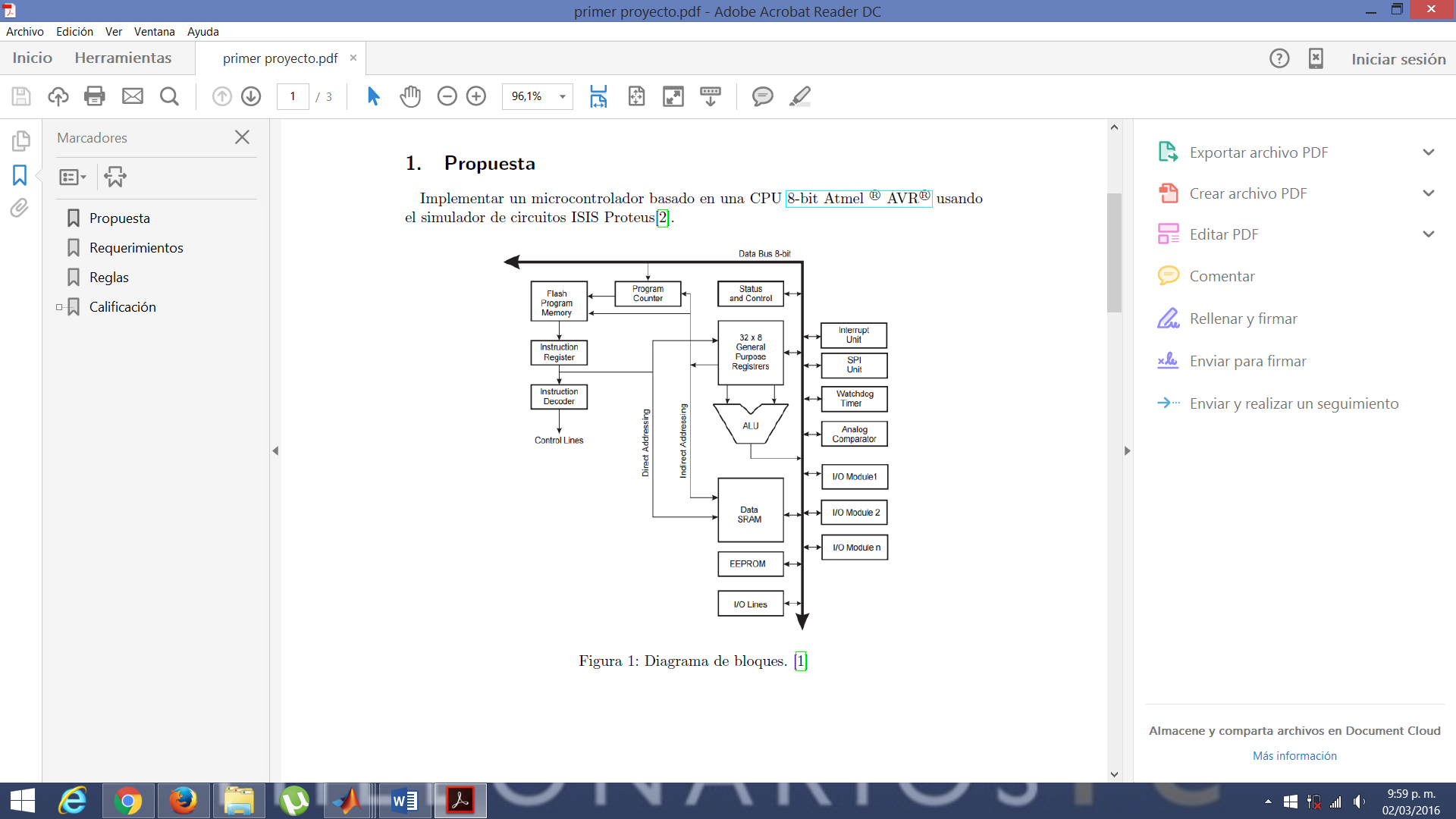
Proyecto 1

Objetivo general:

Implementar un microcontrolador basado en una CPU 8-bit Atmel AVR usando el simulador de circuitos ISIS Proteus.

Objetivos específicos:

* Implementar los módulos de entrada y salida E/S
* Diseñar y adicionar el módulo de memoria flash o memoria del programa al módulo general
* Diseño y adición de la memoria de datos SRAM al módulo general.
* Diseño y adición de la CPU como cerebro del módulo general.



Para realizar el proyecto 1, comenzamos con la previa visualización del diagrama de bloques del microcontrolador, con el cual se establece los módulos de este que se implementaran y su correspondiente forma.

* Puertos de E/S

Algo de recalcar en este momento es que dentro del pin y posterior puerto habrá tres acciones que se podrá hacer, estas son:

* Configuración
* Escritura
* Lectura

El pin está compuesto por tres componentes principales

* ***DDRx*** a este componente se le estableció la dirección (hexadecimal)
* ***PORTx*** se le asignó la dirección
* ***PIN*** se le asignó la dirección

Donde

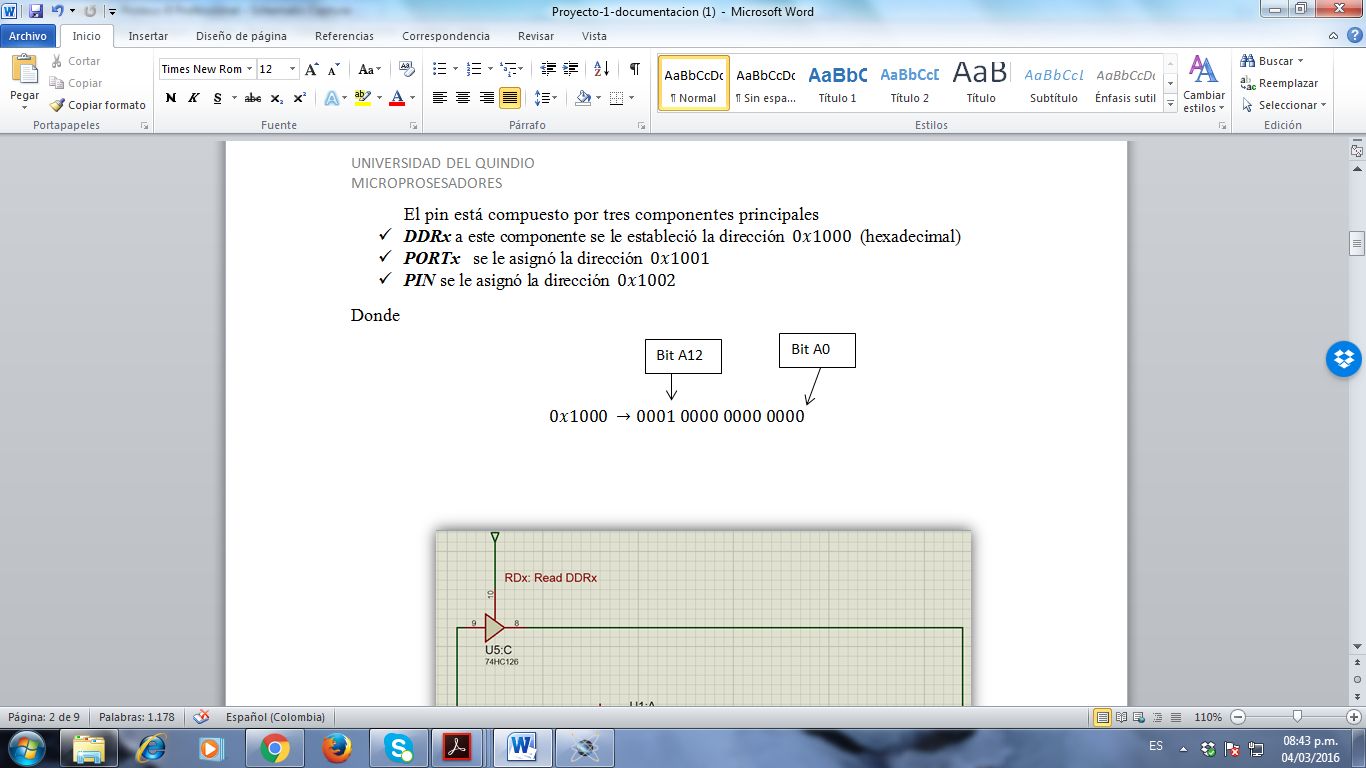


Figura.2.direccion DDRx

Lo anterior nos muestra que algunos de los parámetros que debemos tener en cuenta para la clasificación de que acción realizara (escribir o leer) , estos son A12 y A0, y esto constituye el acceso al DDRx, pero algunos se preguntaran ¿ pero el DDRx tiene dos elementos un RDx y un WPx como puedo acceder a ellos sí parecen tener la misma dirección?, la respuesta a esto es sencilla, la diferencia en la dirección es que debemos tener en cuenta también la decisión del procesador que indica que leerá o escribirá. ( ó ).

A partir de lo anterior se puede inferir que para generalizar la selección, acorde a varios parámetros de cada dirección como y además de la , se deberá tener presente principalmente los bits A12, A0 y A1 además de los y ya que son los parámetros que difieren para cada parámetro de activación en elementos del circuito pin o posteriormente ya constituido como puerto.

