**PREINFORME**

**PRACTICA 3**

**PROYECTO II**

**AUTORES:**

**Gina Catalina Baquero Mojica**

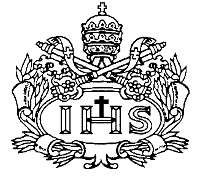
**Juan Sebastián Barreto Jiménez**

**Juan Pablo Clavijo Cáceres**

**PRESENTADO A:**

**Ing. Martha Lucía Cano Morales**

**Ing. José Reinaldo Vuelvas Quintana**



**PONTIFICIA UNIVERSIDAD JAVERIANA**

**FACULTAD DE INGENIERÍA**

**DEPARTAMENTO DE ELECTRÓNICA**

**BOGOTÁ D.C. 2020**

1. Diseñar una máquina de estados qué de solución el siguiente problema:
   1. Se quiere almacenar en una memoria 24LC256 un archivo de 1.000 datos de tipo entero de 16 bits proveniente de un computador. Para esto se usará el microcontrolador ATMEGA328P, el cual recibirá los datos desde el PC a través de la UART y los almacenará en la memoria a través de la interfaz I2C.
   2. Una vez almacenados los datos en la memoria del 24LC256, desde el PC se podrá solicitar nuevamente el archivo para ser graficado en el computador, o se podrá solicitar un dato específico.

Nota: La velocidad de transmisión del serial deberá ser estipulada y justificada en el diseño.

