Universidad de Costa Rica

Microprocesadores

Tarea 04

Yoel Moya Carmona - B75262

31 de Octubre de 2023

Índice

1. Descripción del diseño **2**
1.1. Tarea Teclado . . . 2
1.1.1. Sub rutina Leer Teclado . . . 2
1.1.2. Estados 1 al 3 . . . 3
1.1.3. Estado 4 . . . 3
1.2. Tarea Borra TCL . . . 4

2. Anexos **6**
2.1. Diagrama Leer Teclado . . . 6
2.2. Diagrama Estados 1 al 3 de Teclado . . . 7
2.3. Diagrama Estado 4 de Teclado . . . 8
2.4. Diagrama Borrar TCL . . . 9

1. Descripción del diseño

Todos los diagramas de flujo realizados para cada uno de los problemas se encuentran en la sección 2 de anexos.

Se utilizo la Tarea de Led Testigo, para verificar el funcionamiento apropiado de la maquina de tiempos, así como la Tarea Leer PB y Led PB en conjunto para implementar la funcionalidad de borrado total del arreglo de resultados.

En la configuración del *hardware* se implemento para el puerto A únicamente:

1. `movb #$F0,DDRA` con el objetivo de habilitar la parte alta del puerto como salidas, y la parte baja como entradas.
2. `bset PUCR,$01` las cuales habilitan para el puerto A las resistencias de *pull up* necesarias para el funcionamiento correcto del teclado.

En la Tabla de *timers*, se implemento en la tabla de 1mS un temporizador con el nombre de `Timer_RebTCL`. Se tomó la decisión de mantener `Timer_RebTCL` y `Timer_RebPB` en la tabla de 1mS, debido a que todavía se esta utilizando la interrupción RTI como corazón de la maquina de tiempos, la base de tiempo de $\approx 1mS$ sigue siendo valida. Como más adelante se utilizará *Output Compare*, esta base de tiempos pasará a no ser de utilidad.

En el código principal, se inicializan las variables. El arreglo `Num_Array` se inicializa con un ciclo utilizando como contador `Cont_TCL` para colocar en las primeras 9 posiciones el valor `$FF`, lo que debería ser más que suficiente ya que el valor máximo a sostener son 6 datos.

1.1. Tarea Teclado

Esta tarea posee una subrutina asociada para leer el teclado, la cual retorna por medio de memoria el valor leído, y si no lee ninguno, retorna `$FF`. La variable resultado en este estado se denomina `Tecla`.

Esta tarea se subdivide en cuatro estados, donde los primeros tres estados realizan una lectura inicial del teclado para manejar los rebotes de los botones y no generar múltiples lecturas; una confirmación de la pulsación de una tecla luego que se agote el tiempo de los rebotes; y la confirmación de la liberación de la tecla pulsada; respectivamente.

El cuarto estado toma el valor de la tecla pulsada y decide según su valor y las condiciones en las que se encuentra `Num_Array`, si ingresa la tecla, la ignora, borra valores del arreglo, o finaliza el ingreso de valores al arreglo.

1.1.1. Sub rutina Leer Teclado

El diagrama de esta sub rutina se encuentra en la sección 2.1.

En esta sub rutina se inicializa el patrón de escritura/lectura sobre el teclado, así como la inicialización del acumulador B como offset de fila, el cual, si no se presiona ninguna tecla sostendrá únicamente los valores 0, 3, 6, y 9, los cuales son el valor inicial de cada fila de la matriz del teclado mapeado a la tabla `Teclas`.

Si en algún momento lee un valor, se le suma al acumulador B la posición de la tecla según la columna en la que se encuentra, y así se tiene el *offset* real de la tecla presionada en la tabla de teclas. Se utiliza el índice Y para sostener la dirección base de dicha tabla.

Dicha tabla se definió de la siguiente manera (Como se expresa en el enunciado):

```
$1020 -> $01
$1021 -> $02
$1022 -> $03
$1023 -> $04
$1024 -> $05
$1025 -> $06
$1026 -> $07
$1027 -> $08
$1028 -> $09
```

```
$1029 -> $00
$102A -> $0E
$102B -> $0B
```

Lo cual va a ser importante a la hora de leer lo escrito sobre el estado 4.

1.1.2. Estados 1 al 3

El diagrama de estos estados se encuentra en la sección 2.1.

