en la pantalla de 7 segmentos en el orden habitual. Además en la pantalla LCD se debe desplegar el *Mensaje Resumen*:



Nota: Con el fin de que la variable VelProm sea de 1 byte la velocidad promedio debe ser actualizada en cada vuelta utilizando el contador de vueltas y la velocidad calculada.

3. Estructura de la aplicación.

3.1. Asignación de Hardware.

- i. Se utilizará la pantalla de 7 segmentos de la Dragon 12+, como display de Vueltas:Velocidad del RunMeter 623, cumpliendo todo lo especificado en la Tarea de Pantallas.
- ii. Se utilizará la pantalla LCD de la Dragon 12 como la pantalla de mensajes del RunMeter 623 cumpliendo todo lo especificado en la Tarea de Pantallas.
- iii. Se utilizarán los leds del puerto B, como los leds de estado del RunMeter 623, la asignación de los leds es la siguiente: PB3: Resumen, PB2=Competencia, PB1= Configuración, PB0 = Libre. Los restantes Leds deben permanecer apagados y solo el led correspondiente al modo en que se encuentre el RunMeter 623 deberá estar activo.
- iv. El teclado se va a implementar con la matriz de botones de la Dragon 12+ conectados al puerto A, cumpliendo todas las condiciones de la Tarea de Teclados Matriciales.
- v. Se utilizarán las botones conectados al puerto H para implementar los sensores S1 y S2 del RunMeter 623. La asignación es la siguiente: PH3=S1, PH0=S2. Los botones serán atendidos por interrupción.
- vi. Se utilizará el potenciómetro trimmer ubicado en el PAD7 como control de brillo de la pantalla de 7 segmentos.
- vii. Se utilizarán los interruptores PH7-PH6 como selectores de modo, con la siguiente distribución:
 - PH7-PH6: OFF-OFF : Modo Libre
 - PH7-PH6: OFF-ON : Modo Configuración
 - PH7-PH6: ON-OFF : Modo Resumen
 - PH7-PH6: ON-ON : Modo Competencia.



3.2. Arquitectura del programa.

1. El programa debe tener un encabezado haciendo una explicación general del mismo, **listando todas las subrutinas que incluye**. El programa deberá incluir un bloque de definición de variables, constantes, arreglos y tablas que utilice.
2. Se debe incluir un bloque con la relocalización de todos los vectores de interrupción que se utilicen.
3. Se debe incluir un bloque de programa con la declaración de todas las estructuras de datos a utilizarse. En cada línea se debe indicar para qué se utiliza la estructura de datos declarada.
4. El programa debe incluir una **Rutina de Inicialización, con un bloque de programa que inicie todos los elementos de hardware dentro del S12 y un bloque de programa para inicializar todas las estructuras de datos que lo requieran**. Cualquier otra cosa que deba ser iniciada, deberá incluirse dentro de esta rutina.
5. En todo momento el patrón de encendido de los diodos leds estará almacenado en una variable denominada LEDS.
6. Tablas: El programa manejará dos tablas definidas de la siguiente manera:
 - a. **TECLAS:** El programa utilizará una tabla de dimensión 4x3 llamada TECLAS con los valores asignados a cada una de las teclas utilizadas en la aplicación, tal y como se describe en la tarea de Teclados Matriciales.
 - b. **SEGMENT:** Esta tabla contendrá el patrón de 7 segmentos asociados al valor BCD a desplegar en la pantalla, en orden ascendente para lograr acceder a ella por direccionamiento indexado y se usará de acuerdo a la especificación de la Tarea de **Manejo de Pantallas**. Esta tabla tendrá en sus primeras 10 posiciones **los patrones correspondientes a los números del 0 al 9, en la posición 10 (\$A) deberá tener el patrón correspondiente al dígito g (guion) y en la posición 11 (\$B) el patrón correspondiente a dígito apagado**.
7. El proyecto en su totalidad debe apegarse a la arquitectura de software descrita en este documento. Solo se revisaran proyectos implementados con esta arquitectura.
8. Subrutinas. La aplicación desarrollada deberá incluir las siguientes subrutinas (las subrutinas marcadas con * ya han sido desarrolladas en las tareas y se incluye su descripción solo como referencia o cuando se deba realizar alguna modificación):
 - i. **SUBRUTINA CONFIG (*):** Esta subrutina es llamada desde el programa principal, inicialmente se debe colocar el valor de **NumVueltas en BIN1** para que despliegue su valor en la pantalla. Al ingresar a la subrutina Modo Config, Array_OK debe estar en cero y se debe llamar a la subrutina Tarea_Teclado y retornar. Cuando en un llamado se encuentre Array_OK=1, la subrutina Config llama a la subrutina BCD_BIN y luego valida que ValorVueltas esté en el intervalo válido. De ser así borra la bandera Array_OK, coloca el valor de



