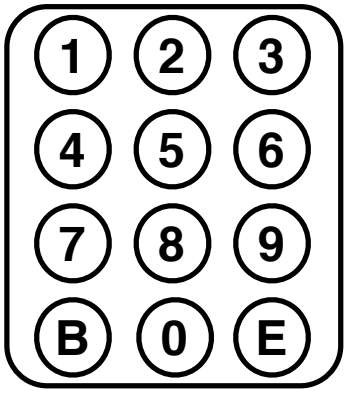
TAREA #4
LECTURA DE TECLADO MATRICIAL

Se debe realizar un programa para leer un teclado matricial. El teclado tiene el siguiente formato:



El programa debe ser capaz de leer los valores de las teclas en BCD y crear un arreglo con la secuencia de teclas ingresada. 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 como se discute más adelante. 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. Utilizando los primeros 12 botones de izquierda a derecha y de arriba hacia abajo.

2. **Arquitectura de Software.**

El programa utilizará una tabla llamada Teclas= [$01, $02,..., $08, $09, $0B, $0, $E), en ese orden. El valor $B es para la tecla B (borrar) y $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 rellenar un arreglo denominado Num_Array con la secuencia de teclas presionadas. En tanto no se inicie una secuencia de teclas presionadas, el arreglo 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.

Programa Principal: El programa principal debe declarar e inicializar las estructuras de datos y configurar las interrupciones RTI y PH0. Además deberá llamar de manera recurrente a la Tarea_Teclado, siempre que la bandera ARRAY_OK sea cero. Esta bandera es puesta en 1 cuando se finaliza una secuencia de teclas y es borrada por el procedimiento que usa la secuencia de teclas leída. En este caso, para efectos de prueba, esta bandera será borrada por medio de la interrupción PH0.

Subrutina RTI_ISR: La subrutina de servicio a la interrupción RTI únicamente debe descontar el contador de rebotes (Cont_Reb) solo si este es diferente de cero. Si este contador es cero esta subrutina no ejecuta ninguna acción y retorna. Esta interrupción se debe generar a una razón de 1 mS.

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). Esta subrutina **debe** ser implementada según el diseño discutido en el video.

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 se debe leer los patrones, en su lugar se debe buscar 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.

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 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 estás 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.

Subrutina PHO_ISR: Esta es la subrutina de servicio a la interrupción de PH0. En esta subrutina lo único que se hace es borrar la bandera ARRAY_OK y borrar Num_Array poniendo todos sus valores en $FF, para permitir el ingreso de una nueva secuencia de teclas.

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 de vectores de interrupción. A continuación debe venir el código de configuración del hardware y después el programa principal, mismo que debe empezar en la dirección $2000. Luego se deben colocar las subrutinas generales y finalmente las subrutinas de servicio de interrupciones.

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 suprimen 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.

Entregue un informe con todos los detalles de diseño, incluyendo los cálculos realizados. Incluya los diagramas de flujo de todas las subrutinas e incluya las estructuras de datos utilizadas, indicando qué función cumple cada una de ellas. No se deben utilizar más estructuras de datos que las indicadas en la tabla de este enunciado. Estructure adecuadamente el programa. No se revisarán diseñs que no sean presentados de una manera ordenada y clara.

Como parte del enunciado de esta tarea se estará compartiendo un video con la explicación de la arquitectura de software de la solución. 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.

Debe remitir el código del programa con el formato SuNombreT4.asm y su documento de diseño con las notas de cálculo y explicaciones respectivas a la plataforma de Mediación Virtual entre las 9:00 a.m. y las 9:30 a.m. de la fecha asignada para entregar la tarea.

TABLA DE ESTRUCTURAS DE DATOS

Nombre	Dirección	Valor Inicial
MAX_TCL	$1000	1-6
Tecla	$1001	$FF
Tecla_IN	$1002	$FF
Cont_Reb	$1003	$00
Cont_TCL	$1004	$00
Patron	$1005	$00
Banderas	$1006	$00
Num_Array	$1007-$100C	$FF
Teclas	$100D-$1018	$01-$0E

BANDERAS: X:X:X:X:X:ARRAY_OK: TCL_LEIDA:TCL_LISTA