



PROYECTO FINAL. RADAR 623.

1. Instrucciones generales.

Este trabajo deberá realizarse individualmente. El trabajo se evaluará en dos partes: informe escrito y funciona o no funciona con un valor del 50% cada una, por lo que todas las subrutinas deberán ser probadas exhaustivamente por el estudiante. Los estudiantes, entregarán un informe escrito y mostrarán el funcionamiento de sus programas. El informe escrito tiene un valor del 50%, mientras que la parte de prueba tiene el 50% restante y se calificará corre o no corre, esto es, 0% o 50%. La parte escrita deberá tener todas las partes que corresponden a un trabajo escrito, tales como índice, resumen, cuerpo del trabajo, conclusiones, recomendaciones, bibliografía, etc. **No se aceptarán trabajos que no se entreguen de este modo.** Se deberán incluir los diagramas de flujo dibujados en forma electrónica, con una explicación de qué hace. El código en lenguaje de máquina deberá contener comentarios y divisiones en las diferentes partes del programa. Las subrutinas deberán tener en su encabezado, una descripción de qué hacen, cómo se le pasan los parámetros de entrada, cómo se reciben los parámetros de salida.

Todas las tablas, figuras o diagramas dentro del informe escrito, deberán estar numeradas y deberá haber una referencia en el texto a todas ellas.

Cualquier cambio al formato o a cualquier especificación indicada en este documento, es causa de rechazo del trabajo.

La entrega completa es obligatoria en la fecha indicada. No se aceptarán en ninguna otra fecha posterior. Quién no entregue una parte, pierde el trabajo final y obtendrá una nota de cero.

2. Descripción de la Aplicación.

Se desea diseñar e implementar un sistema para el despliegue de información de velocidad vehicular en una carretera. El sistema, denominado RADAR 623, cuenta con dos sensores ultrasónicos y una pantalla como se muestra en la Figura #1. En esta figura se puede observar que el sistema utiliza dos sensores ultrasónicos, separados 40 metros entre sí, para detectar el paso de un vehículo y calcular su velocidad. Incluye además una pantalla que contiene un display LCD y 4 dígitos de 7 segmentos, localizada a 200 metros del sensor S2, para presentar la velocidad promedio del vehículo y la velocidad límite en la carretera.

El objetivo es que, con base en la detección del vehículo por medio de los sensores, se calcule su velocidad media y sea desplegada en la pantalla donde el conductor pueda verla. Para esto la velocidad deberá ser desplegada en la pantalla 100 metros antes de que el



IE-623 MICROPROCESADORES
Prof. Geovanny Delgado

vehículo esté debajo de la misma y hasta que este la haya superado. La estimación de cuándo el vehículo se encuentra a 100 metros y de cuándo superó la pantalla, se hará basándose en la velocidad promedio calculada. Observe que la pantalla está colocada de manera transversal a la carretera.

Adicionalmente el sistema deberá desplegar, junto con la velocidad calculada, la velocidad límite especificada. Si la velocidad calculada es mayor que la velocidad límite el RADAR 623 barrerá los leds de arriba hacia abajo de manera repetida mientras se muestre la velocidad, indicando la condición de ALERTA.