ValorVueltas en NumVueltas y este a su vez en BIN1 para que el nuevo valor sea desplegado en la pantalla y retorna. Si el valor de ValorVueltas no está en el intervalo válido, entonces borra Array_OK, Num_Array y ValorVueltas y retorna para volver a iniciar la lectura del teclado. Esta subrutina devuelve al programa principal el valor de NumVueltas a ser utilizado en el Modo Competencia. Observe que **NumVueltas debe estar en cero** después del encendido. Mientras el sistema esté en el Modo Configuración se podrá modificar NumVueltas todas las veces que sea necesario.

- ii. **SUBRUTINA BCD_BIN (*)**: Esta subrutina toma el valor en Num_Array, lo convierte a binario y lo coloca en la variable ValorVueltas.
- iii. **Subrutina RTI_ISR (*)**: Esta aplicación va a manejar una interrupción RTI para la supresión de rebotes, esta interrupción se debe generar con un periodo de 1 mS. Para ello se utiliza una variable Cont_Reb para suprimir los rebotes del teclado matricial y de los botones PH3-PH0. Además las subrutina RTI debe escribir el registro ATD0_CTL5 para dar inicio a las conversiones del ATD. Esta tarea debe realizarse cada 200 mS. Para ello se utilizará una constante denominada CONT_200.
- iv. **TAREA_TECLADO (*)**: Esta subrutina será la encargada de llamar a la subrutina MUX_TECLADO para que capture una tecla presionada. Además esta subrutina realizará las acciones para suprimir los rebotes y para definir el concepto de tecla retenida, leyendo la tecla hasta que la misma sea liberada. En esta subrutina se carga el Cont_Reb cuando se detecta una tecla presionada, se valida si la tecla presionada es válida, **comparando dos lecturas de la misma luego de la supresión de rebotes**. Finalmente la Tarea_Teclado debe llamar a la subrutina FORMAR_ARRAY cuando determine que una tecla ha sido leída de manera correcta (TCL_LISTA =1). Se debe usar la implementación de la Subrutina TAREA_TECLADO de la Tarea de Teclados Matriciales.
- v. **Subrutina MUX_TECLADO (*)**: Esta subrutina es la encargada de leer el teclado propiamente. El teclado matricial deberá leerse de manera iterativa enviando uno de los **4 patrones (\$EF, \$DF, \$BF, \$7F)** al puerto A. Se tendrá un índice denominado **PATRON** que terminará el ciclo **FOR-NEXT** cuando su valor supere 4. Para la lectura del teclado NO lea patrones, en su lugar busque cuál bit de la parte baja del puerto A está en cero para identificar la tecla presionada, de esta manera solo hay 3 posibilidades. El valor de la tecla presionada deberá ser devuelto, al procedimiento que ha llamado esta subrutina, por medio de la variable Tecla. Esta subrutina no recibe ningún parámetro. **Se debe usar la implementación de la subrutina MUX_TECLADO de la Tarea de Teclados Matriciales.**