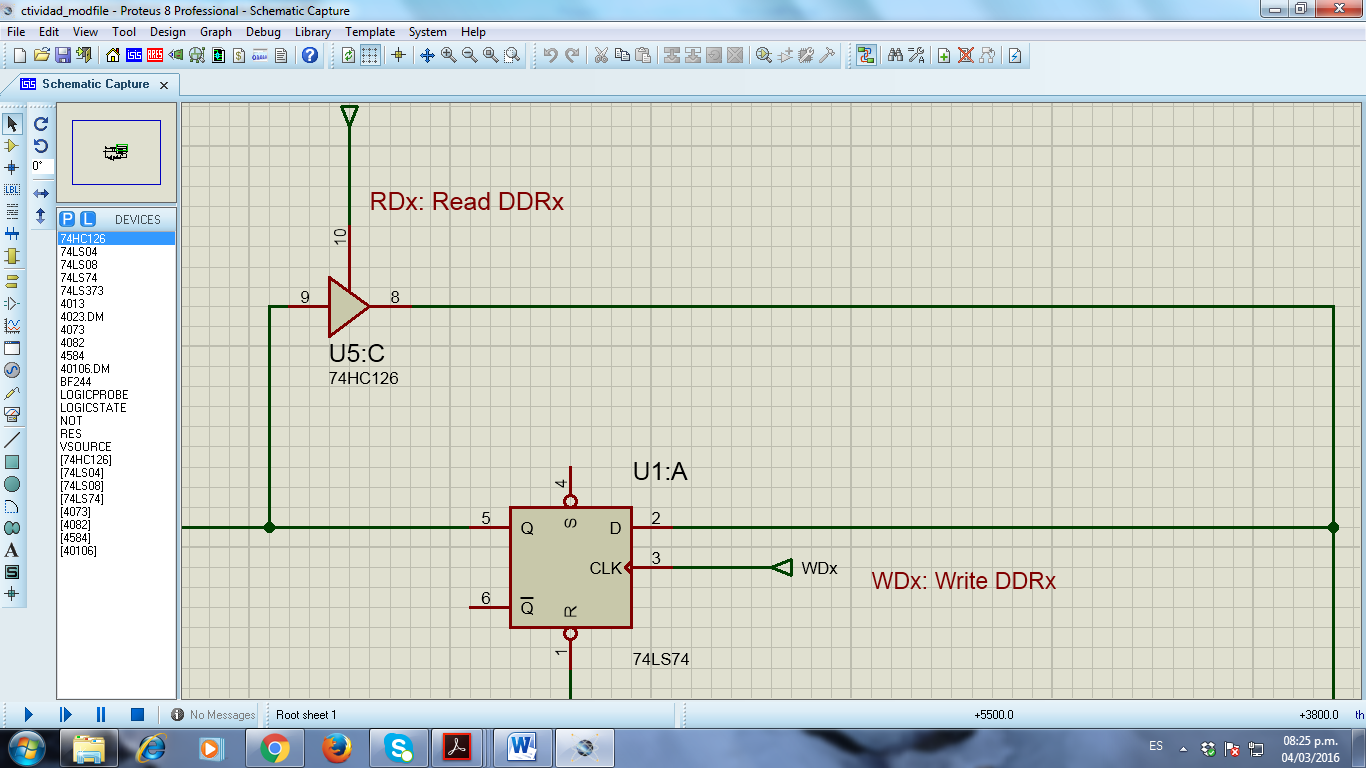


Figura.2. DDRx

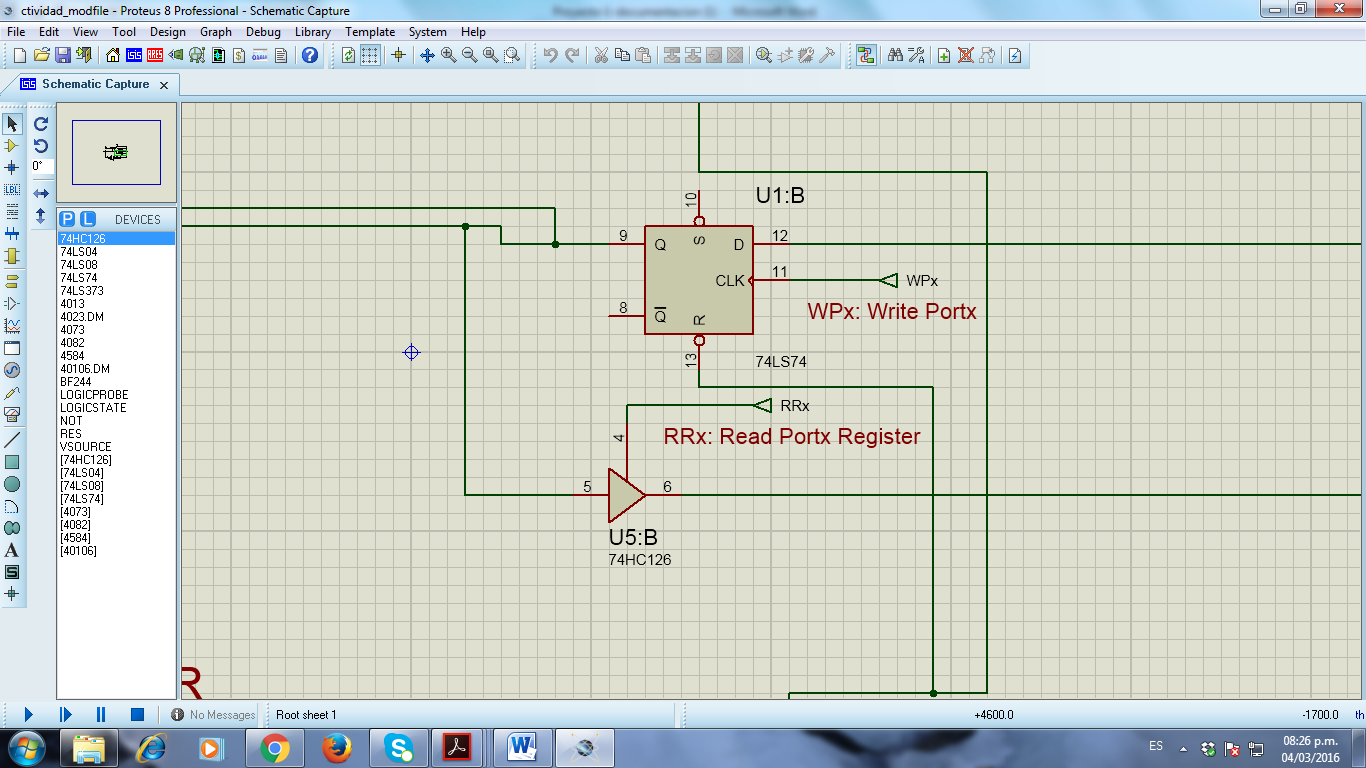


Figura.3.PORTx

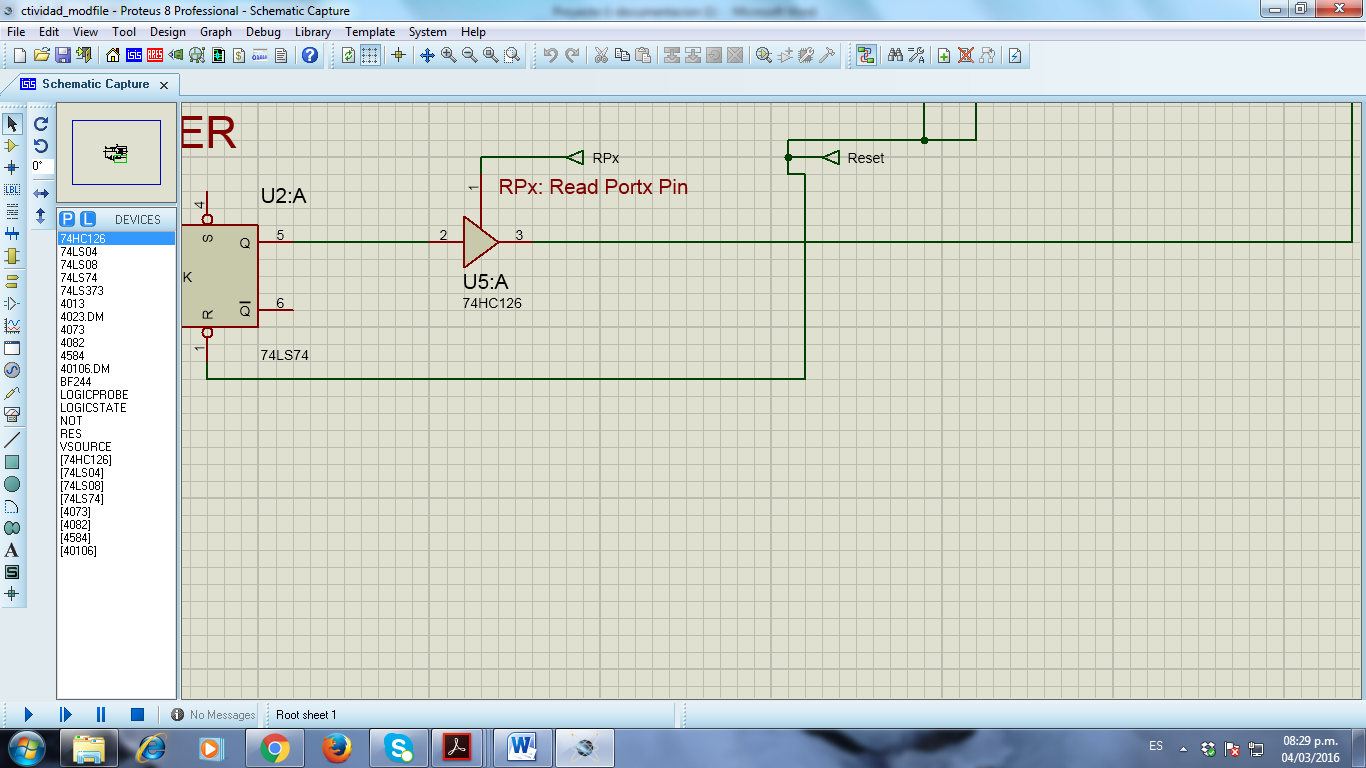


Figura.4.PIN

Este fue el primer módulo a diseñar con la asesoría de los profesores de la respectiva clase, consistió en primer lugar en diseñar un pin del puerto que se quería hacer, posteriormente de acuerdo a ese pin-base implementar el puerto con los 8 pines, habiendo comprobado el funcionamiento previo del pin base. Para ello se utilizaron las herramientas de proteus que permiten diseñar los encapsulados para guardar proporciones en el módulo general.