* + 1. Selección de FSM: la máquina de estados finita propuesta, cuenta con tres procesos diferentes, cada uno de los procesos esta denotado por un color en específico, y la selección de este depende del primer estado [Read Function USART] en este estado se espera a que el computador le envié al microcontrolador la función que el requiere realizar en donde tenemos tres posibles selecciones:
       1. **FSM Write 1000 (rojo)**
          - Este proceso cuenta con un total de 10 estados, para entrar al primer estado [Read Function USART] debe recibir un “write1000” por usart del pc para entrar a este proceso y al recibir esta señal. El microcontrolador envía por usart en ASCII al pc “ready” cuando esté listo, se espera a que la señal de tx\_ena sea ‘1’ para empezar a recibir los 8 bits menos significativos del primer dato y los guarda en la variable data\_lsb, cuando acabe dicho envió, la señal de tx\_ena se pone en ‘1’ para ahora proceder a recibir los 8 bits más significativos del primer dato y estos son almacenados en la variable data\_msb.
          - Después de haber recibido todo el dato completo es decir los 16 bits, el microcontrolador envía por I2C la condición de START y se activa la señal de enable para I2c es decir que I2C\_ena = ‘1’ y cambia al siguiente estado [Setup Slaver Write] en donde el microcontrolador ubica a la memoria (esclavo) a través de su dirección, mandado un código de control, su dirección y el bit de lectura o escritura de memoria, seguido de esto, la memoria baja el bit de ACK en caso de que la conexión haya sido exitosa, si no lo es, el ACK seguirá en alto y se necesitará repetir todo el proceso hasta que si sea exitosa.
          - Al recibir esta confirmación de ACK = ‘1’ cambia al siguiente estado [Send addres\_msb] en donde el microcontrolador envía a memoria los bits más significativos de la dirección a escribir y seguido de esto si el bit de ACK está en ‘0’ microcontrolador envía a memoria los bits menos significativos de la dirección a escribir en el siguiente estado.
          - Cuando este envio sea realizado exitosamente, se baja el bit de ACK y se cambia al estado de [Send data\_lsb] en donde el microcontrolador envía a memoria la variable data\_lsb y al enviar exitosamente este dato se repite el proceso con los msb.
          - Cuando haya finalizado el envío del dato completo el microcontrolador envía por I2C la condición de STOP y se realiza este proceso 999 veces más para cada uno de los datos requeridos. Finalmente, el microcontrolador envía por usart en ASCII al pc “finish” para indicar que ya fueron guardados los 1000 datos en memoria.
       2. **FSM Read 1000 (morado)**
          - Este proceso cuenta con un total de 11 estados, para entrar al primer estado [Read Function USART] debe recibir un “read1000” por usart del pc para entrar a este proceso y al recibir esta señal.
          - El microcontrolador envía por usart en ASCII al pc “ready” cuando esté listo y se espera a que la señal de tx\_ena sea ‘1’ para entrar al estado [Setup Slaver Write] donde el microcontrolador ubica a la memoria (esclavo) a través de su dirección, mandado un código de control con su dirección y el bit de lectura o escritura de memoria. Después de hacer esto se baja el bit de ACK para que el microcontrolador envíe a memoria los bits más significativos de la dirección a leer que va a estar dada por el contador 2000. Al finalizar el envío de baja el bit de ACK como modo de confirmación para proceder a enviar a memoria los bits menos significativos de la dirección a leer.
          - Después de realizar este envío el bit de ACK se pone en ‘0’ y se pasa al siguiente estado [Setup Slaver Read] en donde el microcontrolador envía a la memoria el código de control, su dirección y el bit de RW de memoria para que este en la capacidad de leer los datos.
          - Cuando ACK se baja a ‘0’ y entra en el estado de [Recieve Data Memory] donde la memoria envía al microcontrolador el dato de la dirección solicitada y luego en el siguiente estado [Save Data Memory] el microcontrolador recibe y almacena la información en data\_lsb cuando el contador 2000 es par y en data\_msb cuando el contador 2000 es impar, e incrementa el contador 2000. Cuando el microcontrolador lee el contador 2000, en caso de ser impar, el microcontrolador envía por I2C la condición de STOP, si no repetirá todo el proceso, pero con la nueva dirección del puntero. Despues de esto en los estados [Send data\_msb] y [Send data\_lsb] el microcontrolador envía por usart al pc las variables data\_msb y data\_lsb para que luego el microcontrolador revise que el buffer este vacío y envíe por usart al pc las variables.
          - Cuando haya finalizado todo el proceso el microcontrolador revisa el contador 2000 si es igual a 2000 envía por usart en ASCII al pc “finish” y finalmente, el microcontrolador envía por usart en ASCII al pc “finish” para indicar que ya fueron guardados los 1000 datos en memoria.
       3. **FSM Read data (rosado)**
          - Este es el último proceso posible en la FSM y cuenta con un total de 11 estados, para entrar al primer estado [Read\_1000] debe recibir un “read\_data” por usart del pc para entrar a este proceso y al recibir esta señal.
          - El microcontrolador envía por usart en ASCII al pc “ready” cuando esté listo y se espera a que la señal de tx\_ena sea ‘1’ para entrar al estado [Setup Slaver Write] donde el microcontrolador ubica a la memoria (esclavo) a través de su dirección, mandado un código de control con su dirección y el bit de lectura o escritura de memoria. Después de hacer esto se baja el bit de ACK para que el microcontrolador envíe a memoria los bits más significativos de la dirección. Al finalizar el envío de baja el bit de ACK como modo de confirmación para proceder a enviar a memoria los bits menos significativos de la dirección a leer tal y como se realizó en el proceso descrito anteriormente en morado.
          - Después de realizar este envío el bit de ACK se pone en ‘0’ y se pasa al siguiente estado [Setup Slaver Read] en donde el microcontrolador envía a la memoria el código de control, su dirección y el bit de RW de memoria para que este en la capacidad de leer los datos. Cuando ACK se baja a ‘0’ y entra en el estado de [Recieve Data Memory] donde la memoria envía al microcontrolador el dato de la dirección solicitada y luego en el siguiente estado [Save Data Memory] el microcontrolador recibe y almacena la información en data\_lsb. El microcontrolador recibe y almacena la información en data\_lsb cuando el contador es 0 y en data\_msb cuando el contador es 1, e incrementa el contador y el microcontrolador lee el contador, en caso de estar en 2, el microcontrolador envía por I2C la condición de STOP, si no sucede esto se repetirá dicho proceso, pero ahora con la nueva dirección del puntero.
          - En los estados de [Send data\_msb] y [Send data\_msb] el microcontrolador envía por usart al pc la variable data\_msb, luego revisa que el buffer este vacío y envía por usart al pc la variable data\_lsb.
          - Finalmente el microcontrolador revisa cuando haya finalizado todo el proceso el microcontrolador envía por usart en ASCII al pc “finish” y finalmente, el microcontrolador envía por usart en ASCII al pc “finish” para indicar que ya fueron guardados los 1000 datos en memoria.

