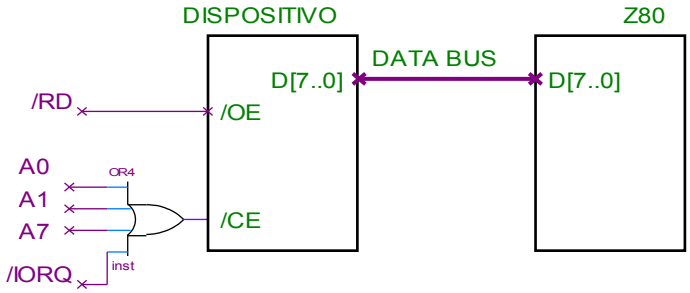
a) Utilice solo un lado de las hojas
b) Incluya un solo problema por hoja.
c) Sea prolijo.
d) Indique Nombre, CI y numere cada hoja.

Ejercicio 1 (18 pts.)

Luego de que se activan /CE y /OE el dispositivo externo de la figura tarda t_D en poner datos estables en sus salidas.

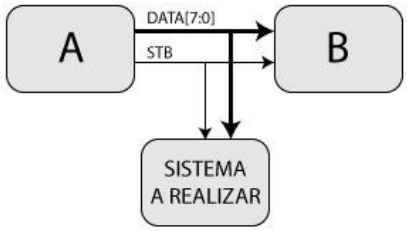


Se pide:

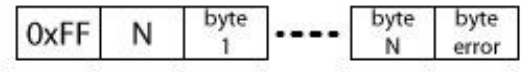
a) En función de la cantidad N de tiempos de espera insertados, los parámetros de tiempos del dispositivo externo, del microprocesador Z80 y de la compuerta OR ($T_{OR\_min} < T_{OR} < T_{OR\_max}$), escribir todas las condiciones necesarias para que en el ciclo de lectura de E/S se respete el tiempo de setup de datos del Z80. Indicar para cada parámetro, cuando corresponda, si debe considerarse el valor máximo o el mínimo. Los períodos o semiperíodos de reloj T que aparezcan en las ecuaciones se deben anotar indicando a qué período corresponden dentro del ciclo (p. ej. T1, $T2_{LOW}$, etc.).
b) Determinar qué requerimientos se debe imponer al parámetro del dispositivo externo para que el sistema funcione correctamente, si N = 0, T_{OR_min} =0, T_{OR_max} =12ns y se está utilizando un Z84C0020 con un reloj de ciclo de trabajo 50% (T_{HIGH}=T_{LOW}) y frecuencia fclk= 20MHz.

Ejercicio 2 (18 pts.)

El equipo A envía mensajes, compuestos por varios bytes, al equipo B mediante un bus de datos de 8 bits y una señal de STB. Cada vez que hay un byte de datos nuevo, el equipo A genera un pulso a cero de 1T en STB. El dato es válido desde el flanco de subida hasta el siguiente pulso en STB.



De forma de sincronizar la comunicación y evitar recibir mensajes empezados, el comienzo está indicado con el byte 0xFF, luego lo sigue un byte que indica la cantidad de bytes que componen el mensaje (entre 1 y 250) y por último un byte para chequeo de errores. Se garantiza que los bytes de datos y el byte de chequeo de error nunca toman el valor 0xFF.



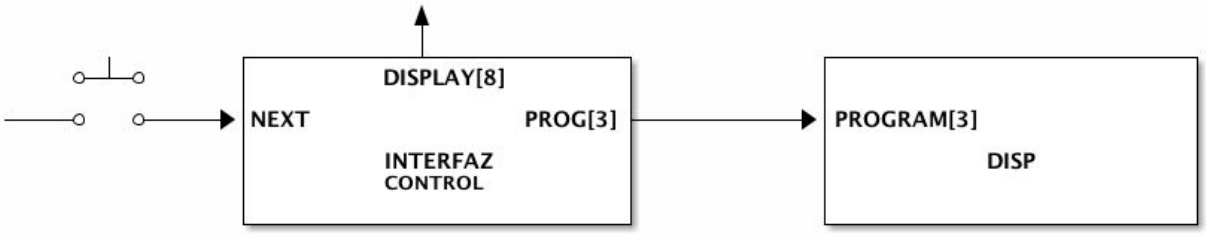
Se deberá implementar, utilizando un Z80 con buses triestado, un sistema que permita detectar el comienzo de un mensaje y el fin, indicando si hubo errores en la transmisión.
Deberá de contar con puertos de entrada para poder "mirar" la comunicación entre A y B, y 3 puertos de salida:

- RECIBIENDO: se mantiene en 1 desde que comienza la recepción de un mensaje (se detecta 0xFF), hasta que se recibe el byte para chequeo de errores.
- FIN_CON_ERROR: se enciende al terminar de recibir un mensaje si hubo error.
- FIN_SIN_ERROR. se enciende al terminar de recibir un mensaje si no hubo error.

Las señales de FIN se apagan al comenzar una nueva recepción.

El chequeo de error consiste en hacer un XOR entre los N bytes de datos que se reciben y compararlo con el byte para chequeo de errores recibido al final.
a) hardware completo del sistema: puertos, memorias, decodificación
b) software a ejecutarse luego de un reset que implemente el funcionamiento especificado.

Ejercicio 3 (64 pts.)



Se desea diseñar una interfaz para controlar un dispositivo *DISP* a través de un único pulsador **NEXT** (presionado=0, liberado=1). El operador humano pulsará **NEXT** con el objetivo de seleccionar uno de los posibles 8 programas que DISP tiene preconfigurados. *DISP* lee cuál de los 8 programas (0, 1, ..., 7) debe ejecutar en su entrada **PROGRAM[3]** que será manejada por la interfaz de control. El funcionamiento del sistema debe ser tal que:

- Cada vez que se presione el pulsador **NEXT**, la interfaz incrementa el valor desplegado en la salida **DISPLAY** en forma circular (INC(7) = 0), sin modificar aún la salida PROG.
- Si **NEXT** se mantiene presionado por un tiempo igual o mayor a **768ms (3 x 256ms)**, se deberá volver a incrementar el valor mostrado en **DISPLAY** cada **256ms** siempre y cuando el pulsador se mantenga activado. Para medir estos intervalos de tiempo de **256ms** se utilizará un bloque timer que interrumpirá al procesador.
- La interfaz únicamente actualizará su salida **PROG** (con el valor de programa correspondiente al desplegado en **DISPLAY**) en la primera interrupción del timer en que se detecte que el pulsador **NEXT** fue liberado.

El sistema funcionará con dos interrupciones: la ya mencionada del bloque timer y otra que se generará con el flanco de bajada del pulsador **NEXT.**
Notar que la interrupción generada por el timer debe poder leer el estado de la señal **NEXT** para determinar si el pulsador ya fue liberado.
La interrupción del pulsador **NEXT** deberá incrementar el valor en **DISPLAY** y configurar el timer para que se produzcan interrupciones sucesivas cada **256ms**.

Se diseñará este sistema con un microprocesador **T80** funcionando en **modo 2** de interrupciones. Se dispone de dos controladores de interrupciones y un bloque timer. Todos los dispositivos del sistema funcionarán con un reloj de **4MHz**.

Para el diseño de la solución se podrá suponer que el tiempo entre que se libera el pulsador y se vuelve a activar es siempre mayor a 256ms.

Se pide:

a) Puertos de entrada y salida, periféricos y su conexionado al T80. Solo está disponible el rango de direcciones **0x00 a 0x7F.**
b) Rutinas de atención a las interrupciones de pulsador **NEXT** y timer.
c) El código de inicialización del sistema, que luego ejecuta un `JP PRINCIPAL`. El programa principal realiza otras tareas que no se pide implementar.

Se debe indicar dónde se ubica en memoria cada sección de código y variables mediante directivas ORG . Se dispone de 32K de ROM y 32K de RAM ya conectados al sistema.

Nota: 256ms = 1/(4MHz) * 4096 * 250