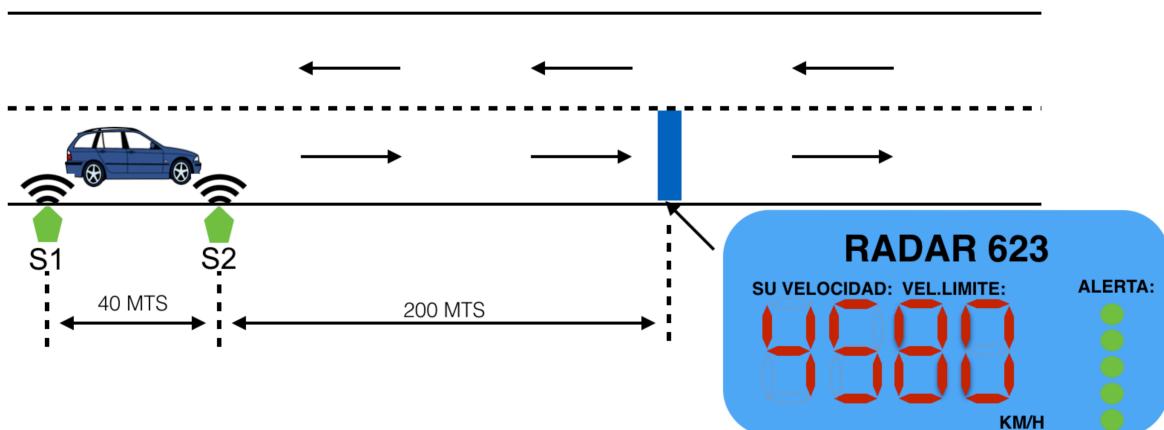


Figura #1
Imagen descriptiva de la operación del RADAR 623

Los sensores emiten una señal ultrasonica en forma transversal a la carretera, cuando pasa un vehículo, se detecta su presencia y se envía una señal al controlador. Cuando se pase frente al primer sensor se inicia el conteo del tiempo que requiere el vehículo para alcanzar el segundo sensor. Con este tiempo y con la distancia entre los sensores, se realiza el cálculo de la velocidad promedio del vehículo, en Km/h. Esta velocidad se presentará al conductor, según se indicó anteriormente.

Cuando el RADAR 623 detecte un vehículo en S1, deberá desplegar el siguiente mensaje en la pantalla LCD:





IE-623 MICROPROCESADORES
Prof. Geovanny Delgado

Los dígitos de 7 segmentos y los leds deberán permanecer apagados, excepto el de modo MED que deberá permanecer encendido mientras el sistema se encuentre en este modo como se discute más adelante.

Cuando el vehículo esté a 100 mts de la pantalla el mensaje en el LCD deberá cambiar a:



En los dígitos de 7 segmentos se desplegará la velocidad calculada en los dígitos de la izquierda y la velocidad límite programada en los dígitos de la derecha.

Luego de que el vehículo supere la posición de la pantalla, el LCD deberá cambiar el mensaje a:

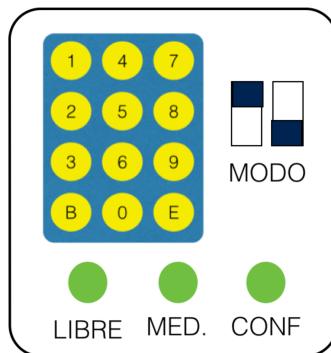


Los dígitos de 7 segmentos y los leds deberán permanecer apagados, excepto el led de modo y el sistema permanecerá en este estado hasta que se detecte nuevamente un vehículo en S1.

Se va considerar que el sistema será utilizado por un solo vehículo a la vez, de tal manera que los sensores estarán inactivos desde que el vehículo pasa frente a ellos y hasta que se termine de desplegar la velocidad en la pantalla. No se deberá realizar ninguna medición para el paso de vehículos en el sentido opuesto. La velocidad mínima a desplegar será $V_{MIN}=30\text{ Km/h}$ y la velocidad mayor será $V_{MAX}=99\text{ Km/h}$. Si algún objeto tiene una velocidad fuera de rango se presentará en los dígitos de 7 segmentos de la izquierda un par guiones (--) durante dos segundos. Inmediatamente después de esto el RADAR 623 cambiará el LCD al mensaje de “ESPERANDO...” y deberá quedar habilitado para leer los sensores y calcular una nueva velocidad a desplegar.

El RADAR 623 tendrá tres modos de operación: Libre, Medición y Configuración. Estos modos se seleccionarán por medio de dos interruptores denominados Modo. Los modos de operación del RADAR 623 son excluyentes, es decir, solo se puede estar en un modo a la vez. Cada vez que el RADAR 623 se pase de Modo Medición a cualquier otro modo se debe borrar la velocidad calculada y suspender cualquier cálculo en curso.