1. Defina un protocolo de comunicación para:
   1. Enviar desde el computador al microcontrolador los 1000 datos de 16 bits.
      1. El pc envía por usart en ASCII al microcontrolador "write1000".
      2. El microcontrolador envía por usart en ASCII al pc “ready”.
      3. El pc envía por usart los bits menos significativos del dato 0.
      4. El microcontrolador recibe por usart los bits menos significativos del dato 0 y los guarda en la variable data\_lsb.
      5. El pc envía por usart los bits más significativos del dato 0.
      6. El microcontrolador recibe por usart los bits más significativos del dato 0 y los guarda en la variable data\_msb.
      7. El microcontrolador envía por I2C la condición de START.
      8. El microcontrolador ubica a la memoria (esclavo) a través de su dirección, mandado un código de control, su dirección y el bit de lectura o escritura de memoria.

Código de Control Dirección Esclavo

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 0 | A2 | A1 | A0 | 0 |

* + 1. La memoria baja el bit de ACK en caso de que la conexión exitosa haya sido exitosa, si no lo es, el ACK seguirá en alto y se necesitará repetir desde el ítem **vii**.
    2. El microcontrolador envía a memoria los bits más significativos de la dirección a escribir.
    3. La memoria baja el bit de ACK en caso de que él envió haya sido exitoso, si no lo es, el ACK seguirá en alto y se necesitará repetir desde el ítem **x**.
    4. El microcontrolador envía a memoria los bits menos significativos de la dirección a escribir.
    5. La memoria baja el bit de ACK en caso de que el envió haya sido exitoso, si no lo es, el ACK seguirá en alto y se necesitará repetir desde el ítem **xii**.
    6. El microcontrolador envía a memoria la variable data\_lsb.
    7. La memoria baja el bit de ACK en caso de que el envió haya sido exitoso, si no lo es, el ACK seguirá en alto y se necesitará repetir desde el ítem **xiv**.
    8. El microcontrolador envía a memoria la variable data\_msb.
    9. La memoria baja el bit de ACK en caso de que el envió haya sido exitoso, si no lo es, el ACK seguirá en alto y se necesitará repetir desde el ítem **xvi**.
    10. El microcontrolador envía por I2C la condición de STOP.
    11. S
    12. S
    13. El pc envía por usart “continue” para seguir enviando datos o “stop” para avisar que ya no hay más datos a enviar y este lo guarda en una variable “alert”.
    14. Se repetirá desde el ítem **iii** hasta el **xviii** por cada dato hasta el dato 999 y “alert” sea “continue”, si el contador es 1000 o “alert” sea “stop”.
    15. El microcontrolador envía por usart en ASCII al pc “finish”.
    16. El pc lee “finish” en ASCII para saber que ya se guardaron los 1000 datos en memoria.
  1. Enviar dese el microcontrolador al computador los 1000 datos de 16 bits almacenados en memoria.
     1. El pc envía por usart en ASCII al microcontrolador "read1000".
     2. El microcontrolador envía por usart en ASCII al pc “ready”.
     3. El microcontrolador envía por I2C la condición de START y se inicia el contador 2000 en 0.
     4. El microcontrolador ubica a la memoria (esclavo) a través de su dirección, mandado un código de control, su dirección y el bit de lectura o escritura de memoria.

Código de Control Dirección Esclavo

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 0 | A2 | A1 | A0 | 0 |

* + 1. La memoria baja el bit de ACK en caso de que la conexión exitosa haya sido exitosa, si no lo es, el ACK seguirá en alto y se necesitará repetir desde el ítem **iii**.
    2. El microcontrolador envía a memoria los bits más significativos de la dirección a leer que va a estar dada por el contador 2000.
    3. La memoria baja el bit de ACK en caso de que él envió haya sido exitoso, si no lo es, el ACK seguirá en alto y se necesitará repetir desde el ítem **vi**.
    4. El microcontrolador envía a memoria los bits menos significativos de la dirección a leer que va a estar dada por el contador 2000.
    5. La memoria baja el bit de ACK en caso de que él envió haya sido exitoso, si no lo es, el ACK seguirá en alto y se necesitará repetir desde el ítem **viii**.
    6. El microcontrolador envía a la memoria código de control, su dirección y el bit de lectura o escritura de memoria.