En primer lugar se planteó que era necesario tener unas direcciones que entregara el procesador por medio de las cuales pudiera acceder o controlar las acciones de cada pin como se ha mencionado anteriormente. De esta forma se estableció que había 5 elementos de un pin a controlar mediante componentes de dirección para escribir en el pin o para leer del algún tipo de información del pin. Las direcciones dadas por salidas del procesador llegan entonces a un conjunto de compuertas las cuales hacen en este caso de seleccionador, es decir, de acuerdo a la dirección que envió el procesador se activara la salida en alguna de las compuertas, de este modo alguna terminal se configurara como lectura o como escritura en cualquiera de los casos el procesador escribirá un 1 o un 0 o leerá alguno, de estos dos valores.

Es importante saber que dependiendo de la dirección que llegue solo se activara una sola o ninguna es decir el procesador podrá escribir en el puerto o leer del puerto pero no ambos al tiempo esto ocasionaría un error lógico en tiempo de ejecución.

Por ejemplo:

La figura 5 corresponde a la compuerta de selección para la terminal de activación (RDx) (ver figura 2) de una compuerta tri-state (si le llega a la terminal de selección un “uno” actúa como cortocircuito, si le llega un “cero” es alta impedancia) . Las configuraciones en la entrada se deben a que se empleó el método de mintérminos para saber ante que entradas se nos activaría.

Como se dijo en párrafos anteriores al RDx y al RPx se accede mediante a dirección además de seleccionar ya que va a leer (entonces estará en alto)

Tabla1. Niveles necesarios para activar RDx

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | A12 | A1 | A0 |  |  |
| Nivel lógico | 1 | 0 | 0 | 1 | 0 |
| minterminos | A12 |  |  |  |  |
| Entradas a AND | A12\*\* \* \* | | | | |
|  |

Para proceder a deducir la compuerta a través de minterminos (producto de las entradas) se tiene en cuenta que si hay un valor de 1 bajo la variable esta se representara con el mismo nombre de la variable si hay un cero bajo la variable se representara mediante el nombre de la variable negada al final el resultado es el producto de todas las entradas (AND) como se observa en la misma tabla y sustentado en la figura 5.

Las demás direcciones sustentaron de igual forma la selección de las compuertas como se muestra en la siguiente tabla.2.

Tabla1. Direcciones de acceso y de deducción de compuertas de control.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | A12 | A1 | A0 |  |  |
| RDx | 1 | 0 | 0 | 1 | 0 |
| WDx | 1 | 0 | 0 | 0 |  |
| RRx | 1 | 0 | 1 | 1 | 0 |
| WPX | 1 | 0 | 1 | 0 | 1 |
| RPX | 1 | 1 | 0 | 1 | 0 |

Las demás compuertas se sacaron de la misma manera que la explicada en la tabla 1.

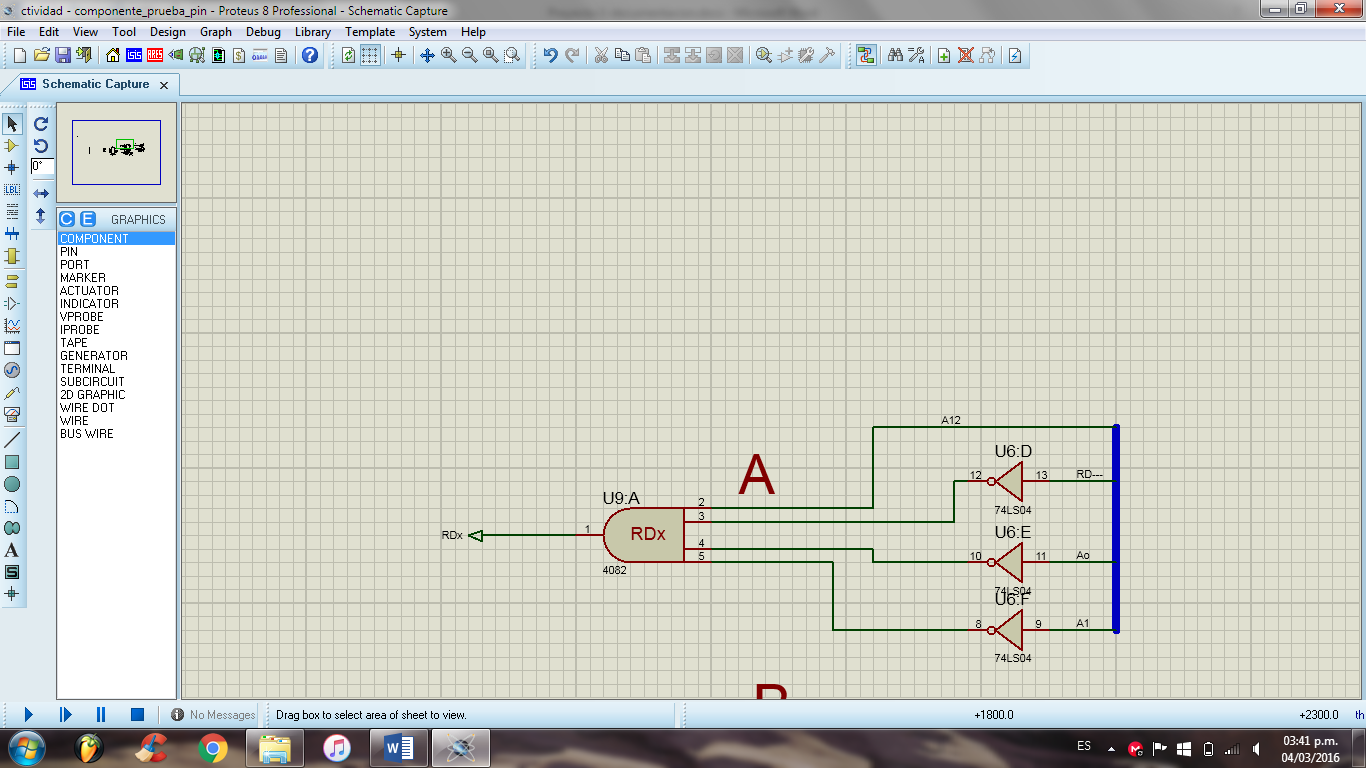


Figura.5. compuerta de control para tristate

De acuerdo a la descripción del párrafo anterior se planteó la compuerta de control para la terminal RDx la cual permite al procesador leer lo que haya en la salida del flip-flop (terminal Q del WDx)