asumo que es igual

- vi. **Subrutina FORMAR_ARRAY (*):** Esta subrutina recibe el valor de la tecla presionada válida en la variable Tecla_IN. Además cuenta con el valor de la constante que define cuál es la longitud máxima de la secuencia de teclas almacenado en MAX_TCL, esta constante podrá tener un valor de 2. La subrutina debe colocar de manera ordenada los valores de las teclas recibidas en Tecla_IN en un arreglo denominado Num_Array. La subrutina utilizará una variable llamada Cont_TCL para almacenar el número de tecla en Num_Array. Este arreglo debe ser accesado por direccionamiento indexado por acumular B (cargando en B el contenido de Cont_TCL). Cada vez que se ingrese a FORMAR_ARRAY se debe validar primero si se alcanzó MAX_TCL; de ser así se valida si la nueva tecla recibida en Tecla_IN es \$0E (Enter) en cuyo caso se hace ARRAY_OK =1 indicando que se finalizó el Num_Array y se repone Cont_TCL=0, para que quede listo para una nueva secuencia de entrada. Si lo que se recibió en Tecla_IN es \$0B se deberá poner \$FF en la actual posición de Num_Array y descontar Cont_TCL, solo si este no es cero, para que en la próxima iteración (nuevo ingreso de una tecla) esta sea almacenada en la posición anterior (función de borrado). Lo indicado, respecto a recepción en FORMAR_ARRAY de una tecla E o B, aplica en cualquier momento que se reciba una tecla en Tecla_IN, excepto con la primera tecla, pues si se recibe como primera tecla \$0B o \$0E estas deben ser ignoradas. Debe recordarse que cuando Cont_TCL alcance el valor de MAX_TCL las únicas teclas válidas a procesar son E y B, cualquier otra tecla presionada debe ser ignorada. Finalmente debe notarse que la secuencia ingresada se termina con una tecla E y su longitud puede ser cualquiera entre 1 y MAX_TCL. Además cuando se termina la secuencia de teclas se pone en 1 la bandera ARRAY_OK. Se debe usar la implementación de la subrutina FORMAR_ARRAY de la Tarea de Teclados Matriciales.
- vii. **Subrutina ATD_ISR:** El convertidor analógico/digital será utilizado para leer el valor del potenciómetro ubicado en PAD7 de la tarjeta Dragon 12+, como control de brillo de la pantalla. Se debe programar el convertidor para realizar 6 conversiones de este canal, con 4 periodos de reloj para el muestreo, en 8 bits sin signo y en la frecuencia de operación de 600 KHz. El valor de las lecturas será capturado por interrupción. La subrutina ATD_ISR deberá calcular el promedio de las 6 lecturas y colocarlo en la variable POT. A partir de POT se debe obtener la variable BRILLO, misma que maneja el valor de K, para el control de brillo de la pantalla. Dado que el brillo de la pantalla solo puede ser ajustado con una resolución de 20 pasos, según lo descrito en la Tarea de Pantallas, entonces el valor de BRILLO se calcula como una progresión lineal del contenido de POT [BRILLO = (20 X POT)/255]. Las conversiones se realizarán cada 200 mS, para ello la subrutina RTI iniciará la secuencia de conversión, como ha sido indicado. Esta fuente de interrupción estará habilitada en todos los modos de operación, excepto en el Modo Libre.



- viii. **Subrutina Competencia:** La subrutina Competencia pone el *Mensaje Inicial* en el LCD y llama a la subrutina PANT_CTRL solo si la velocidad (Veloc) es diferente de cero. Al entrar a esta subrutina se debe poner \$0B en BIN1 y BIN2 para que inicialmente mantenga los display de 7 segmentos apagados. Cada vez que el RunMeter 623 se salga del Modo Competencia se deberá borrar las variables Veloc y CantVueltas.
- ix. **Subrutina PANT_CTRL:** Esta subrutina es llamada desde la subrutina Competencia y será la encargada de calcular el retardo de tiempo, con base en la velocidad calculada, para cambiar el mensaje en la pantalla LCD y la información en la pantalla de 7 segmentos cuando el ciclista esté a 100 metros de la pantalla y luego cuando la haya superado. Para ello la subrutina PANT_CTRL calculará los contadores de ticks necesarios para desplegar el *Mensaje de Competencia* (TICK_EN) y para cambiarlo por el *Mensaje Inicial* luego de que el ciclista supere la pantalla (TICK_DIS). El cálculo de los ticks lo deberá realizar solo la primera vez que entre en la subrutina, para ello se usará una bandera llamada CALC_TICKS. Los valores de TICK_EN y TICK_DIS son decrementados por la subrutina TCNT_ISR. Luego de esto la subrutina PANT_CTRL debe esperar a que, en un nuevo llamado a la misma, PANT_FLAG sea 1 para cambiar el mensaje en la pantalla y desplegar el valor de la velocidad y de CantVueltas, colocando CantVueltas en BIN1 y VELOC en BIN2 para que sean desplegados en los dígitos de 7 segmentos. Finalmente en esta subrutina se determina cuando PANT_CTRL vuelve a ser cero para apagar los dígitos de 7 segmentos **cargando \$BB en BIN1 y BIN2** y desplegar el *Mensaje Inicial* en el LCD.