El RADAR 623 tiene una Consola de Control con la distribución de la siguiente figura:



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 en la pantalla:



En este modo los display de **7 segmentos** deberán permanecer apagados. El LED “Libre” deberá encenderse para indicar que el sistema se encuentra en este modo.

i. **Modo MEDICION.**

En el Modo Medición el RADAR 623 operará de acuerdo a las especificaciones descritas en la sección 2. El LED “MED” debe estar encendido en este modo.

ii. **Modo CONFIGURACIÓN.**

En el modo configuración es posible programar la velocidad límite V_LIM que activará la ALARMA. La velocidad límite solo podrá estar entre **45 km/h y 90 km/h**. Dicha cantidad deberá ser ingresada por el teclado del RADAR 623, que tiene el formato mostrado en la Consola de Control. Mientras el RADAR 623 permanezca en el Modo Configuración, pueden ser realizados sucesivos cambios a V_LIM. Cada cambio será reconocido y aceptado cuando se presiona la tecla E. Es posible borrar las secuencias de teclas ingresadas con la tecla B.



En este modo el LCD deberá mostrar el siguiente mensaje:



En el Modo Configuración el valor V_LIM programado deberá aparecer en la pantalla de 7 segmentos en los dos dígitos de la derecha, en todo momento, hasta que se salga del Modo Configuración. Cuando se vaya a cambiar se debe sobreescribir sobre el valor programado. El LED Config deberá estar encendido en este modo.

Al encendido se debe iniciar en modo CONFIG y deberá permanecer en este modo hasta que se programe un valor de V_LIM.

3. Estructura de la aplicación.

3.1. Asignación de Hardware.

- i. Se utilizará la pantalla LCD y los 4 dígitos de 7 segmentos de la Dragon 12+, como pantalla del RADAR 623, cumpliendo con todas las condiciones y estructuras de datos de la Tarea de Pantallas.
- ii. Se utilizarán los leds del puerto B, como los leds del RADAR 623, la asignación de los leds es la siguiente: PB2=LIBRE, PB1=MEDICION, PB0=CONIGURACION. Los leds PB7 a PB3 se usan para la alerta de sobrevelocidad y se encienden de izquierda a derecha (PB7 a PB3) simulando de arriba a abajo en el RADAR 623.
- iii. El teclado se va a implementar con la matriz de botones de la Dragon 12+ conectados al puerto A, cumpliendo todas las condiciones y estructuras de datos de la Tarea de Teclados Matriciales.
- iv. Se utilizarán los botones conectados al puerto H para implementar los sensores ópticos del RADAR 623. La asignación es la siguiente: PH3 = S1, PH0=S2. Los botones serán atendidos por interrupción.
- v. Se utilizarán los dipswitches PH7-PH6 de la Dragon 12 para definir el modo de operación, de la siguiente manera:

PH7-PH6 = ON-OFF	:MODO LIBRE
PH7-PH6 = ON-ON	:MODO MEDICION.
PH7-PH6 = OFF-OFF	:MODO CONIGURACIÓN
PH7-PH6 = OFF-ON	:NO ASIGNADO



- vi. Se utilizará el potenciómetro trimmer ubicado en el PAD7 como control de brillo de la pantalla.

3.2. Estructura 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- El programa debe incluir una Rutina de Inicialización que inicie todos los elementos dentro del S12, tal como borrado de las posiciones de RAM que lo requieran, programación de registros de control de los periféricos, inicio del puntero de pila y habilitación de interrupciones. Cualquier otra cosa que deba ser iniciada, deberá incluirse dentro de esta rutina. Init_HW :)
- 3- En todo momento el patrón de encendido de los diodos leds estará almacenado en una variable denominada LEDS.
- 4- 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 asecendente 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.
- 5- Subrutinas. La aplicación desarrollada deberá incluir las siguientes subrutinas (las subrutinas con * ya han sido desarrolladas en las tareas y se incluye su descripción solo como referencia o cuando se deba realizar alguna modificación):
 - i. **MODO CONFIG (*)**: Esta subrutina es llamada desde el programa principal. Esta subrutina llamará a la Tarea_Teclado con base en el valor de Array_OK. Al ingresar a la subrutina Modo Config, Array_OK debe estar en cero para permitir el primer llamado a la Tarea_Teclado, se debe colocar el valor de V_LIM en BIN1 y retornar. Cuando en un llamado se encuentre Array_OK=1, la subrutina Modo Config llama a la subrutina BCD_BIN y luego valida que V_LIM esté en el intervalo válido. De ser así borra la bandera Array_OK, coloca el valor de V_LIM en BIN1 para que el nuevo valor sea desplegado en la pantalla y retorna. Si el valor de V_LIM no está en el intervalo válido, entonces borra Array_OK, borra el valor en V_LIM y retorna para volver a iniciar la lectura de V_LIM en el teclado. Esta



