

	<p style="text-align: center;"><b>UNIVERSIDAD DE COSTA RICA</b> <b>ESCUELA DE INGENIERÍA ELÉCTRICA</b></p>	<p><b>EIE</b> Escuela de Ingeniería Eléctrica</p>
<p style="text-align: center;"><b>IE-623 MICROPROCESADORES</b> <b>Prof. Geovanny Delgado</b></p>		

## TAREA #5 MANEJO DE PANTALLAS.

En una línea de producción de tornillos se tiene una máquina dispensadora para empaque. Esta máquina debe contar los tornillos que se dispensan para empaque y al alcanzar la cuenta programada (CPROG) activar una salida (SAL) que despachará el empaque con la cantidad de tornillos completada y quedará lista para una nueva secuencia de dispensado.

El control de la máquina tendrá dos modos de operación: CONFIG y RUN. Para seleccionar entre estos dos modos se tiene un selector de modo MODSEL de dos posiciones.

**MODO CONFIG:** En este modo la máquina no contará tornillos sino que leerá el teclado para recibir la cuenta programada (CPROG). Cuando se esté en modo CONFIG se deberá encender el **LED PB1** y todos los demás LEDs deberán estar apagados. La cantidad a dispensar (CPROG), será ingresada por medio del teclado y podrá ser una cantidad entre **12 y 96 tornillos**. Cualquier valor ingresado fuera del intervalo deberá ser ignorado y se debe esperar un nuevo valor ingresado. La lectura del teclado se hará con base en las especificaciones de la Tarea de Teclados y se utilizarán las mismas estructuras de datos. Cuando se ingrese al modo CONFIG, en la pantalla LCD deberá aparecer el siguiente mensaje:

### **MODO CONFIG INGRESE CPROG:**

En la pantalla de 7 segmentos (DISP3-DISP4) deberá aparecer la cuenta **00** al encendido del sistema y deberá cambiar al último valor válido ingresado por el teclado. Este valor debe estar presente en la pantalla toda vez que el sistema esté en Modo CONFIG. Los otros dos dígitos DISP1-DISP2 deberán permanecer apagados. Luego de ingresado un valor válido para CPROG, este se desplegará en la pantalla y será posible pasar al modo RUN. Al encendido el sistema debe iniciar en modo CONFIG, para que deba ingresar un valor válido para CPROG. También se puede llegar al modo CONFIG desde el modo RUN.

**MODO RUN:** En el modo **RUN se contarán los tornillos detectados y se actualizará la cuenta en la pantalla de 7 segmentos**. Adicionalmente en el modo RUN se llevará un contador de los empaques procesados (ACUMUL). Este contador estará entre 0 y 99 y podrá rebasar a **cero al llegar a 99**. También este contador tendrá un botón de borrado manual (**ACUCLR**). Cuando el sistema esté en modo RUN deberá encenderse el LED PB0 y todos los demás LEDs deben estar apagados.

	<p style="text-align: center;"><b>UNIVERSIDAD DE COSTA RICA</b> <b>ESCUELA DE INGENIERÍA ELÉCTRICA</b></p>	<p><b>EIE</b> Escuela de Ingeniería Eléctrica</p>
<p style="text-align: center;"><b>IE-623 MICROPROCESADORES</b> <b>Prof. Geovanny Delgado</b></p>		

Al ingresar al modo RUN la pantalla LCD desplegará el siguiente mensaje:

**MODO RUN:**  
**ACUMUL. - CUENTA**

En los dígitos de la pantalla de 7 segmentos de la izquierda (DISP1-DISP2) se deberá mostrar la cuenta acumulada ACUMUL y los dígitos de la derecha se deberá mostrar el valor de la CUENTA en curso. Cuando CUENTA alcance el valor de CPROG el contador debe detenerse, se deberá incrementar el valor de ACUMUL y se debe accionar la salida SAL. El sistema permanecerá en este estado hasta que se presione el botón de REINICIO, momento en el que CUENTA vuelve a cero, se desactiva la salida SAL y la CUENTA se incrementará nuevamente con la detección de los tornillos, repitiendo la secuencia. Estando en el modo RUN es posible pasar al modo CONFIG en cualquier momento para modificar el valor de CPROG, en cuyo caso se deben borrar los valores de CUENTA y ACUMUL.