Código de Control Dirección Esclavo

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 0 | A2 | A1 | A0 | 1 |

* + 1. La memoria baja el bit de ACK en caso de que la conexión exitosa haya sido exitosa, si no lo es, el ACK seguirá en alto y se necesitará repetir desde el ítem **x**.
    2. La memoria envía al microcontrolador el dato de la dirección solicitada.
    3. La memoria baja el ACK para indicar al microcontrolador no ha acabado la transmisión e incrementa el puntero de dirección de lectura.
    4. El microcontrolador recibe y almacena la información en data\_lsb cuando el contador 2000 es par y en data\_msb cuando el contador 2000 es impar, e incrementa el contador 2000.
    5. El microcontrolador lee el contador 2000, en caso de ser impar, el microcontrolador envía por I2C la condición de STOP, si no repetirá desde el ítem **xii** pero con la nueva dirección del puntero.
    6. El microcontrolador envía por usart al pc la variable data\_msb.
    7. El microcontrolador revisa que el buffer este vacío y envía por usart al pc la variable data\_lsb.
    8. El pc envía por usart “continue” para seguir enviando datos o “stop” para avisar que ya no hay más datos a enviar y este lo guarda en una variable “alert”.
    9. El microcontrolador revisa el contador 2000 si es igual a 2000 o que “alert” sea “stop” envía por usart en ASCII al pc “finish”, si el contador no es 2000 y “alert” sea “continue” y repite desde el ítem **iii**.
    10. El pc lee “finish” en ASCII para saber que ya acabo la transmisión de los 1000 datos.
  1. Recibir desde el microcontrolador al PC, uno de los 1000 datos almacenados en memoria
     1. El pc envía por usart en ASCII al microcontrolador "readdata".
     2. El microcontrolador envía por usart en ASCII al pc “ready”.
     3. El pc envía por usart los bits menos significativos del número del dato a enviar.
     4. El microcontrolador recibe por usart los bits menos significativos del número del dato a enviar y los guarda en la variable number\_data\_lsb.
     5. El pc envía por usart los bits menos significativos del número del dato a enviar.
     6. El microcontrolador recibe por usart los bits menos significativos del número del dato a enviar y los guarda en la variable number\_data\_msb
     7. El microcontrolador envía por I2C la condición de START.
     8. El microcontrolador ubica a la memoria (esclavo) a través de su dirección, mandado un código de control, su dirección y el bit de lectura o escritura de memoria.

Código de Control Dirección Esclavo

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 0 | A2 | A1 | A0 | 0 |

* + 1. La memoria baja el bit de ACK en caso de que la conexión exitosa haya sido exitosa, si no lo es, el ACK seguirá en alto y se necesitará repetir desde el ítem **iii**.
    2. El microcontrolador envía a memoria los bits más significativos de la dirección a leer.
    3. La memoria baja el bit de ACK en caso de que él envió haya sido exitoso, si no lo es, el ACK seguirá en alto y se necesitará repetir desde el ítem **vi**.
    4. El microcontrolador envía a memoria los bits menos significativos de la dirección a leer.
    5. La memoria baja el bit de ACK en caso de que él envió haya sido exitoso, si no lo es, el ACK seguirá en alto y se necesitará repetir desde el ítem **viii**.
    6. El microcontrolador envía a la memoria código de control, su dirección y el bit de lectura o escritura de memoria.

Código de Control Dirección Esclavo

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 0 | A2 | A1 | A0 | 1 |

* + 1. La memoria baja el bit de ACK en caso de que la conexión exitosa haya sido exitosa, si no lo es, el ACK seguirá en alto y se necesitará repetir desde el ítem **x**.
    2. La memoria envía al microcontrolador el dato de la dirección solicitada.
    3. La memoria baja el ACK para indicar al microcontrolador no ha acabado la transmisión e incrementa el puntero de dirección de lectura.
    4. El microcontrolador recibe y almacena la información en data\_lsb cuando el contador es 0 y en data\_msb cuando el contador es 1, e incrementa el contador.
    5. El microcontrolador lee el contador, en caso de estar en 2, el microcontrolador envía por I2C la condición de STOP, si no repetirá desde el ítem **xii** pero con la nueva dirección del puntero.
    6. El microcontrolador envía por usart al pc la variable data\_msb.
    7. El microcontrolador revisa que el buffer este vacío y envía por usart al pc la variable data\_lsb.
    8. El microcontrolador envía por usart en ASCII al pc “finish”.
    9. El pc lee “finish” en ASCII para saber que ya acabo la transmisión del dato solicitado.