Estado 1:

Se realiza una lectura del teclado y se compara con `$FF`, el cual es el valor de ninguna tecla presionada. En dado caso que el valor sea distinto, se inicia el *timer* de supresión de rebotes, y se asigna el estado 2 como proximo estado. En este momento se asigna el valor de `Tecla` a `Tecla_IN`, la cual en este diseño cumple la función de variable temporal la cual sera de utilidad en los próximos estados.

Estado 2:

Se mantiene en este estado hasta que el tiempo de supresión de rebotes haya finalizado. Una vez que este tiempo sea cero, se realiza una lectura del teclado, y el resultado de esta lectura se compara con la variable `Tecla_IN`, que contiene la posible tecla presionada en el estado 1.

Si esta tecla es la misma que retorna la subrutina `Leer_Teclado` por `Tecla`, se puede tomar el valor de `Tecla` como presionado, y no como un resultado de ruido eléctrico o alguna otra fuente de error. En ese caso se puede pasar al estado 3, y de no haber sido la misma tecla presionada, se ignora el valor y se devuelve al estado 1.

Estado 3:

En este estado se realiza una nueva lectura del teclado, con el objetivo de buscar el momento en el que la tecla presionada se suelte. Para este momento, ya se conoce cual es el valor de la tecla, el cual se encuentra en la variable temporal `Tecla_IN`, por lo que se mantiene en este estado hasta que el valor que devuelve la subrutina `Leer_Teclado` sea diferente de este. El momento que esto suceda, se asigna a la variable `Tecla` el valor de `Tecla_IN` y se pasa al cuarto y ultimo estado.

1.1.3. Estado 4

El diagrama de este estado se encuentra en la sección 2.3.

En la creación del diagrama se dio bastante uso a los conectores dentro de pagina. Como este estado solo debe realizar cuatro acciones, se dividió en bloques donde se definen estas cuatro acciones, y luego bloques donde se evalúa la tecla presionada y el estado en ese instante del arreglo `Num_Array` para definir a cual de los bloques se debe direccionar el programa.

Los bloques de acciones se definen como (Los cuales son representados por los conectores de pagina del mismo color):

- *Agregar Valor:* Donde se toma el valor de la tecla presionada y se ingresa al arreglo de resultados tomando la dirección base y sumándole el offset, seguidamente se incrementa el offset en uno. Este offset se encuentra en la variable `Cont_TCL`.
- *Borrar Valor:* Se decrementa el offset, y se le suma este valor a la dirección base, seguidamente se escribe un `$FF` en esa posición.
- *Reestablecer:* Devuelve la maquina al primer estado, limpia la variable `Tecla` y `Tecla_IN` escribiendo `$FF` en ambas.
- *Finalizar Array:* Limpia la variable `Cont_TCL` poniendola en cero, y levanta la bandera de `ARRAY_OK`, la que simboliza que el arreglo se encuentra listo.

Los incrementos y decrementos de la variable `Cont_TCL` a la hora de escribir o borrar teclas se realizan de tal manera que el Offset siempre quede apuntando al primer valor vacío del arreglo, lo que resulta en mayor facilidad para comparar con `MAX_TCL`.

La sección de decisiones se representa en la parte superior de la pagina, donde es necesario mencionar la interpretación que se le dio a los valores de la tabla de teclas.

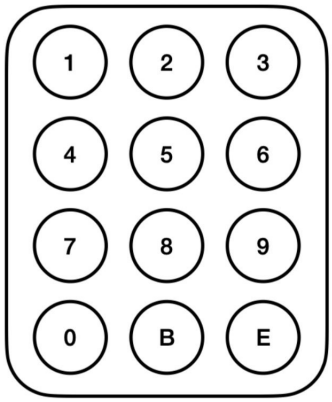


Figura #3

Figura 1: Representación de los valores que posee cada tecla

Como se puede observar en la Figura 1, la fila 1 corresponde a los primeros 3 valores de la tabla `Teclas`, la fila 2 a los siguientes 3 y así sucesivamente, hasta la ultima fila, en la cual el valor de B de la Figura corresponde al valor `$0E` de la Tabla, y el valor E de la Figura corresponde al valor `$0B` de la tabla. Teniendo esto en cuenta, para mantener consistencia con el enunciado, se decide interpretar los valores `$0B` y `$0E` como E y B de la Figura 1 respectivamente. De esta manera se puede mantener de manera operativa el funcionamiento esperado en la Figura 1, y la definición especificada de la tabla.