**ARQUITECTURA DE HARDWARE.**

**Teclado:** Se utilizará el teclado matricial con todos los alcances descritos en la Tarea de teclados.

**Sensor de tornillos:** El sensor de los tornillos se va a simular por medio de la interrupción RTI. El valor de CUENTA deberá incrementarse a una razón de 4 HZ.

**Pantalla LCD:** Se utilizará la pantalla LCD de la Dragon 12 para desplegar los mensajes descritos.

**Pantalla 7 segmentos:** Se utilizará la pantalla de 4 dígitos de 7 segmentos de la Dragon 12 para desplegar la información numérica.

**ACUCLR:** Este botón se implementará con el botón PH1.

**REINICIO:** Este botón se va a implementar con el botón PH0.

**MODOSEL:** Este interruptor se implementará leyendo por pooling el dipswitch PH7. En ON estará en Modo CONFIG y en OFF estará en Modo RUN.

**Salida SAL:** Activación del relé de la DRAGON 12.

	<p style="text-align: center;"><b>UNIVERSIDAD DE COSTA RICA</b> <b>ESCUELA DE INGENIERÍA ELÉCTRICA</b></p>	<p><b>EIE</b> Escuela de Ingeniería Eléctrica</p>
<p style="text-align: center;"><b>IE-623 MICROPROCESADORES</b> <b>Prof. Geovanny Delgado</b></p>		

## ARQUITECTURA DE SOFTWARE.

En el programa principal se debe leer recurrentemente el interruptor de **MODSEL** para poner el sistema en el modo correspondiente. **El valor de este interruptor es copiado como una bandera al bit 4 del registro BANDERAS**, cuando esté en 1 se pasa al Modo CONFIG y en cero se pasa al modo RUN. **Cabe destacar que no se puede poner el sistema en el Modo RUN si el valor de CPROG es cero**. Además cada vez que el valor de MODSEL se cambie de valor se debe poner la bandera CAMBIO\_MODO, que será utilizada para evitar el refrescamiento continuo de la pantalla LCD, como se discute más adelante.

**CambioModo=bit5 de banderas**

Se debe tener una variable LEDs que contiene el estado de los LEDs a ser desplegado.

El programa deberá incluir las siguientes subrutinas:

**MODULO CONFIG:** Esta subrutina es llamada desde el programa principal siempre que MODSEL =1. 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 CPROG 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 CPROG esté en el intervalo válido. De ser así borra la bandera Array\_OK, coloca el valor de CPROG en BIN1 para que el nuevo valor sea desplegado en la pantalla y retorna. Si el valor de CPROG no está en el intervalo válido, entonces borra Array\_OK, borra el valor en CPROG y retorna para volver a iniciar la lectura de CPROG en el teclado. Esta subrutina devuelve al programa principal el valor de CPROG a ser utilizado en el Modo RUN. Observe que CPROG debe estar en cero después del encendido. Mientras el sistema esté en el Modo CONFIG se podrá modificar CPROG todas las veces que sea necesario.

**SUBROUTINA BCD\_BIN:** Esta subrutina toma el valor en Num\_Array, lo convierte a binario y lo coloca en la variable CPROG.

**MODULO RUN:** Esta subrutina es la encargada de implementar el modo RUN. Al ingresar a esta subrutina una variable TIMER\_CUENTA debe haber sido cargada en su valor máximo (constante VMAX). Esta variable será decrementada por la subrutina RTI\_ISR para dar la cadencia de incremento de CUENTA. La subrutina Modo RUN debe primero comparar el valor de CUENTA con CPROG. Si CUENTA es igual a CPROG la subrutina MODO RUN, retorna (no hace nada). Si los valores no son iguales se verifica si TIMER\_CUENTA = 0, de ser así debe incrementar en valor de CUENTA y recargar TIMER\_CUENTA, para reiniciar un nuevo periodo de tiempo de la razón de incremento de CUENTA. Adicionalmente con el incremento de CUENTA se debe validar si su valor

	<p style="text-align: center;"><b>UNIVERSIDAD DE COSTA RICA</b> <b>ESCUELA DE INGENIERÍA ELÉCTRICA</b></p>	<p><b>EIE</b> Escuela de Ingeniería Eléctrica</p>
<p style="text-align: center;"><b>IE-623 MICROPROCESADORES</b> <b>Prof. Geovanny Delgado</b></p>		

alcanzó CPROG, de ser así debe incrementar el valor de ACUMUL. Finalmente la subrutina MODO RUN debe colocar el valor de CUENTA en BIN1 y el valor de ACUMUL en BIN2 para que sean desplegados adecuadamente en la pantalla de 7 segmentos.