Una tarea adicional de esta subrutina, la primera que debe realizar, es determinar si el valor de VELOC está fuera de rango ($V_{MAX} < VELOC < V_{MIN}$). Si VELOC está fuera de rango debe poner un valor \$AA en VELOC y en CantVueltas para que se desplieguen los guiones (--) en la pantalla de 7 segmentos y enviar el *Mensaje de Alerta* al LCD, además cargar el TICK_DIS para generar el retardo de 3 segundos y TICK_EN=1 para que despliegue inmediatamente. Cuando TICK_DIS sea cero se debe apagar los display de 7 segmentos, cargando \$BB en BIN1 y BIN2, hacer VELOC = 0 y cambiar en el LCD al *Mensaje Inicial* para que inicie un nuevo ciclo de medición.

Finalmente será en el subrutina PANT_CTRL donde se determine cuando la cantidad de vueltas realizada (Vueltas) alcanza el valor programado en un ciclo de competencia (NumVueltas). En ese momento se deshabilita la lectura de los sensores para terminar el ciclo de competencia.



- x. **Subrutina CALCULAR:** Esta será la subrutina de servicio de interrupciones del puerto H y estará habilitada únicamente en el **Modo Competencia**. Esta subrutina será la encargada de calcular la velocidad del ciclista en Km/H y la cantidad de vueltas. La **interrupción por PH3 (S1)** pondrá la variable **TICK_MED en cero e incrementará la cantidad de vueltas**, mientras que la interrupción por **PH0 (S2)** leerá **TICK_MED** y lo utilizará para calcular la velocidad del ciclista. La velocidad se colocará en una variable llamada **Veloc** y la cantidad de vueltas en una variable denominada **Vueltas**. Además cada vez que se realice un cálculo se debe actualizar la velocidad promedio acumulada **VelProm**. Al darse la interrupción de **PH3** debe enviar al LCD el **Mensaje Calculando**. Esta interrupción solo estará habilitada en el Modo Competencia. **En el encabezado de la subrutina se debe incluir la ecuación utilizada para el cálculo de la velocidad y de la longitud del tubo.**
- xi. **Subrutina TCNT_ISR:** Se va a utilizar la interrupción por Timer **Overflow** del Módulo de Tiempos (TCNT), con un preescalador igual a **8**, para realizar dos tareas:
- Incrementar la variables **TICK_MED** para el cálculo de la velocidad del ciclista, esta variable la calcula la subrutina **CALCULAR**.
 - Control de los tiempos de retardo de cambio de mensajes en la pantalla (tanto LCD como 7 segmentos). Para el control de cambio de mensajes de la pantalla la subrutina debe, en cada interrupción, decrementar las variables **TICK_EN** y **TICK_DIS** si no están en cero. Cuando **TICK_EN** **llegue a cero** será el momento en que se debe desplegar la información en pantalla, entonces la subrutina **TCNT_ISR** debe hacer la bandera **PANT_FLAG=1**. Cuando **TICK_DIS** llegue a cero **TCNT_ISR** pondrá la bandera **PANT_FLAG = 0** nuevamente. La interrupción de rebase del TCNT solo estará habilitada en el Modo **Competencia**.
- xii. **SUBRUTINA CONV_BIN_BCD (*):** Esta subrutina es llamada recurrentemente desde **OC4_ISR** y recibe dos variables denominadas **BIN1** y **BIN2** ambas menores o iguales a 99 y convierte sus valores a BCD, llamando a **BIN_BCD** dos veces. El resultado de la conversión lo devuelve en la variable **BCD1** para **BIN1** y **BCD2** para **BIN2**. La subrutina devolverá un valor de \$B en las posiciones de las variables correspondientes a los dígitos de pantalla que deben permanecer apagados. Si los valores en **BIN1** o **BIN2** son \$BB deberá devolver **BCD1** y/o **BCD2** en \$BB indicando que los dígitos correspondientes deben estar apagados. Si los valores en **BIN1** o **BIN2** son \$AA deberá devolver **BCD1** y/o **BCD2** en \$AA indicando que en los dígitos correspondientes deben aparecer guiones (--).
- xiii. **SUBRUTINA BIN_BCD (*):** Esta subrutina es llamada desde la subrutina **CONV_BIN_BCD** y convierte un número binario recibido por el acumulador A en



el valor correspondiente a BCD, devolviendo su valor en la variable BCD_L. En este caso no se requiere de BCD_H pues el número a convertir es menor que 99. Esta subrutina debe ser implementada con el algoritmo XS3.