Teniendo esto en cuenta, se puede comenzar con el bloque de decisiones del diagrama:

Inicialmente se evalúa si el arreglo esta lleno, comparando la máxima cantidad de teclas permitidas con el offset que apunta a la posición del primer valor vació (que se puede interpretar como la cantidad de datos actual en el arreglo). Si estos dos son iguales, o de alguna manera el offset es mayor, se deben ignorar todos los valores que no sean una acción de *borrado*(B) o de *enter*(E), por lo que se evalúa la variable `Tecla` por los valores `$0E` y `$0B` y se redirige al bloque de acción de Borrar Valor o de Finalizar Arreglo respectivamente. En dado caso de no ser ninguno de esos, se pasa al bloque Reestablecer.

Si el array no esta lleno, se evalúa si esta completamente vació, ya que no se debería de realizar ninguna acción con B o E, por lo que se redirige a Reestablecer. En caso que la variable `Tecla` no tenga `$0E` o `$0B`, se redirige a Agregar Valor.
Si el array no esta lleno, y no esta vació, las tres opciones de `Tecla` son validas, por lo que se realiza la redirección hacia Borrar Valor, Finalizar Array, y Agregar Valor si el valor en `Tecla` es `$0E`, `$0B` o diferente de estos dos, respectivamente.

1.2. Tarea Borra TCL

El diagrama de esta Tarea se encuentra en la sección 2.4.

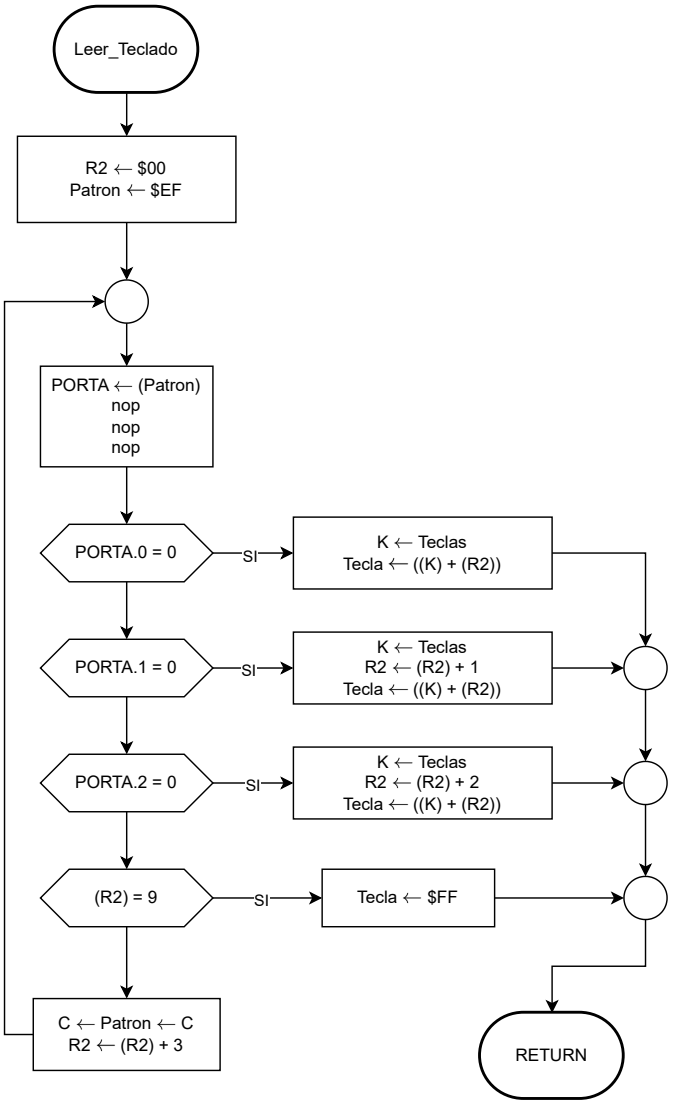
Esta tarea implementa la misma arquitectura del ciclo que inicializó el arreglo `Num_Array` en el código principal para borrar las variables, además de un `bclr` para borrar la bandera de `ARRAY_OK`.