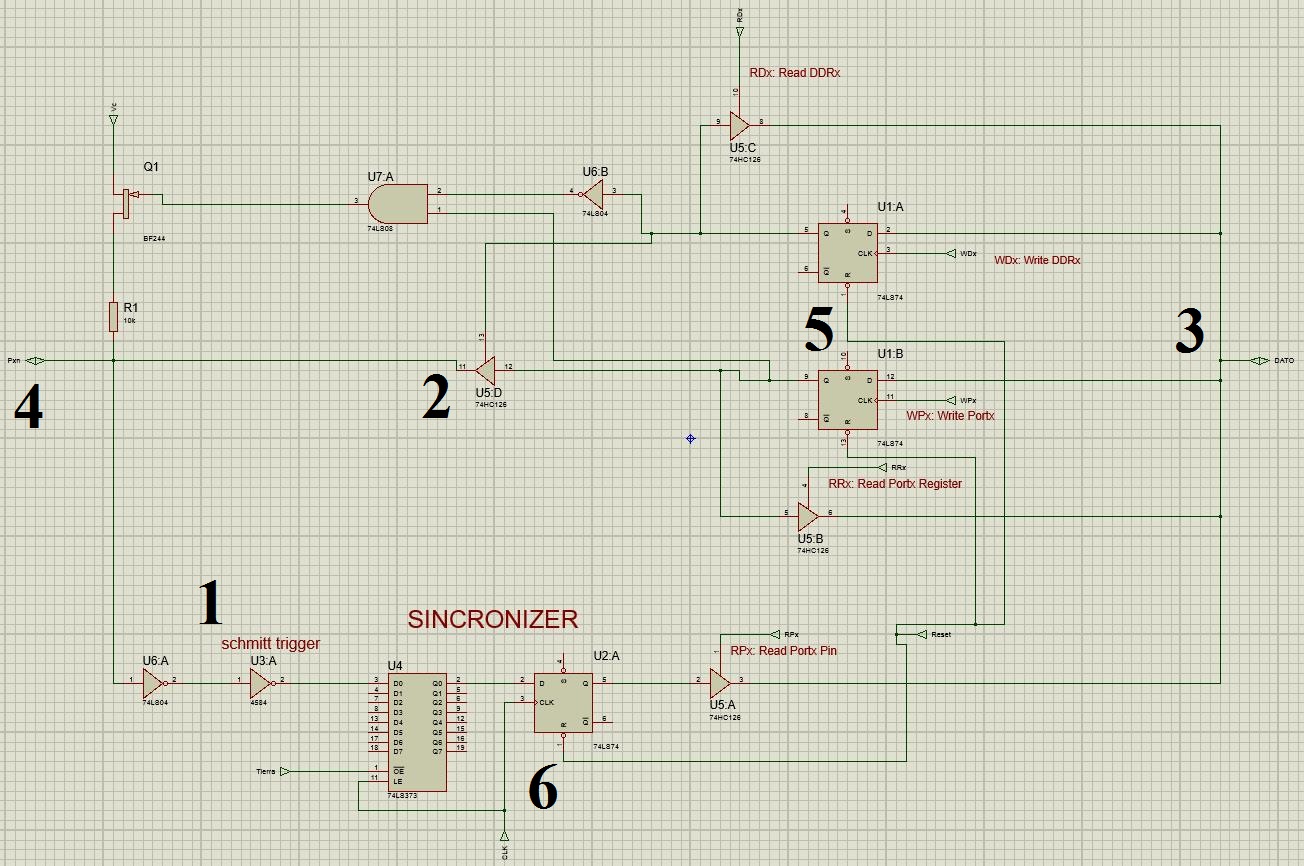


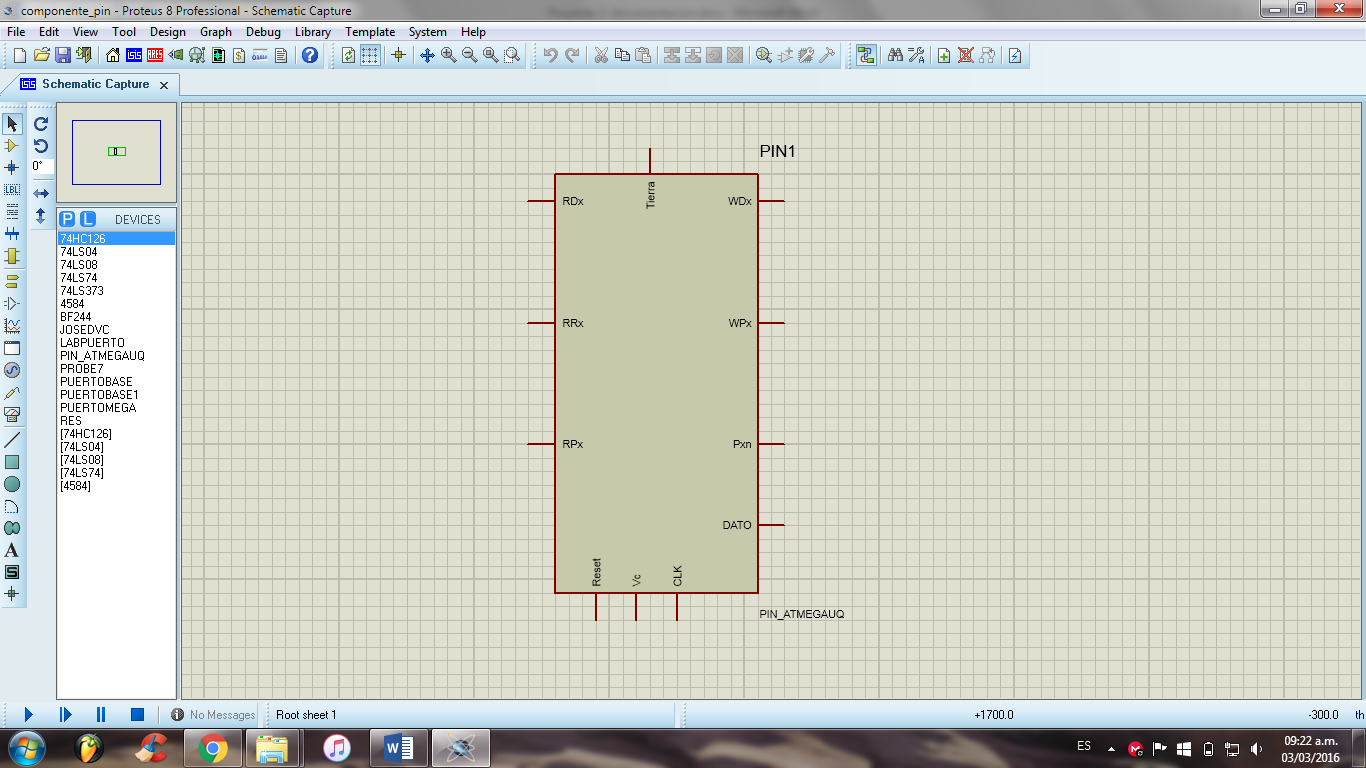
Fig. 6 Puerto E/S, estructura del componente.

1. El Schmitt trigger rectifica la señal por medio de la histéresis conservando el nivel lógico hasta que no haya un cambio muy brusco en ella. Esto previene el ruido presente que podría afectar a la señal original.
2. Este tristate actua como un switch respecto al pin pxn, ya que hace que este actue como entrada o salida dependiendo de la señal logica recibida. Por ejemplo si el tristate recibe una señal en 0 quedará como switch abierto haciendo que el pin Pxn se convierta en entrada mientras que con una señal 1 seria todo lo contrario.
3. En cuanto a los pines RRx, RPx, RDx, WDx y WPx, estos controlan el manejo de datos en el registro de cada direccion, ya sea, almacenando datos u obteniendo una lectura de estos mismos.

Los pines con la letra R son los encargados de hacer la lectura y los que tienen la letra W escribirían los datos.

1. Pxn es bidireccional por lo que por este es por donde saldrán o entraran datos, dependiendo de la configuración que se establezca en los registros.
2. Los flip flops con referencia 74LS7, se encargan de pasar cada dato recibido en cada flanco de reloj dependiendo de lo que el procesador establezca, por ejemplo para WDx este será de escritura.
3. El Latch y el flip flip 74LS74 no son dominados por el procesador, ya que, tienen reloj independiente y estos estarán pasando datos del Pxn constantemente, si el RPx se activa por el procesador se leería el dato proveniente del Pxn.

Mediante el uso de las compuertas se comprobó el funcionamiento de la estructura del componente, para prueba se eligieron las direcciones 1000, 1001 y 1002 ya que el circuito se divide en tres componentes que son el DDR, el PORT y el PIN. La activación de cada uno dependía de los valores de A12, A0 y A1 además del valor del procesador da a RD y WR si se quiere leer o escribir, por lo que con el procedimiento de min términos se llegó a un circuito combinacional que nos permitió el control de los módulos internos anteriormente descritos.



Se realizó el componente el cual estaba estructurado por el circuito de la fig. 6 con los pines de entrada y salida necesarios para su funcionamiento (fig.7), seguido a esto se corroboró el funcionamiento de dicho componente de la misma forma como anteriormente se describió (fig. 8).

Fig 7. Integrado del componente

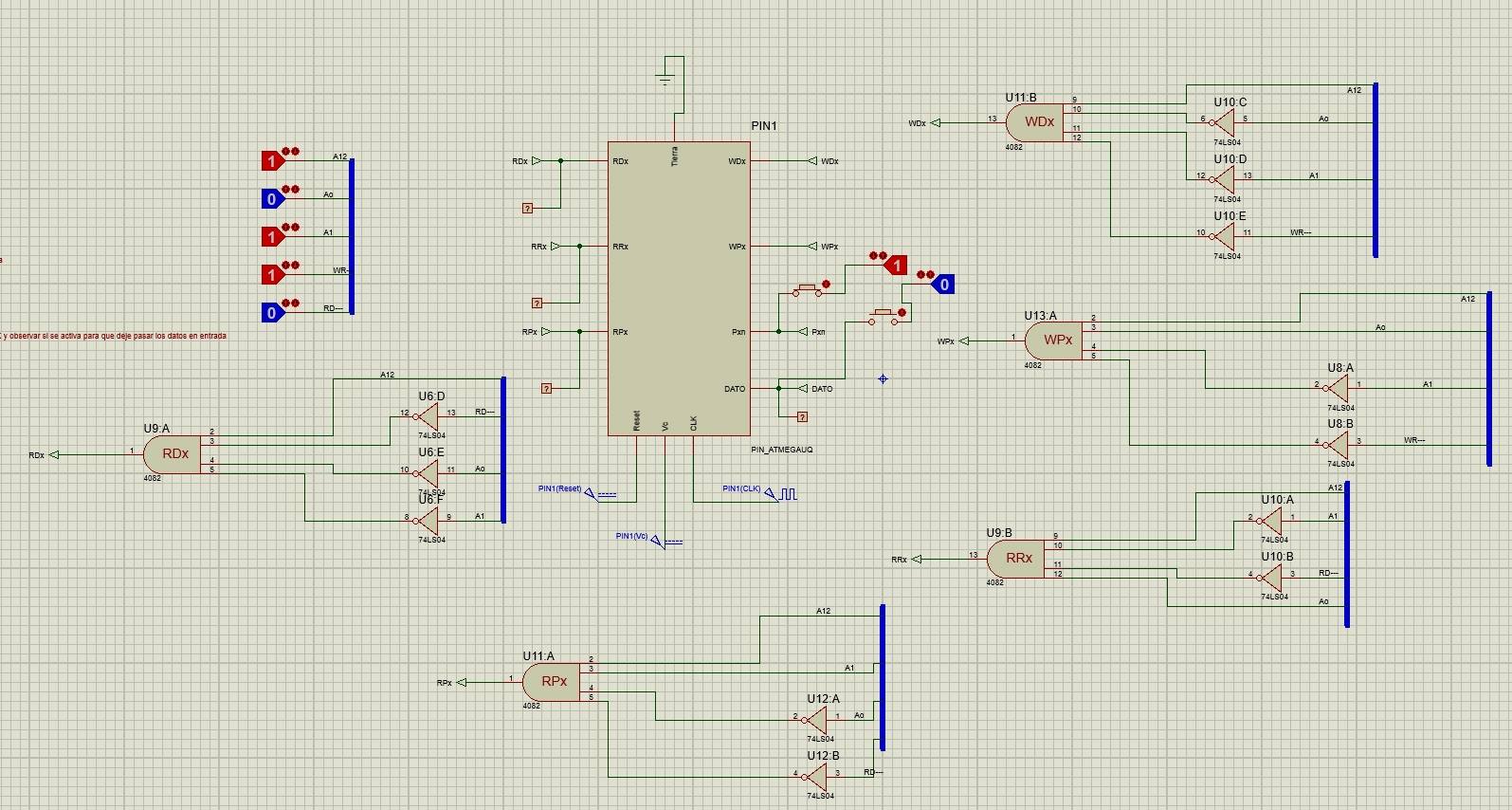


Fig 8. Funcionamiento del integrado del componente

Después de haber corroborado el funcionamiento del componente se completaron 8 pastillas indicando los pines comunes y los pines independientes para cada una (fig. 9), al igual que las compuertas de direccionamiento para empaquetar esto en una sola pastilla la cual sería el puerto, en la figura 6 se puede apreciar detalladamente cada uno de los pines de la pastilla.