- xiv. **SUBRUTINA BCD_7SEG (*)**: Esta subrutina es llamada de manera recurrente desde OC4_ISR y convierte los valores de BCD a 7 segmentos. La subrutina deberá leer en una tabla llamada SEGMENT el patrón de 7 segmentos asociado al valor BCD almacenado en las variables BCD2 y BCD1 y devolverlo al programa principal, por medio de las variables BCD2: DISP1, DISP2 y BCD1: DISP3 y DISP4. Los valores de la Tabla deben ser accesados con direccionamiento indexado, utilizando como offset los valores en las variables BCD2 y BCD1. Esta subrutina deja “cargadas” las variables a ser desplegadas en los display de 7 segmentos.
- xv. **SUBRUTINA OC4_ISR(*)**: Se deberá utilizar la subrutina llamada OC4_ISR desarrollada en la Tarea de Pantallas para que realice las siguientes tareas:
- Recibir el valor en 7 segmentos a ser desplegado (variables DISP1 a DISP4) y la variable LEDS y desplegarlo en la pantalla y el puerto de leds de manera multiplexada, según la técnica de multiplexación vista en clase. La frecuencia de multiplexación deberá ser de 100 HZ/ dígito-Leds. Los contadores CONT_DIG y CONT_TICKS se incrementarán utilizando la interrupción del módulo de tiempos en configuración Output Compare, utilizando el canal 4 a una frecuencia de interrupción de 50 KHz. Consecuentemente el contador CONT_DIG se incrementará cada vez que CONT_TICKS =100. El Ciclo de Trabajo del encendido para los dígitos y el puerto de Leds, deberá ser manejado por medio de una variable llamada DT (DT= N-K). El valor de esta variable funcionará como un control de brillo de la pantalla. Para la determinación de la variable K se utilizará el contador de brillo almacenado en la variable BRILLO.
 - Cada 100 mS (10 Hz) la subrutina OC4_ISR llamará a la subrutina CONV_BIN_BCD primero y luego a la subrutina BCD_7SEG para que actualice los valores a ser desplegados en la pantalla. Para controlar el tiempo de llamado se utilizará una variable tipo word denominada CONT_7SEG.
 - Debe decrementar el Cont_Delay en caso de que este no sea cero.
- xvi. **Subrutina Cargar_LCD(*)**: Esta subrutina debe ser invocada cada vez que se deba actualizar el mensaje a ser desplegado en la pantalla LCD. La subrutina deberá escoger entre los diferentes mensajes (un mensaje para cada línea del



LCD) a ser enviados. Los mensajes a ser desplegados los recibe la subrutina por medio de los índices X y Y, que apuntan a cada uno de los mensajes.

- xvii. **Subrutina Delay(*)**: Esta subrutina es la encargada de esperar el retardo que se cumple cuando Cont_Delay =0.
- xviii. **Subrutinas Send_Command y Send_Data(*)**: Estas subrutinas son las encargadas de enviar los bytes de comando y datos al LCD. El byte a ser transmitido es pasado a las subrutinas por el acumular A.
- xix. **Subrutina Libre**: Esta subrutina será llamada únicamente en el Modo Libre. Esta subrutina enviará a la pantalla LCD el *Mensaje Modo Libre*, apagará los display de 7 segmentos y encenderá únicamente el led de Modo Libre.
- xx. **Subrutina Resumen**: Esta subrutina implementa el Modo Resumen, donde debe desplegarse el *Mensaje Resumen* en el LCD y los valores de VelProm y Vueltas en la pantalla de 7 segmentos.

4. Entregables.

4.1. Requisitos de admisibilidad

El proyecto final tiene dos requisitos de admisibilidad que deben ser cumplidos para pasar a la etapa de revisión.

- i. Que el informe esté completo incluyendo todas las partes que se detallan en la sección 4.2.
- ii. Que el programa cumpla con todos los requerimiento funcionales descritos en este enunciado.