Se adaptó la Tarea LED PB desarrollada en clase, la cual utiliza las banderas `ShortP` y `LongP` para determinar si el botón en la posición cero de `PORTB` es un *short press* o un *long press* respectivamente.

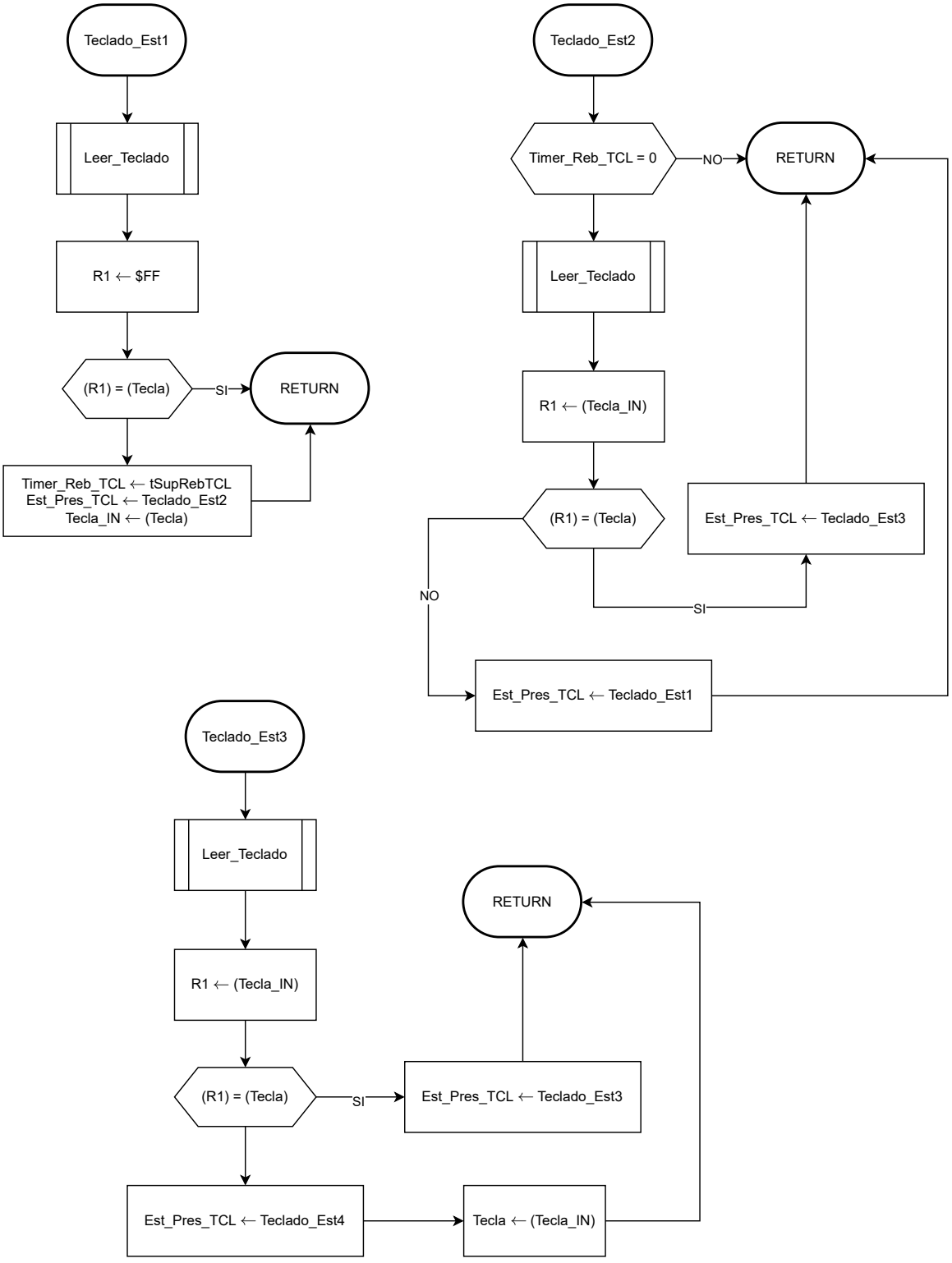
Cabe mencionar que el *long press* necesita almenos 5 segundos, los cuales pueden ser contados de manera intuitiva como: *un missisipi, dos missisipi, ..., cinco missisipi*. Esto se menciona ya que durante el proceso de desarrollo y plan de pruebas, este detalle fue de suma importancia en el proceso de *debugging*.

