



TAREA #4

LECTURA DE UN TECLADO MATRICIAL

Como ha sido discutido en clase, los teclados matriciales constituyen un elemento fundamental de un sistema incrustado pues permiten ingresar información al programa en tiempo de ejecución. Dicha información puede ser la selección de una función de la aplicación, un valor numérico o incluso un valor alfanumérico. Existe una gran variedad de teclados matriciales autocontenidos disponibles en el mercado y son, en general, componentes de bajo costo. En la Figura #1 se muestran teclados matriciales típicos disponible en las tiendas de electrónica.

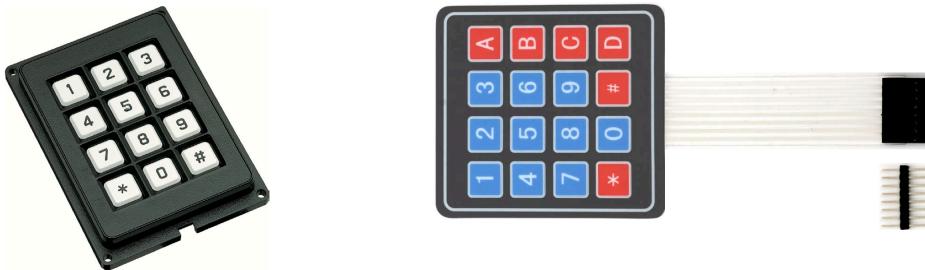


Figura #1

En el caso de la Dragon 12, el fabricante ha dispuesto una matriz de botones alambrada de forma matricial para ser usada como teclado. En la Figura #2 aparece una fotografía del teclado matricial de la Dragon 12. Consulte los diagramas esquemáticos de la Dragon 12 para efectos de entender el alambrado del teclado.

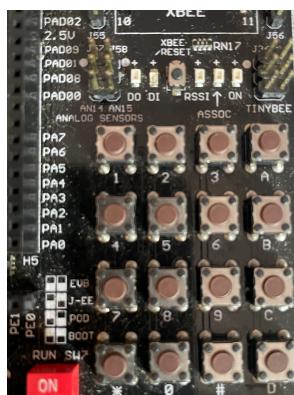


Figura #2

Teclado matricial de la Dragon 12.



En esta tarea se va a desarrollar el código necesario para leer exitosamente un teclado matricial. El teclado a leer será el teclado a utilizarse en el Proyecto Final del curso, de ahí la importancia de desarrollar con éxito esta tarea, pues todo el código será reutilizado y es preciso dominar todo el desarrollo del código para poder insertarlo con éxito en el Proyecto Final. El teclado a utilizarse será de 16 teclas ordenado con la distribución de la Figura #3.

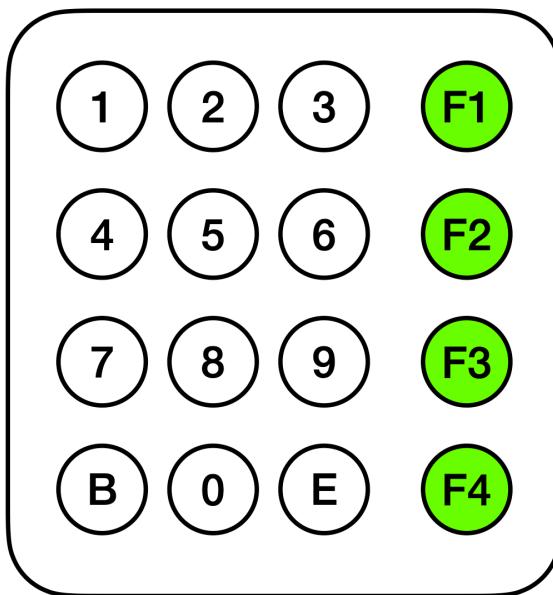


Figura #3
Distribución del teclado a leer.

