Tarea 5

Indicaciones:

- Debe ser resuelta en las parejas de trabajo definidos.
- La fecha máxima de entrega es el día 12 de Febrero del 2021, como se describe en el cronograma del curso.
- Debe ser entregada a través del sistema Mediación Virtual. Remita el código fuente únicamente (Archivo .asm) así como el PDF con los diseños, explicaciones, notas de cálculo, etc. El formato del archivo debe ser NOMBRE1_NOMBRE2_T5.asm, puede remitir todos estos archivos de manera comprimida en un solo archivo .zip y el documento pdf debe nombrarse NOMBRE1_NOMBRE2_T5.pdf.

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

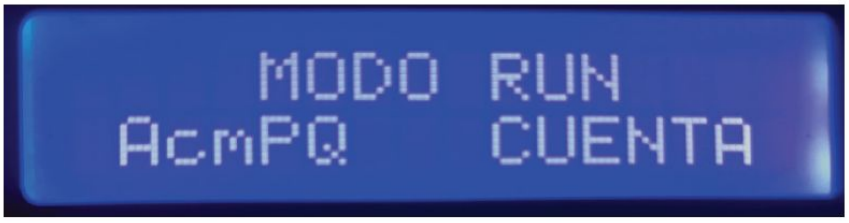
El control de la máquina tendrá dos modos de operación: CONFIG y RUN. Para seleccionar entre estos dos modos se tiene un selector de modo MODSEL de dos posiciones.

MODO CONFIG: En este modo la máquina no contará tornillos sino que leerá el teclado para recibir la cantidad de tornillos que van por paquete (CantPQ). Cuando se esté en modo CONFIG se deberá encender el LED PB1 y todos los demás LEDs deberán estar apagados. La cantidad a dispensar (CantPQ), será ingresada por medio del teclado y podrá ser una cantidad entre 25 y 85 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:

En la pantalla de 7 segmentos (DISP3-DISP4) deberá aparecer el 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 CantPQ, 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 CantPQ. También se puede llegar el 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 (AcmPQ). 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 (AcmCLR). Cuando el sistema esté en modo RUN deberá encenderse el LED PB0 y todos los demás LEDs deben estar apagados.

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



En los dígitos de la pantalla de 7 segmentos de la izquierda (DISP1-DISP2) se deberá mostrar la cuenta acumulada AcmPQ y los dígitos de la derecha se deberá mostrar el valor de la CUENTA en curso. Cuando CUENTA alcance el valor de CantPQ el contador debe detenerse, se deberá incrementar el valor de AcmPQ y se debe accionar la salida SAL. El sistema permanecerá en este estado hasta que se presione el botón de CuentaCLR, 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 CantPQ, en cuyo caso se deben borrar los valores de CUENTA y AcmPQ.

PTH

Arquitectura de Hardware:

1. Teclado: Se utilizará el teclado matricial con todos los alcances descritos en la Tarea de teclados.
2. 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 razón de 3 HZ.
3. Pantalla LCD: Se utilizará la pantalla LCD de la Dragon 12 para desplegar los mensajes descritos.
4. Pantalla 7 segmentos: Se utilizará la pantalla de 4 dígitos de 7 segmentos de la Dragon 12 para desplegar la información numérica.
5. AcmCLR: Este botón se implementará con el botón PH1.
6. CuentaCLR: Este botón se va a implementar con el botón PH0.
7. MODOSEL: Este interruptor se implementará leyendo por polling el dipswitch PH7. En ON estará en Modo CONFIG y en OFF estará en Modo RUN.
8. Salida SAL: Activación del relé de la DRAGON 12.

Arquitectura de Software:

En el programa principal se debe leer recurrentemente el interruptor de MODSEL para poner el sistema en el modo correspondiente, con MODSEL=1 se están en Modo Config, en caso contrario se pasa a Modo RUN. El valor de este interruptor es copiado a una bandera denominada ModActual, con el fin de que cada vez que el valor de MODSEL se cambie de valor se debe poner la bandera CambMod, que será utilizada para evitar el refrescamiento continuo de la pantalla LCD, como se muestra en el diagrama de flujos del programa principal.

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

El programa deberá incluir las siguientes subrutinas:

MODO 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 se debe colocar el valor de CantPQ en BIN1, para su despliegue en la pantalla de 7 segmentos, además Array_OK debe estar en cero para permitir el primer llamado a la Tarea_Teclado y se retorna. Cuando en un llamado se encuentre Array_OK=1, la subrutina Modo Config llama a la subrutina BCD_BIN y luego valida que CantPQ esté en el intervalo válido. De ser así borra la bandera Array_OK, coloca el valor de CantPQ en BIN1 para que el nuevo valor sea desplegado en la pantalla y retorna. Si el valor de CantPQ no está en el intervalo válido, entonces borra Array_OK, borra el valor en CantPQ y retorna para volver a iniciar la lectura de CantPQ en el teclado. Esta subrutina devuelve al programa principal el valor de CantPQ a ser utilizado en el Modo RUN. Mientras el sistema esté en el Modo CONFIG se podrá modificar CantPQ todas las veces que sea necesario. Debido al procedimiento descrito cada vez que se ingrese a Modo Config un valor válido de CantPQ debe ser ingresado.