2. Anexos

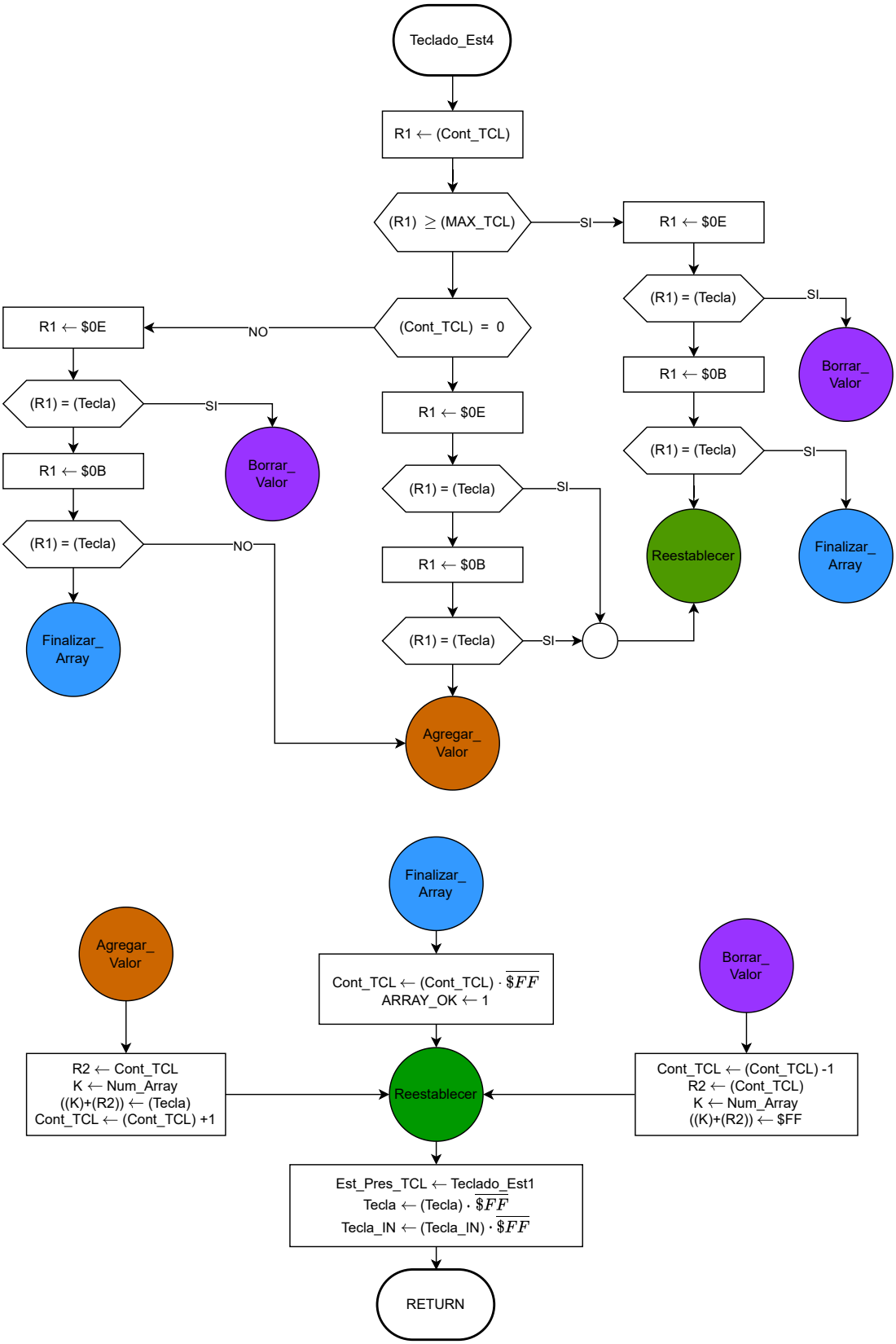
2.1. Diagrama Leer Teclado



2.2. Diagrama Estados 1 al 3 de Teclado



2.3. Diagrama Estado 4 de Teclado



2.4. Diagrama Borrar TCL