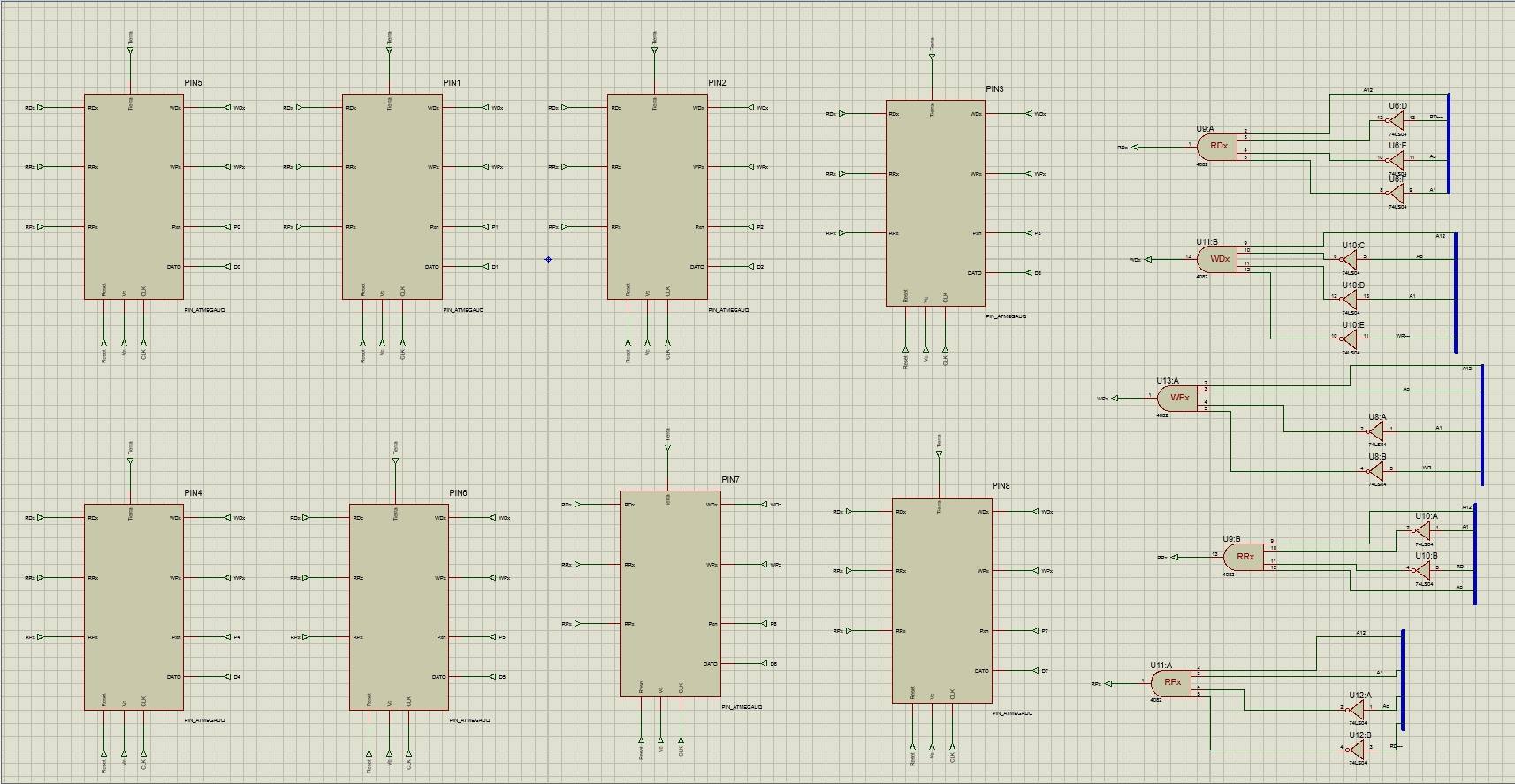


Fig 9. Estructura que compone el puerto

Para cada componente de pin, se estableció una entrada o salida de datos llamada P que iba desde p0 hasta p7, al igual que una entrada y salida de datos, D que también iba desde D0 hasta D7 (las RDx de cada pastilla eran comunes entre si, así como las demás no descritas). además de integrar las compuertas correspondientes al circuito combinacional, también se incorporó una señal de reloj CLK que activa los registros, un Vc de alimentación, un reset que sería común para los flip flops. Y A12, A0, A1, WR y RD que corresponden a los valores por los cuales se activan las configuraciones de las compuertas, de acuerdo a lo que se tenga en el procesador.

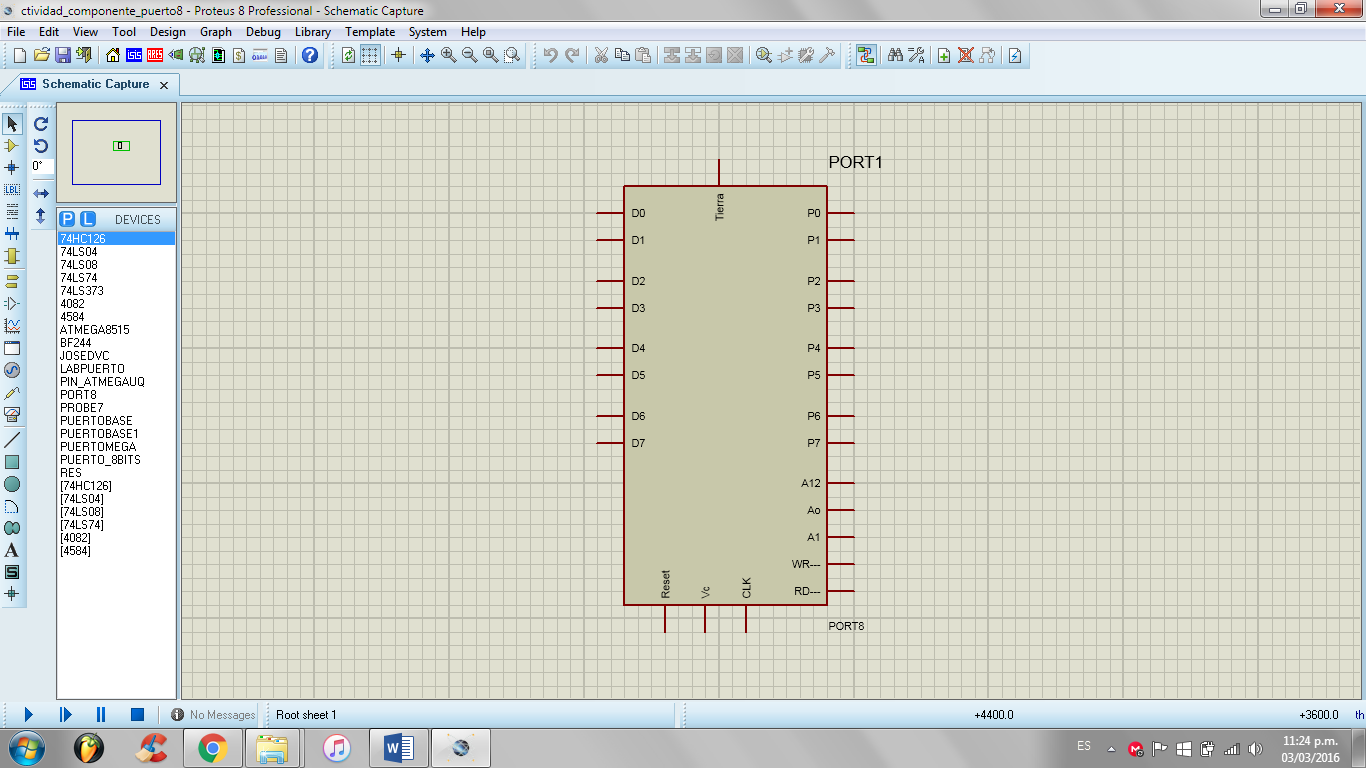


Fig 10. Puerto de 8 bits.