**SUBROUTINA RTI\_ISR:** La RTI deberá implementarse con un periodo de 1 mS. En esta subrutina se deberá decrementar el valor de TIMER\_CUENTA siempre que este sea diferente de cero. Adicionalmente la subrutina RTI\_ISR deberá descontar Cont\_Reb toda vez que este sea diferente de cero.

**SUBROUTINA PTH\_ISR:** Esta subrutina será la encargada de atender tres tareas en función de la fuente de interrupción, como se describe a continuación. Todas las interrupciones deben atenderse por el flanco decreciente de las respectivas entradas.

- a. Interrupción PTH0: En esta interrupción se borra el valor de CUENTA. Esta interrupción solo debe estar habilitada en el modo RUN.
- b. Interrupción PTH1: En esta interrupción se borra el valor de ACUMUL. Esta interrupción solo debe estar habilitada en el modo RUN.
- c. Interrupción PTH3/PH2: En estas interrupciones se debe incrementar (PH3)/Decrementar (PTH2) un contador de brillo, almacenado en una variable denominada BRILLO cuyo valor estará entre 0 y 100. Por cada activación de uno de estos botones se debe incrementar/decrementar en **5 la variable brillo.** A partir de la variable brillo se deberá obtener el valor de K en la subrutina OC4\_ISR.

Para la lectura de los botones deberá implementarse la supresión de rebotes para ello se utilizará la variable Cont\_Reb de la supresión de rebotes del teclado, dado que no hay conflicto con el teclado, pues los botones y el teclado se utilizan en Modos diferentes.

**SUBROUTINA BIN\_BCD:** Esta subrutina recibe dos variables denominadas BIN1 y BIN2 ambas menores o iguales a 99 y convierte sus valores a BCD. El resultado de la conversión lo devuelve en la variable BCD1 para BIN1 y BCD2 para BIN2. La subrutina devolverá un valor de \$F en las posiciones de las variables correspondientes a los dígitos de pantalla que deben permanecer apagados. Esta subrutina debe ser implementada con el algoritmo visto en clase.

**SUBROUTINA BCD\_7SEG:** Esta subrutina 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 accedados con direccionamiento indexado,

	<p style="text-align: center;"><b>UNIVERSIDAD DE COSTA RICA</b> <b>ESCUELA DE INGENIERÍA ELÉCTRICA</b></p>	<p><b>EIE</b> Escuela de Ingeniería Eléctrica</p>
<p style="text-align: center;"><b>IE-623 MICROPROCESADORES</b> <b>Prof. Geovanny Delgado</b></p>		

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.

**SUBROUTINA OC4\_ISR:** Se deberá crear una subrutina llamada OC4\_ISR, que atenderá la interrupción Output Compare del Canal 4. Dicha subrutina de servicio recibe el valor en 7 segmentos a ser desplegado (variables DISP1 a DISP4) y la variable LEDS y se encargará de 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\_TICS 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 ( $DT = N \cdot K$ ). 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 BCD\_7SEG para que actualice los valores a ser desplegados en la pantalla. Para ello se utilizará una variable tipo word denominada CONT\_7SEG.

Adicionalmente la subrutina OC4 debe decrementar el Cont\_Delay en caso de que este no sea cero.

**Subrutina Cargar\_LCD:** Esta subrutina debe ser invocada cada vez que se deba actualizar el mensaje a ser desplegado en la pantalla LCD, es decir, cada vez que se cambie el modo de operación. Para ello se va a tener una bandera que estará en el bit 5 del registro de BANDERAS, denominada CAMBIO\_MODO. Esta bandera se activará cada vez que el sistema se cambie de modo, con el fin de no estar refrescando el LCD en cada ciclo del programa. La subrutina deberá escoger entre los cuatro 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.