CantPQ debe estar en cero después del encendido y el Modo Config debe ser el modo por defecto luego del power-up, independientemente de la posición del selector de modo. Cuando el operador haya ingresado un valor válido para CantPQ, podrá salir del modo Config y avanzar a la lectura de MODOSEL para seguir operando conforme a esto.

SUBRUTINA BCD_BIN: Esta subrutina toma el valor en Num_Array, lo convierte a binario y lo coloca en la variable CantPQ.

MODO RUN: Esta subrutina es la encargada de implementar el modo RUN y es llamada de manera recurrente desde el programa principal. Al ingresar a esta subrutina una variable TIMER_CUENTA debe haber sido cargada en el power-up en su valor máximo, que se declarará en el programa como la etiqueta VMAX. TIMER_CUENTA será decrementada por la subrutina RTI_ISR para dar la cadencia de incremento de CUENTA. Luego, en el modo RUN se debe esperar a que el valor de TIMER_CUENTA sea cero, cada vez que esto

ocurra la subrutina RUN debe recargar la variable TIMER_CUENTA, además en este momento se debe incrementar el valor de CUENTA y se verifica si su valor llegó a CantPQ. De ser así se debe incrementar el valor de AcmPQ y se detiene la interrupción RTI, esto detiene el proceso de conteo de tiempo e incremento de cuenta. Si el valor de CUENTA no ha llegado a CantPQ, simplemente se retorna para iniciar un nuevo ciclo de temporización de TIMER_CUENTA para el incremento de CUENTA.

Finalmente la subrutina MODO RUN debe colocar el valor de CUENTA en BIN1 y el valor de AcmPQ en BIN2 para que sean desplegados adecuadamente en la pantalla de 7 segmentos.

SUBRUTINA 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.

SUBRUTINA 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 (CuentaCLR): En esta interrupción se borra el valor de CUENTA y se habilitan las interrupciones de RTI. Esta interrupción solo debe estar habilitada en el modo RUN.
b. Interrupción PTH1 (AcmCLR): En esta interrupción se borra el valor de AcmPQ. Esta interrupción sólo 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.

SUBRUTINA 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 de 7 segmentos 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-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.

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

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.

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.

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.

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 la bandera denominada CambMod. Esta bandera se activará cada vez que el sistema cambie de modo, con el fin de no estar refrescando el LCD en cada ciclo del programa. 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.

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 acumulador A.

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 relocalicen los vectores de interrupciones, luego se declaran 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 anexa se muestran las posiciones para las estructuras de datos a utilizarse, únicamente se deben utilizar estas estructuras de datos.

Estructuras de datos:

NOMBRE	TIPO	DIRECCION
Banderas	Variable byte	$1000
MAX_TCL	Constante byte	$1001
Tecla	Variable byte	$1002
Tecla_IN	Variable byte	$1003
Cont_Reb	Variable byte	$1004
Cont_TCL	Variable byte	$1005
Patron	Variable byte	$1006
Num_Array	ARREGLO	$1007
CUENTA	Variable byte	$1009
AcmPQ	Variable byte	$100A
CantPQ	Variable byte	$100B
TIMER _CUENTA	Variable byte	$100C
LEDS	Variable byte	$100D
BRILLO	Variable byte	$100E
CONT_DIG	Variable byte	$100F
C0NT_TICKS	Variable byte	$1010
DT	Variable byte	$1011
BIN1	Variable byte	$1012
BIN2	Variable byte	$1013
BCD_L	Variable Byte	$1014
LOW	Variable byte	$1015
TEMP	Variable byte	$1016
BCD1	Variable byte	$1017

BCD2	Variable byte	$1018
DISP1	Variable byte	$1019
DISP2	Variable byte	$101A
DISP3	Variable byte	$101B
DISP4	Variable byte	$101C
CONT_7SEG	Variable word	$101D
Cont_Delay	Variable byte	$101F
D2mS	Constante byte	$1020
D260uS	Constante byte	$1021
D40uS	Constante byte	$1022
Clear_LCD	Constante byte	$1023
ADD_L1	Constante byte	$1024
ADD_L2	Constante byte	$1025
Teclas	ARREGLO	$1030
SEGMENT	ARREGLO	$1040
iniDsp	ARREGLO	$1050
Inicio de mensajes*	ARREGLO	$1060
Inicio de mensajes*: Dirección a partir de la cual se colocan los mensajes ASCII		

BANDERAS: X:X:X:CambMod:ModActual:ARRAY_OK: TCL_LEIDA:TCL_LISTA