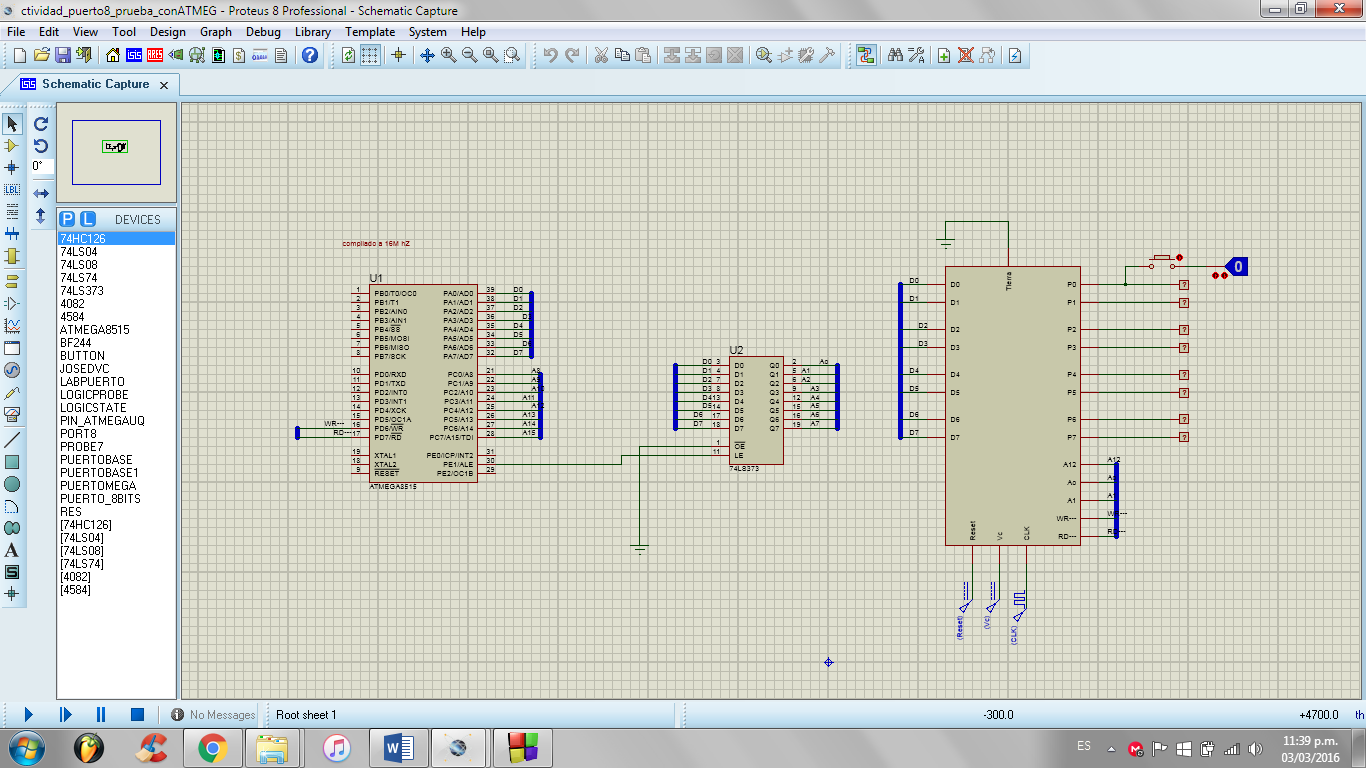


Fig 11. Funcionamiento del puerto de 8 bits.

Para hacer el primer ensayo de funcionamiento de la pastilla se conectó el puerto de 8 Bits al procesador ATMega8515.

A partir del código .hex que se le agrega al procesador:

En la la librería port.h se encuentran valores definidios, estos valores están establecidos como DDR\_A, DDR\_B, PIN\_A, PIN\_B, PORT\_B, etc; a los cuales se les daba un numero de una dirección, por ejemplo DDR\_A en el archivo port.h vale 0x2000, si se escribe el DDR en los puertos, estos van a configurar si el puerto queda como entrada o salida, en el código actual el puerto A esta configurado como entrada y el B como salida.

En el ciclo infinito While(1) se lee el pin A, es decir, el puerto configurado como entrada, el valor obtenido en la lectura se manda a la función “secuencias”, este proceso se estaría llevando a cabo para los 8 pines obteniendo un valor ya de 8 bits dependiendo de la dirección que se le ponga.

La función “secuencias” tiene varios tipos de casos donde se valida dependiendo del valor que reciba, ya sea, un valor en 0 o en 1; por ejemplo en el case 4, lo que se hace es leer el valor que tiene el puerto, y cuando se lee el puerto lee el valor que ya había obtenido y le incrementa un uno para obtenerlo otra vez, en el case 8 se hace lo mismo, la diferencia es que no incrementa sino que decrementa.

Para los case 1 y 2 varia funcionamiento, ya que, no incrementaría sino que se desplazaría y se pondría de nuevo en la misma dirección.

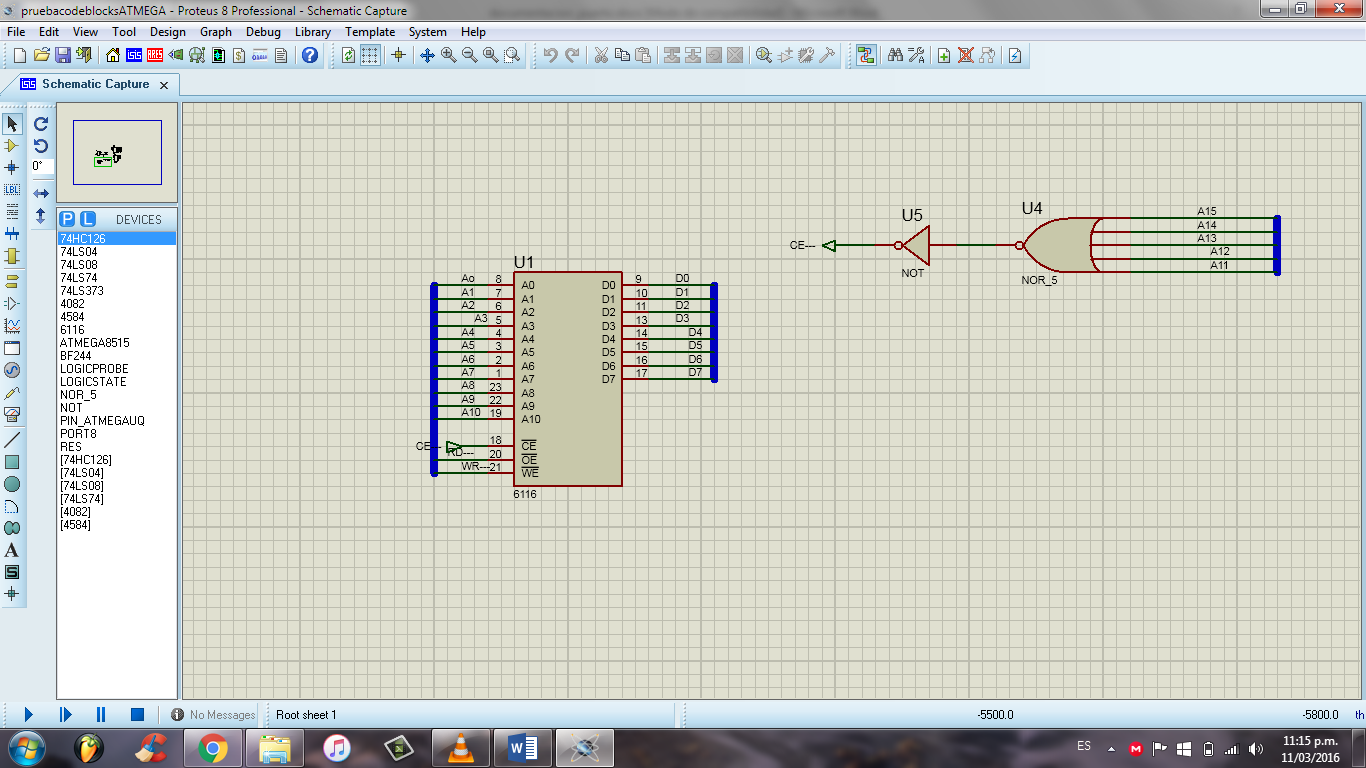


Fig 12. Conexión de la RAM al puerto.

Para esta se conexión se manejan 11 direcciones aparte de 0x2000 y 0x4000 con sus respectivas subdirecciones, primero porque la RAM va de 0 a 10 siendo su función almacenar, entonces para eso se pone la compuerta, para que se almacene solo cuando se le da la orden.

CE se activa en 1, y la operación realizada en la parte de las compuertas lo que hará es seleccionar una de las 11 direcciones establecidas de acuerdo al número de A0 a A10, así mismo CE activa la memoria para leer o escribir.