**Subrutina Delay:** Esta subrutina es la encargada de esperar el retardo que se cumple cuando Cont\_Delay =0.


**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.

	<p style="text-align: center;"><b>UNIVERSIDAD DE COSTA RICA</b> <b>ESCUELA DE INGENIERÍA ELÉCTRICA</b></p>	<p><b>EIE</b> Escuela de Ingeniería Eléctrica</p>
<p style="text-align: center;"><b>IE-623 MICROPROCESADORES</b> <b>Prof. Geovanny Delgado</b></p>		

**NOTA:** Los ceros a la izquierda de cualquiera de las cuentas NO deben ser desplegados en la pantalla de 7 segmentos y consecuentemente estos dígitos deben permanecer apagados.

Entregue un informe con todos los detalles de diseño, incluyendo los cálculos realizados para el manejo de las frecuencias de interrupción. Incluya los diagramas de flujo de todas las subrutinas. Estructure el programa de tal manera que primero se realocalicen los vectores de interrupciones, luego se declaren todas las estructuras de datos. El código del programa debe ubicarse a partir de la posición \$2000 y debe primero configurarse todo el hardware a utilizarse, luego deben inicializarse todas las estructuras de datos y finalmente se debe incluir el código del programa principal. A continuación deben incluirse todas las subrutinas general y finalmente las subrutinas de servicio de las interrupciones. En la Tabla #1 se muestran las posiciones para las estructuras de datos a utilizarse, **únicamente** se deben utilizar estas estructuras de datos.

Debe enviar el código del programa con el formato SuNombre#5.asm a la dirección del curso a más tardar a las 8:00 a.m. del día en que debe entregar su tarea.

	<p style="text-align: center;"><b>UNIVERSIDAD DE COSTA RICA</b>  <b>ESCUELA DE INGENIERÍA ELÉCTRICA</b></p>	<p><b>EIE</b>  Escuela de  Ingeniería Eléctrica</p>
<p style="text-align: center;"><b>IE-623 MICROPROCESADORES</b>  <b>Prof. Geovanny Delgado</b></p>		

### ESTRUCTURAS DE DATOS

NOMBRE	TIPO	DIRECCION
MAX_TCL	Constante byte	\$1000
Tecla	Variable byte	\$1001
Tecla_IN	Variable byte	\$1002
Cont_Reb	Variable byte	\$1003
Cont_TCL	Variable byte	\$1004
Patron	Variable byte	\$1005
Banderas	Variable byte	\$1006
Num_Array	ARREGLO	\$1030
Teclas	ARREGLO	\$1040
CUENTA	Variable byte	\$1007
ACUMUL	Variable byte	\$1008
CPROG	Variable byte	\$1009
VMAX	Constante byte	\$100A
TIMER_CUENTA	Variable byte	\$100B
LEDS	Variable byte	\$100C
BRILLO	Variable byte	\$100D
CONT_DIG	Variable byte	\$100E
CONT_TICKS	Variable byte	\$100F
DT	Variable byte	\$1010
LOW	Variable byte	\$1011
BCD1	Variable byte	\$1012
BCD2	Variable byte	\$1013
DISP1	Variable byte	\$1014
DISP2	Variable byte	\$1015
DISP3	Variable byte	\$1016
DISP4	Variable byte	\$1017
SEGMENT	ARREGLO	\$1050
CONT_7SEG	Variable word	\$1019-\$101A
Cont_Delay	Variable byte	\$101B
D2mS	Constante byte	\$101C
D240uS	Constante byte	\$101D
D60uS	Constante byte	\$101E
Clear_LCD	Constante byte	\$101F
ADD_L1	Constante byte	\$1020
ADD_L2	Constante byte	\$1021
iniDsp	ARREGLO	\$1060
Inicio de mensajes*	ARREGLO	\$1070
Inicio de mensajes*: Dirección a partir de la cual se colocan los mensajes ASCII		

Empezar por OC  
luego por binbcd  
luego el resto