El programa debe ser capaz de:

- i. Leer las teclas de función (F1 a F4) y poner un cero en el bit correspondiente del nibble superior de la variable denominada Funcion, siendo el bit Funcion.4 el asociado a la función F1. En la variable función debe haber solo un 0 y este estará ubicado en el bit asociado a la última tecla de función presionada. Al iniciarse el programa la variable función debe estar en \$FF.
- ii. Leer las teclas presionadas y crear un arreglo con los valores en BCD de la secuencia de teclas ingresadas. La secuencia de teclas tendrá una longitud definida por medio de una constante y su valor está entre 1 y 5. Es decir, se podrá leer una secuencia de mínimo una y máximo 5 teclas. El usuario podrá digitar una secuencia de teclas en el intervalo definido hasta presionar la tecla E (enter). La secuencia de teclas presionada debe quedar almacenada en un arreglo denominado NUM_ARRAY. Durante el ingreso de la secuencia de teclas el usuario puede presionar la tecla B (borrar) en cuyo caso se borrará la última tecla presionada colocada en el arreglo. Luego de alcanzada la longitud



máxima de la secuencia el programa solo podrá atender la tecla E o la tecla B. Cualquier otra tecla presionada debe ser ignorada. Si en una secuencia de teclas la primera tecla que se presiona es la tecla E o la tecla B estas deben ser ignoradas. El programa solo podrá leer una tecla a la vez, es decir, no se implementa “Rollover”.

1. Arquitectura de Hardware.

El teclado se va a implementar con la matriz de botones de la Dragon 12+ conectados al puerto A. Además:

- i. Se dispondrán de los leds menos significativos (PB3_PB0) para indicar, en todo momento, cuál función está activa.
- ii. Se utilizará el botón ubicado en PH0 para borrar la secuencia de teclas ingresada en cualquier momento, mediante un LongPress de dicho botón. Dicho botón se denominará CLEAR y al presionarlo debe borrarse el NUM_Array y la bandera Array_OK, dejando el programa listo para el ingreso de una nueva secuencia de teclas. Un ShortPress de Clear encenderá el Led PORTB.6 y una acción de borrado lo apagará. Se debe crear una subrutina llamada Borrar_Num_Array que borra el arreglo utilizando direccionamiento indexado de post incremento. Esta subrutina borrará el arreglo cuyo tamaño está definido en la constante MAX_TCL.

2. Arquitectura de Software.

Para el ingreso de una secuencia de teclas válidas (ingreso de un valor numérico válido), el programa utilizará una tabla denominada Teclas que contendrá: \$01, \$02,..., \$08, \$09, \$B, \$0, \$E, en ese orden. El valor \$B es el código asignado para la tecla B (borrar) y el valor \$E para la tecla E (enter). La tabla Teclas deberá ser accesada por direccionamiento indexado por acumulador, donde la dirección base es Teclas y el offset es definido por la tecla presionada. El objetivo del programa es llenar un arreglo denominado Num_Array con la secuencia de teclas presionadas y activar un bit (en cero) en caso de que una tecla de función haya sido presionada. En tanto no se inicie una secuencia de teclas presionadas, el arreglo Num_Array estará borrado y en todas las posiciones de su contenido habrá un \$FF. La longitud de ese arreglo es el tamaño de la secuencia de teclas más larga que se puede ingresar, mismo que está almacenado en una constante denominada MAX_TCL. Adjunto encontrará las direcciones de las estructuras de datos definidas en este enunciado. Serán esas y solamente esas estructuras de datos que deban ser utilizadas en la solución.



Programa Principal: El programa principal debe declarar e inicializar las estructuras de datos y configurar la interrupción RTI y todos los periféricos necesarios. El programa deberá desarrollarse a partir del código de leer un botón pulsador (PB), desarrollado en clase. Dicho código ya cuenta con una Máquina de Tiempos, una Tarea para el Led testigo y el código completo para leer un PB.

Descripción de la Arquitectura del Software: En el video que acompaña este enunciado se discute la arquitectura del programa a utilizar. Los diagramas de flujo presentados en este video NO se admiten como parte de la solución, pues son diagramas generales para orientar el diseño que debe realizar. El programa debe estructurarse como es usual, con encabezado de documentación, versión del programa, autor, fecha y comentarios generales. Luego debe aparecer la declaración de las estructuras de datos del programa y relocalización del vector de interrupción. A continuación debe venir el código de configuración del hardware y después el programa principal. El código debe iniciar en la dirección \$2000. Luego se debe colocar el código de las máquinas de estado, las subrutinas generales (si las hay) y finalmente la subrutina de servicio de la interrupción, en ese orden.

PRUEBAS A VALIDARSE.