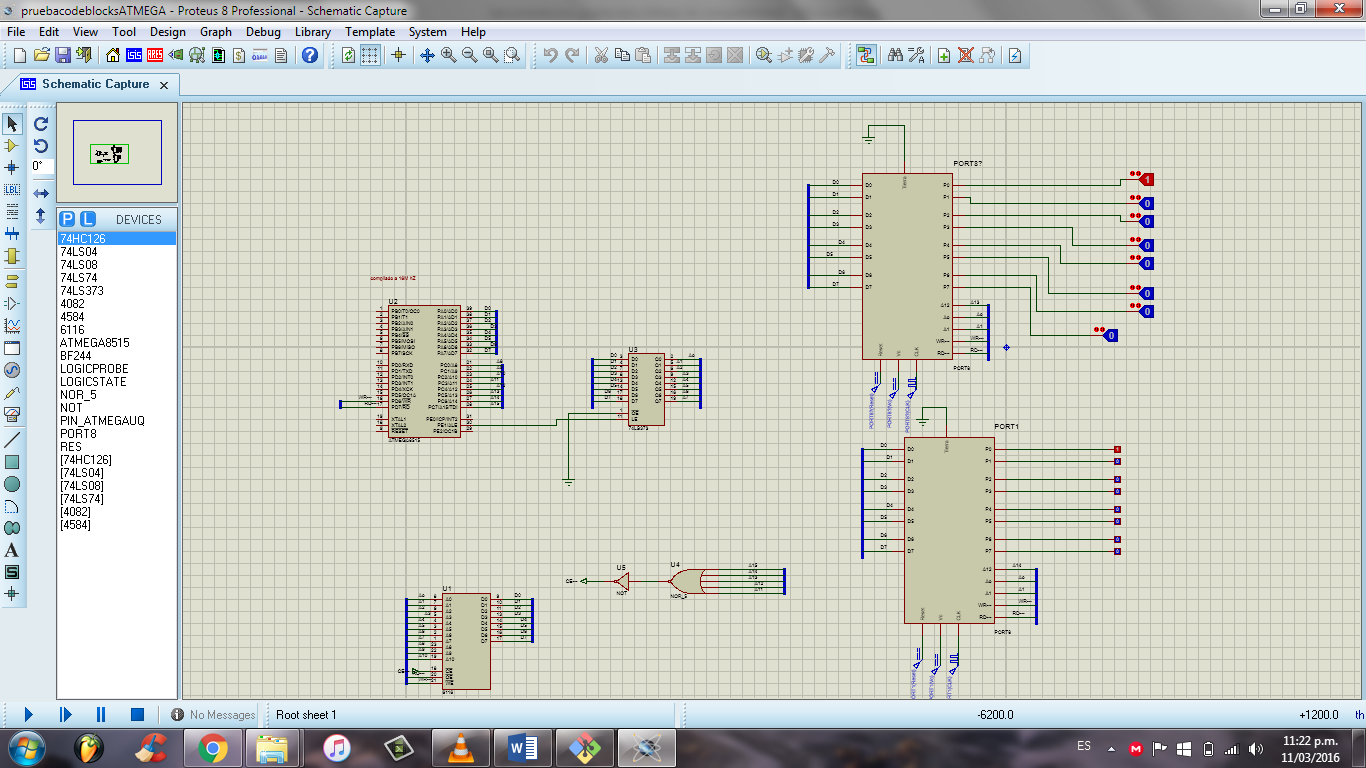


Fig 13. Conexión completa de la RAM al puerto.

Para facilitar la interpretación del programa implementado se establecieron algunas definiciones relacionados con el montaje.

PREESCALADOR: Tiene como función dividir la frecuencia dependiendo del valor ingresado en el OCR.

TCCR: Este pin de registro nos permite darle el funcionamiento al temporizador implementado siendo este por decirlo así, un pin de registro de control.

CLKT: La señal de reloj proveniente del preescalador.

TCNT: Contiene los datos actuales del temporizador haciendo un registro de ellos.

OCR: Tiene la función de determinar el ancho de pulso mediante comparaciones con valores registrados provenientes del temporizador.

OC: Pin de salida del temporizador siendo por este pin donde se generan las señales PWM.

Count: Cumple la función de incrementar o decrementar el valor del temporizador de a uno.

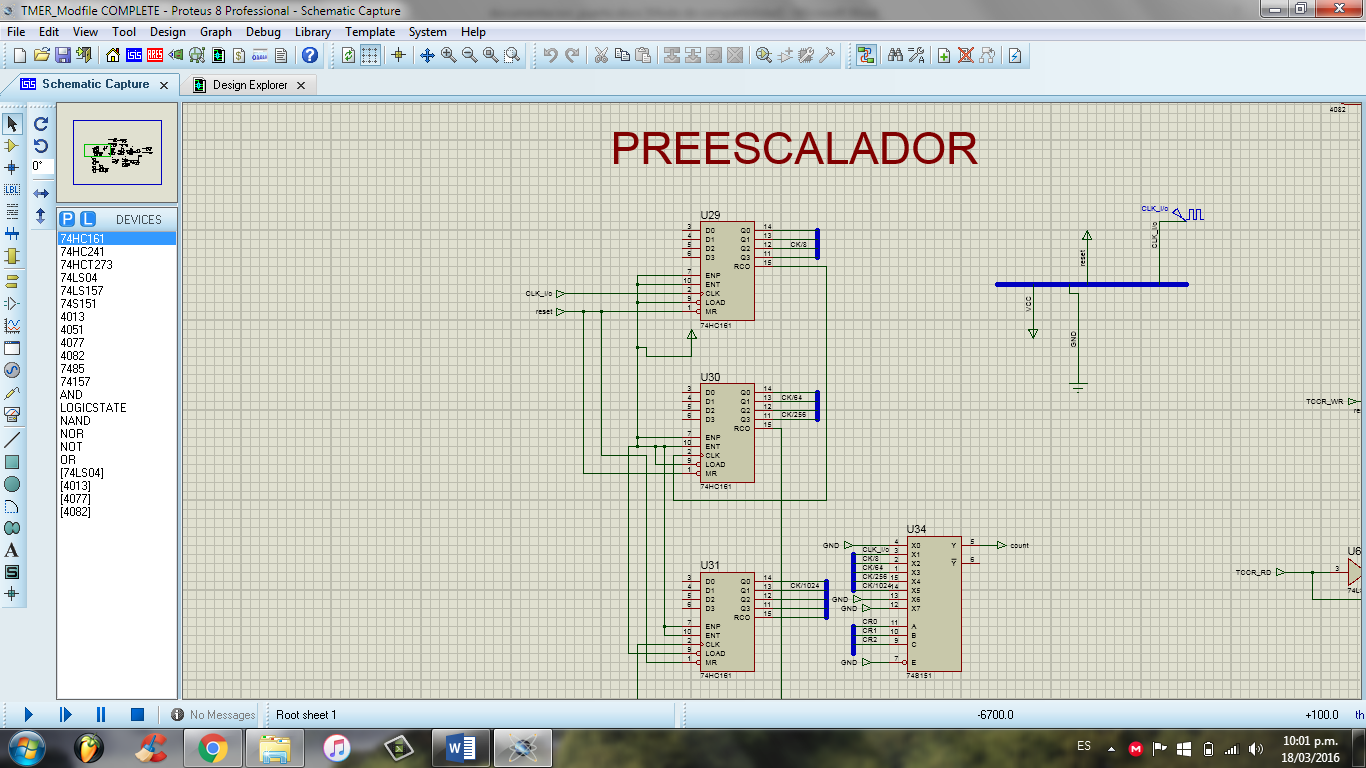


Fig 14. Montaje preescalador del Timer.

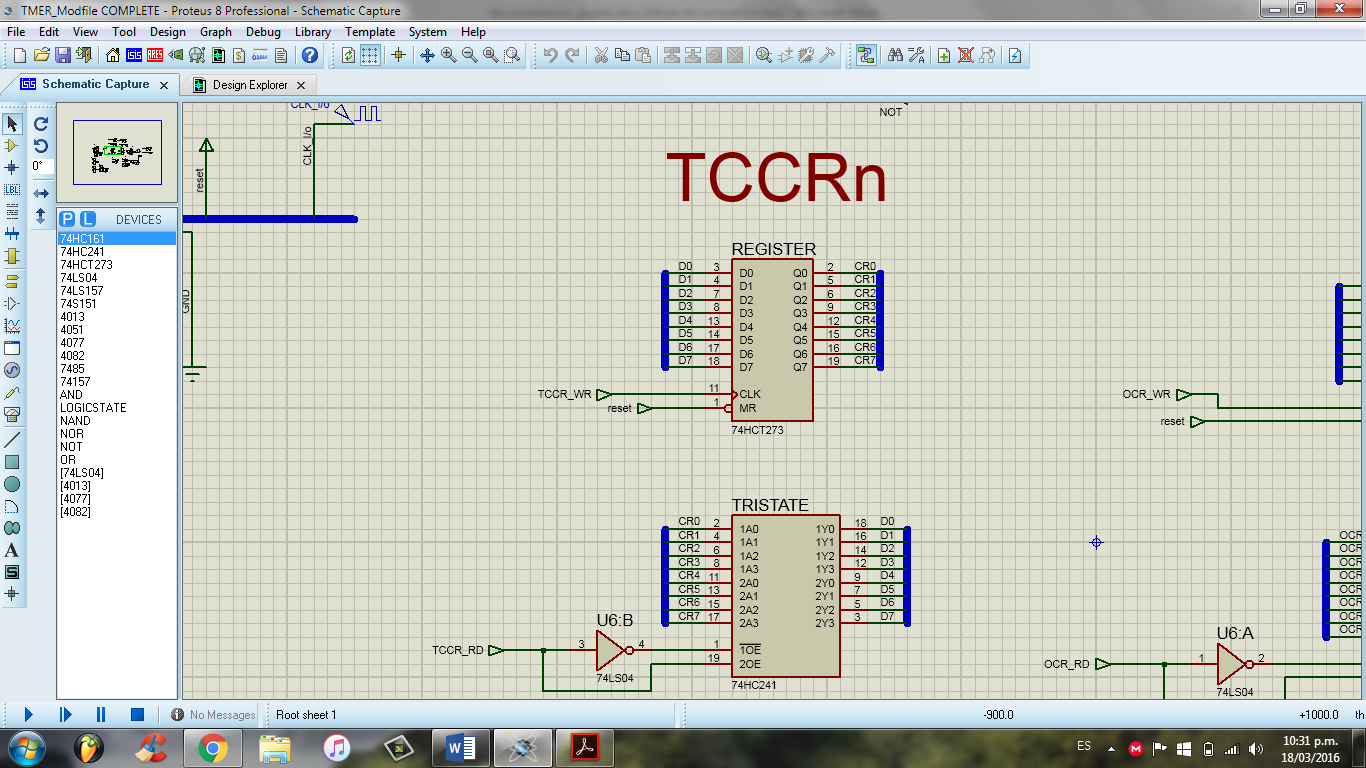


Fig 15. Registro TCCR.