Si alguno de los requisitos no es satisfecho el proyecto no se revisará y se asignará una nota de cero.

4.2 Informe escrito.

El informe escrito deberá incluir al menos las siguientes partes.

- a) Resumen: explicando el problema planteado y los resultados obtenidos, no mayor de una página ni menor a media página.
- b) Diseño de la aplicación: Este debe ser un capítulo que debe incluir un diagrama de flujos general y los diagramas de flujos de todos los programas que componen



la aplicación según su diseño, **cada uno de ellos con su explicación respectiva** y editados en forma electrónica. Las subrutinas deberán tener antes de su explicación, una descripción clara y sucinta de qué hacen, cuáles son y cómo se pasan los parámetros de entrada. Lo mismo deberá realizarse para los parámetros de salida. Todas las figuras y los diagramas de flujo deberán tener un número y una pequeña descripción al pie y deberán ser explicados, haciendo una referencia en el texto, usando el número respectivo.

- c) Todas las tablas, figuras o diagramas dentro del informe escrito, deberán estar numeradas y deberá haber una referencia en el texto a todas ellas.
- d) Conclusiones y comentarios personales sobre el trabajo realizado.
- e) Recomendaciones donde indique qué dificultades afrontó y recomendaciones sobre cómo podrían resolverse, dirigidas a quién vuelva a trabajar este problema.
- f) Bibliografía. Debe escribirse en un formato correcto.
- g) El informe escrito debe incluir un índice.

4.3 Codificación de la aplicación.

La codificación deberá hacerse 100% en lenguaje ensamblador del S12. Deberá incluir una parte sumamente ordenada para la declaración de las variables y las tablas.

Se adjunta la tabla con las estructuras de datos a utilizarse y su localización. Se ha incluido en esta tabla un registro con las banderas a utilizarse en el programa, dicho registro se llama BANDERAS y colocado en la dirección \$1000, algunas de las banderas han sido definidas y se han dejado algunas disponibles para utilizarlas en caso de ser necesarias . **En esta tabla se han dispuesto algunos espacios de memoria que pueden ser utilizados en caso de ser necesario. Si en el diseño se utilizan más estructuras de datos que las declaradas, es decir, si se usan los espacios disponibles, debe incluir en su declaración qué función cumplen y porque fue necesario utilizarlas.**

El programa deberá iniciarse a partir de la posición \$2000 y la **pila debe ubicarse a partir de la última posición de la memoria RAM disponible**. El **orden** de los programas deberá ser: rutina de **iniciación**, programa **principal**, subrutinas de **interrupciones**, subrutinas **generales**. Todo segmento de programa o subrutina, deberá tener un **encabezado** describiendo qué hace, **cómo se pasan los parámetros de entrada y de salida**. Deberán tener los **comentarios** respectivos al lado de las acciones más importantes.

Por medio de un video se presentaran los diagramas de flujo que muestran la arquitectura del programa a ser implementado y será esta arquitectura y solo esta la que deba implementarse. Con base en esta arquitectura deben ser diseñados las distintas partes del programa y los diagramas de flujo producto de ese diseño deben



UNIVERSIDAD DE COSTA RICA
ESCUELA DE INGENIERÍA ELÉCTRICA
MICROPROCESADORES
IE0623

EIE
Escuela de
Ingeniería Eléctrica

ser incluidos en el informe escrito. NO se admite que se incluyan, como parte del informe escrito, los diagramas de flujo discutidos en el video.

El trabajo escrito deberá remitirse de manera electrónica, en formato pdf.

Se habilitará la recepción del trabajo escrito y el código del programa en Mediación Virtual el día 30 de noviembre del 2020 entre las 8:00 a.m. y las 8:30 a.m. No se recibirán proyectos después de esa hora. El formato del nombre del archivo debe ser **NOMBRE_TF.asm** y **NOMBRE_TF.pdf** donde NOMBRE es su nombre personal. La defensa del proyecto será el día **3** de diciembre, por medio de la aplicación Zoom, en una cita individual que se asignará el día 30 de noviembre.



UNIVERSIDAD DE COSTA RICA
ESCUELA DE INGENIERÍA ELÉCTRICA
MICROPROCESADORES
IE0623

EIE
Escuela de
Ingeniería Eléctrica

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			