- a. Se definirá un valor para MAX_TCL.
- b. Se ingresarán secuencias de teclas de longitud menor que MAX_TCL y se validará que las mismas queden almacenadas correctamente en Num_Array.
- c. Se ingresará una secuencia de teclas de longitud igual que MAX_TCL y se validará que la misma quede almacenadas correctamente en Num_Array.
- d. Se ingresarán secuencias de longitud mayor que MAX_TCL y luego Enter y se validará que solo las primeras teclas hasta una longitud igual que MAX_TCL quedaron en Num_Array.
- e. Se presiona las teclas B y E como primera tecla y se valida que no se modifica Num_Array.
- f. Al presionar una única tecla de manera prolongada y luego la tecla E, el valor de la primera queda en Num_Array. Validación de funcionalidad de tecla retenida.
- g. Al presionar teclas rápidamente, seguidas de la tecla E las mismas quedan en Num_Array sin ser repetidas. Se suprime correctamente los rebotes.
- h. Al presionar varias teclas consecutivas y luego una secuencia B---B-E, se valida la correcta conformación de Num_Array. Validación de la función de borrado.
- i. Al presionar una tecla de función se activa únicamente el led respectivo del puerto B.



- j. Al presionar el botón Clear se borran los valores almacenados en Num_Array.
- k. Los valores de las teclas deben leerse de manera fluida sin ningún retraso y repetición de tecla para que pueda ser leída.
- l. Se validará que ninguna tecla de función está activa por defecto.
- m. Se presionará una tecla de función y se validará que el respectivo led se active.
- n. Se validará que solo una función puede estar activa a la vez.

Entregue un informe con todos los detalles de diseño, incluyendo los cálculos realizados. Incluya los diagramas de flujo de todas las subrutinas indicando qué función cumple cada una de ellas, qué parámetros se pasa en el llamado y cómo devuelve los resultados. Debe incluir qué estructuras de datos utiliza cada subrutina. No se deben utilizar más estructuras de datos que las indicadas en la tabla de este enunciado. Estructure adecuadamente el programa. No se revisarán diseños que no sean presentados de una manera ordenada y clara.

Debe remitir el código del programa con el formato SuNombreT4.asm y su informe en formato SuNombreT4.pdf

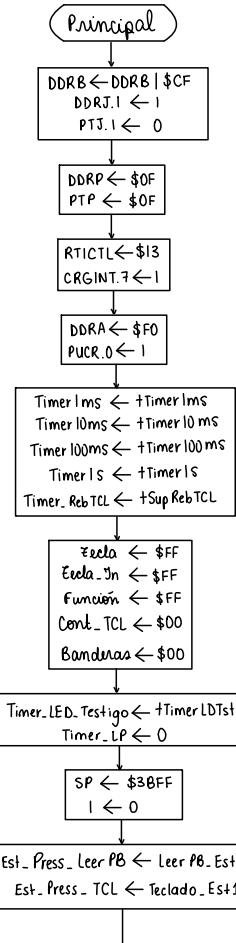


Tabla de Estructuras de Datos

constante →

Nombre	Dirección	Valor Inicial	VALORES
TAREA_TECLADO			
MAX_TCL	\$1000	1-5	
Tecla	\$1001	\$FF	
Tecla_IN	\$1002	\$FF	
Cont_TCL	\$1003	\$00	
Patron	\$1004	\$00	
Funcion	\$1005	\$FF	
Est_Pres_TCL	\$1006-\$1007	TareaTCL_Est1	tSuprRebTCL
Num_Array	\$1010	\$FF	
BANDERAS			
Banderas	\$100C	\$00	ShortP Mask \$01
			LongP Mask \$02
			ArrayOK Mask \$04
TAREA LEER_PB			
EstPres_LeerPB	\$100D-\$100E	LeerPB_Est1	PortPB, Mask \$01
			tSupRebPB
			tShortP
			tLongP
TABLAS			
Teclas	\$1020	Codigos de Teclas	
TABLA DE TIMER		\$1040	
Tabla_Timers_BaseT	Timer1mS		tTimer1mS
	Timer10mS		tTimer10mS
	Timer100mS		tTimer10mS
	Timer1S		tTimer1S
	\$FFFF		
Tabla_Timers_1mS	Timer_RebPB		
	Timer_RebTCL		
	\$FF		
Tabla_Timers_Base10mS	Timer_SHP		
	\$FF		
Tabla_Timers_Base100mS	Timer1_100mS		
	\$FF		
Tabla_Timers_Base1S	Timer_LP		tTimerLongP
	Timer_LED_Testigo		tTimerLDTst
	\$FF		

BANDERAS: X:X:X:ARRAY_OK:LongP:ShortP



- Despachador de tarea**
- máquina de tiempo RTI
 - Tarea_Dece_EcaleTimers
 - Tarea_Led_Estigos
 - Tarea_LeerPB
 - Tarea_Eeclado
 - Tarea_Leds

