- Nombre y CI en cada hoja
- Numere las hojas
- Indique el total de hojas en la primera
- Utilice solo un lado de las hojas
- Incluya un solo problema por hoja
- **Sea prolijo**

Aprobación: mínimo UN problema

Problema 1

Se debe implementar, utilizando un Z80 y periféricos, un teclado para un sistema de control de iluminación inteligente.

Los teclados del sistema tienen una dirección única (máxima cantidad de direcciones: 32) configurable mediante 5 llaves selectoras en el frente, 2 teclas para selección de escenas de iluminación, 2 leds a un lado de las teclas y se comunican con el procesador central mediante un bus de comunicación serie.

El teclado tiene 2 modos de funcionamiento: modo programación y modo normal.

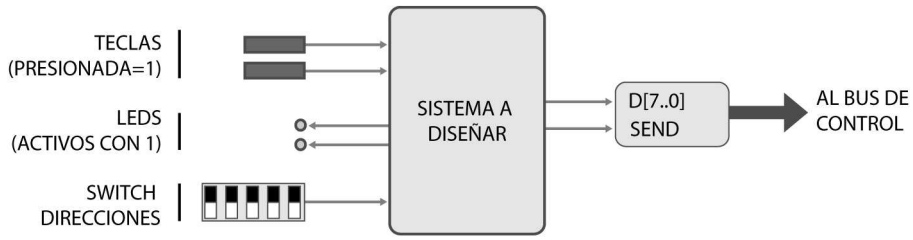
Modo programación
El modo programación es utilizado por el instalador para agregar nuevos teclados al sistema. Una vez conectado el teclado al bus de control, este debe ser puesto en modo programación.

En este modo, el teclado envía al bus 10 veces un byte especial (uno cada 1 segundo) que contiene la dirección que le fue asignada con las llaves selectoras y enciende en forma intermitente (un segundo prendido, un segundo apagado) todos sus leds. El formato de este byte es: [0 S4 S3 S2 S1 S0 0 0] siendo Si el valor de llave i.

A este modo se ingresa manteniendo presionada por más de 3 segundos las teclas 1 y 2. Si estando en la espera de que transcurran 3 segundos se suelta una de las teclas, se deberá operar como si se hubiese presionado solo la tecla aún activa.

Modo normal
Este es el modo normal de operación. Cada vez que se presiona una tecla, al soltarla se debe encender el led correspondiente a la tecla presionada y enviar por el bus serie un byte al sistema, informándole la tecla presionada y en que teclado. El formato de este byte es: [1 S4 S3 S2 S1 S0 T1 T0] siendo Si el valor de llave i y T1 Y T0 se corresponden con cada una de las 2 teclas:
Se pide:
a) armar todo el HW del sistema con :
* puertos de entrada para leer llaves con dirección y monitorear el estado de las teclas (por polling),
* puertos de salida para manejar los leds y enviar bytes a trasmisor serie,
* CTC para medir los tiempos (no utilizar interrupciones),
* memoria necesaria.

Cada vez que el trasmisor serie recibe un flanco de subida en su entrada SEND, envía el byte que se encuentra en su entrada D.



b) Todo el software de inicialización y rutina principal para que sistema funcione como se especificó.
Luego de un reset, el teclado debe comenzar con todos sus leds apagados.

Notas:
- Frecuencia del reloj del sistema: 4Mhz
- 1 seg aprox= 250ns * 256 * 256 * 61 y 3 seg aprox= 250ns * 256 * 256 * 183

Problema 2

Un sistema recibe bytes de información por su entrada **dat[]** y los reenvía encriptados por su salida **cifrado[]**. El procedimiento de cifrado se basa en un bloque combinatorio FUNC, con entrada de 8 bits y salida de 8 bits.
Para el primer byte, el dato cifrado se obtiene evaluando la funcion **FUNC** sobre el resultado del XOR entre el dato recibido y una clave inicial. De ahí en más el dato cifrado se obtiene con el mismo procedimiento, pero utilizando el dato cifrado anterior en lugar de la clave.

$cifrado_1$ = FUNC[dat_1 XOR CLAVE] ; para el primer byte
$cifrado_n$ = FUNC[dat_n XOR $cifrado_{n-1}$] ; para los siguientes

La presencia de un dato válido en **dat[]** es indicada por un flanco de subida en **dav**.
Cada vez que se escribe un nuevo dato en la salida **cifrado[]** la señal **rdy** debe subir a 1 hasta que desde el exterior indiquen que el dato fue consumido con un pulso a nivel bajo de la señal **stb**.

Un flanco de subida en la entrada **restart** indicará el comienzo de un nuevo flujo de datos. Para cifrar el primer dato del nuevo flujo deberá utilizarse nuevamente CLAVE en lugar del dato cifrado anterior. Luego de un reset, CLAVE debe inicializarse con el valor presente en la entrada **clave_inicial[8]** y quedar a la espera de un flanco en **restart** que indique el comienzo del primer flujo de datos. Los datos que lleguen por la entrada **dat[]** antes del primer **restart** deben ser ignorados.

Diseñar el sistema encriptador utilizando un sistema con Z80. La recepción de cada nuevo dato debe generar una interrupción en la cual se debe cifrar el dato y escribir el valor correspondiente en la salida **cifrado[]**. El programa principal debe encargarse de detectar el comienzo de un nuevo flujo de datos.
Se dispone de un bloque hardware que evalúa la función **FUNC**. Con la ayuda de un buffer triestado, el bloque FUNC deberá conectarse mapeado en el espacio de memoria a partir de la dirección 0400h como si fuera una ROM de 256x8, de manera que FUNC(XXh) pueda obtenerse leyendo el contenido de la dirección 04XXh.

a) Diseñar completamente el sistema, incluyendo:
- i) Todo el hardware
- ii) Rutina de atención a interrupciones
- iii) Inicialización, programa principal, directivas de reserva de memoria

b) Escribir las condiciones que deben cumplir los parámetros de tiempos del bloque FUNC, de la lógica de decodificación, del Z80 y del resto de los componentes utilizados para que no sea necesario insertar tiempos de espera en las lecturas del bloque FUNC.

tdeco-min =< tdeco =< tdeco-max ; retardo de toda la lógica de decodificación
tfunc-min =< tfunc =< tfunc-max ; retardo entrada-salida del bloque FUNC
toe-min =< toe =< toe-max ; retardo habilitación-salida del buffer triestado
tprop-min =< tprop =< tprop-max ; retardo entrada-salida del buffer triestado

Nota: si bien en la versión inicial del sistema todos los programas son escritos por Ud., el sistema debe soportar en forma sencilla la incorporación futura de nuevas tareas al programa principal que serán escritas por otros desarrolladores.