IE-623 MICROPROCESADORES
Prof. Geovanny Delgado

subrutina devuelve al programa principal el valor de V_LIM a ser utilizado en el Modo Medición. Observe que V_LIM debe estar en cero después del encendido. Mientras el sistema esté en el Modo Configuración se podrá modificar V_LIM 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 V_LIM.
- 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.
- 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.
- 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 entre 1 y 6. 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



IE-623 MICROPROCESADORES
Prof. Geovanny Delgado

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 más baja posible para el ATD. 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 OC4_ISR iniciará la secuencia de conversión, como se indica más adelante. Esta fuente de interrupción estará habilitada en los tres modos de operación.
- viii. **MODO MEDICION:** La subrutina MODO MEDICION pone el mensaje MODO MEDICION/ESPERANDO... 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 asegurar que BIN1 y BIN2 estén en \$0B, para que inicialmente mantenga los display de 7 segmentos apagados. Cada vez que el RADAR 623 se salga del Modo Medición se deberá borrar las variables VELOC y ALERTA.
- ix. **Subrutina PANT_CTRL:** Esta subrutina es llamada desde MODO MEDICION y será la encargada de calcular el retardo de tiempo, con base en la velocidad



IE-623 MICROPROCESADORES
Prof. Geovanny Delgado

calculada, para cambiar el mensaje en la pantalla cuando el vehículo esté a 100 metros de la misma y luego cuando el vehículo la haya superado. Para ello la subrutina PANT_CTRL calculará los contadores de ticks necesarios para desplegar el primer mensaje (TICK_EN) y para cambiarlo luego de que el vehículo 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. Este valor de ticks es decrementado por la subrutina TCNT_ISR. Luego de esto la subrutina PANT_CTRL debe esperar, en un nuevo llamado a la misma, a que PANT_FLAG sea 1 para cambiar el mensaje en la pantalla y desplegar el valor de la velocidad, colocando V_LIM 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 de “MODO MEDICION/ESPERANDO...” 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 es mayor que V_LIM activará una bandera llamada ALERTA para que los leds de alerta indiquen esta condición. Si VELOC está fuera de rango debe poner un valor \$AA en VELOC para que se desplieguen los guiones (--) en la pantalla de 7 segmentos, cargar el TICK_DIS para generar el retardo de 2 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 el mensaje en el LCD a “MODO MEDICION/ESPERANDO...” para que inicie un nuevo ciclo de medición.

- x. **Subrutina CALCULAR:** Esta será la subrutina de servicio de interrupciones del puerto H y estará habilitada únicamente en el Modo Medición. Esta subrutina será la encargada de calcular la velocidad del vehículo en KM/H. La interrupción por PH3 pondrá la variable TICK_VEL en cero, mientras que la interrupción por PH0 leerá TICK_VEL y lo utilizará para calcular la velocidad del vehículo. La velocidad se colocará en una variable llamada VELOC y luego debe borrar el valor del TICK_VEL. Al darse la interrupción de PH0 debe cambiar el mensaje en el LCD a “CALCULANDO...”. Esta interrupción solo estará habilitada en el modo Medición. En el encabezado de la subrutina se debe incluir la ecuación utilizada para el cálculo de la velocidad del vehículo.
- 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:
 - a. Incrementar la variable TICK_VEL para el cálculo de la velocidad, dicha velocidad la calcula la subrutina CALCULAR.



IE-623 MICROPROCESADORES
Prof. Geovanny Delgado

- b. 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 Medición.
- 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 de manera recurrente desde OC4_ISR 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 visto en clase.
- 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



IE-623 MICROPROCESADORES
Prof. Geovanny Delgado

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.

- b. 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.
 - c. Debe decrementar el Cont_Delay en caso de que este no sea cero.
 - d. 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.
 - e. Cada 200 mS (5 HZ) debe llamar a la subrutina PATRON_LEDS.
- xvi. **PATRON LEDS:** Esta subrutina es llamada recurrentemente desde OC4_ISR y debe actualizar el valor de LEDS con el patrón de barrido de PB7-PB3 siempre que la bandera ALERTA esté en 1. Si ALERTA está en cero debe dejar los leds PB7 a PB3 apagados.
- xvii. **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.
- xviii. **Subrutina Delay(*):** Esta subrutina es la encargada de esperar el retardo que se cumple cuando Cont_Delay =0.
- xix. **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.



- xx. **Subrutina LIBRE(*):** Esta subrutina será llamada únicamente en el estado LIBRE. Esta subrutina enviará a la pantalla LCD el mensaje de modo libre, apagará los display de 7 segmentos y encenderá únicamente el led de Modo Libre.

4. Entregables.

4.1 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 incluir un diagrama de flujos general y los diagramas de flujos de 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 concisa 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.

Cualquier cambio al formato o a cualquier especificación indicada en este documento, es causa de rechazo del trabajo.

4.2 Codificación de la aplicación.

La codificación deberá hacerse 100% en lenguaje ensamblador. 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 deberá colocarse en la dirección \$1000. 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.



IE-623 MICROPROCESADORES
Prof. Geovanny Delgado

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.

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

No serán aceptados trabajos que no contengan todos estos elementos. El trabajo escrito deberá remitirse de manera electrónica, en formato pdf.

El trabajo escrito y el código del programa deberán enviarse a la dirección microp@cr-automation.com el día 9 de diciembre del 2019 a más tardar a las 8:00 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 11 de diciembre, en una cita individual que oportunamente se asignará.



UNIVERSIDAD DE COSTA RICA
ESCUELA DE INGENIERÍA ELÉCTRICA



IE-623 MICROPROCESADORES
Prof. Geovanny Delgado

ESTRUCTURAS DE DATOS - RADAR 623

VARIABLES				BANDERAS (\$1000-\$1001)		TABLAS
SUBRUTINAS	NOMBRE	TAMAÑO (BYTES)	POSICION	BANDERA	BIT	NOMBRE
MODO CONFIG	V_LIM	1	\$1002	TCL_LISTA	0	TECLAS (\$1030)
TAREA TECLADO:	MAX_TCL	1	\$1003	TCL_LEIDA	1	SEGMENT (\$1040)
	Tecla	1	\$1004	ARRAY_OK	2	InitDisp (\$1050)
	Tecla_IN	1	\$1005	PANT_FLAG	3	Inicio Mensajes (\$1060)
	Cont_Reb	1	\$1006	ALERTA	4	
	Cont_TCL	1	\$1007	CALC_TICKS	5	
	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			
TCNT_ISR	TICK_VEL	1	\$1012			
CONV_BIN_BCD	BIN1	1	\$1013			
	BIN2	1	\$1014			
	BCD1	1	\$1015			
	BCD2	1	\$1016			
BIN_BCD	BCD_L	1	\$1017			
	BCD_H	1	\$1018			
BCD_7SEG	DISP1	1	\$1019			
	DISP2	1	\$101A			
	DISP3	1	\$101B			
	DISP4	1	\$101C			
PATRON_LEDS	LEDS	1	\$101D			
OC4_ISR	CONT_DIG	1	\$101E			
	CONT_TICKS	1	\$101F			
	DT	1	\$1020			
	CONT_7SEG	1	\$1021			
	CONT_200	1	\$1020			
SUBRUTINAS LCD	Cont_Delay	1	\$1021			
	D2mS	1	\$1020			
	D240uS	1	\$1021			
	D60uS	1	\$1020			
	Clear_LCD	1	\$1021			
	ADD_L1	1	\$1020			
	ADD_L2	1	\$1021			
DISPONIBLES	A DEFINIR	4	\$1022-\$1025			