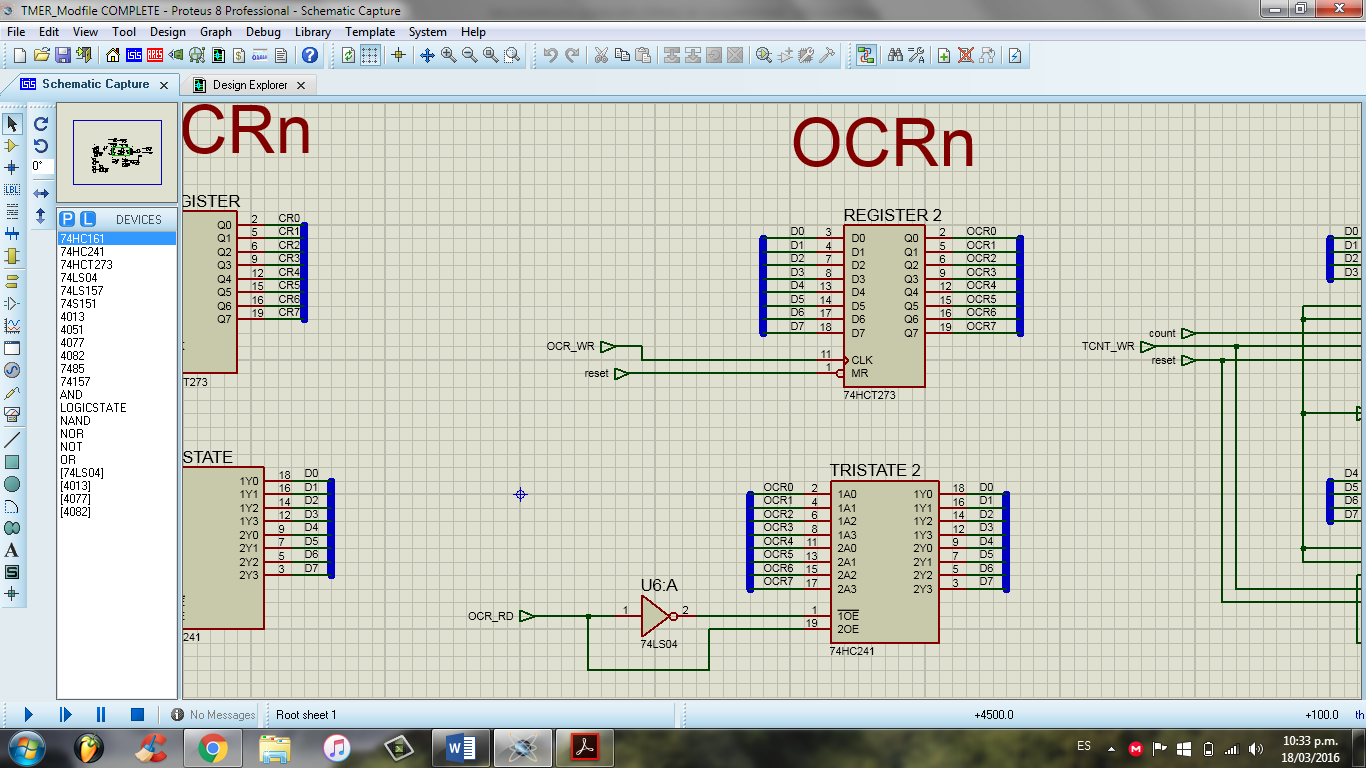


Fig 16. Registro OCR.

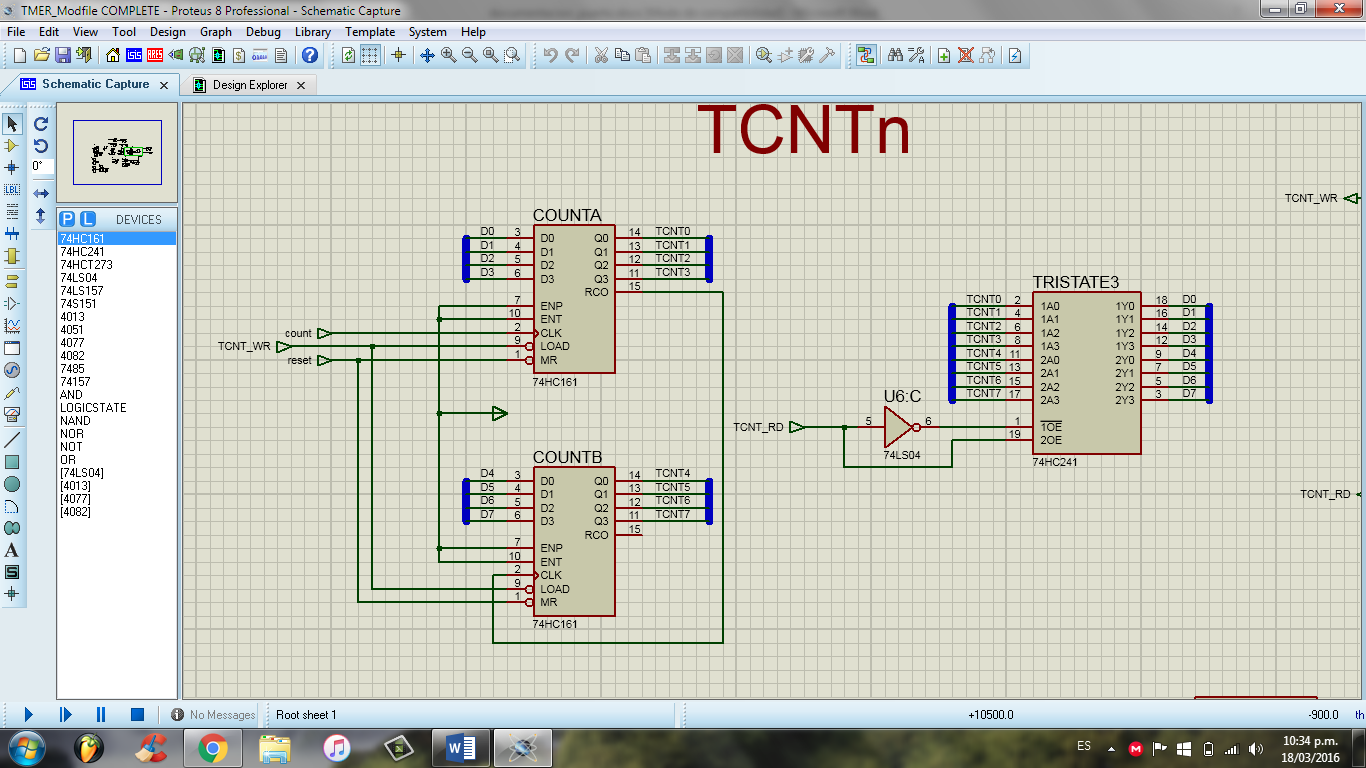


Fig 17. Registro TCNT.

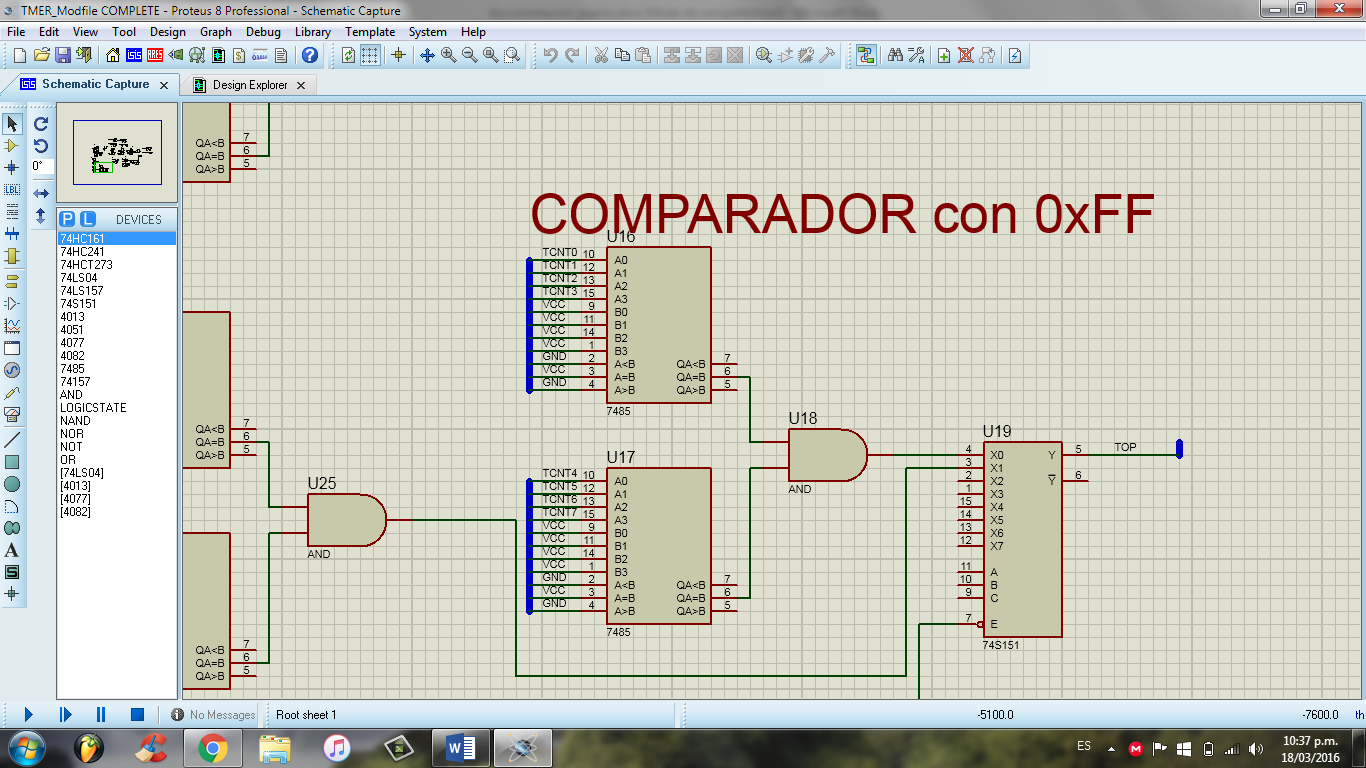


Fig 18. Circuito comparador del registro TCNT con 1.

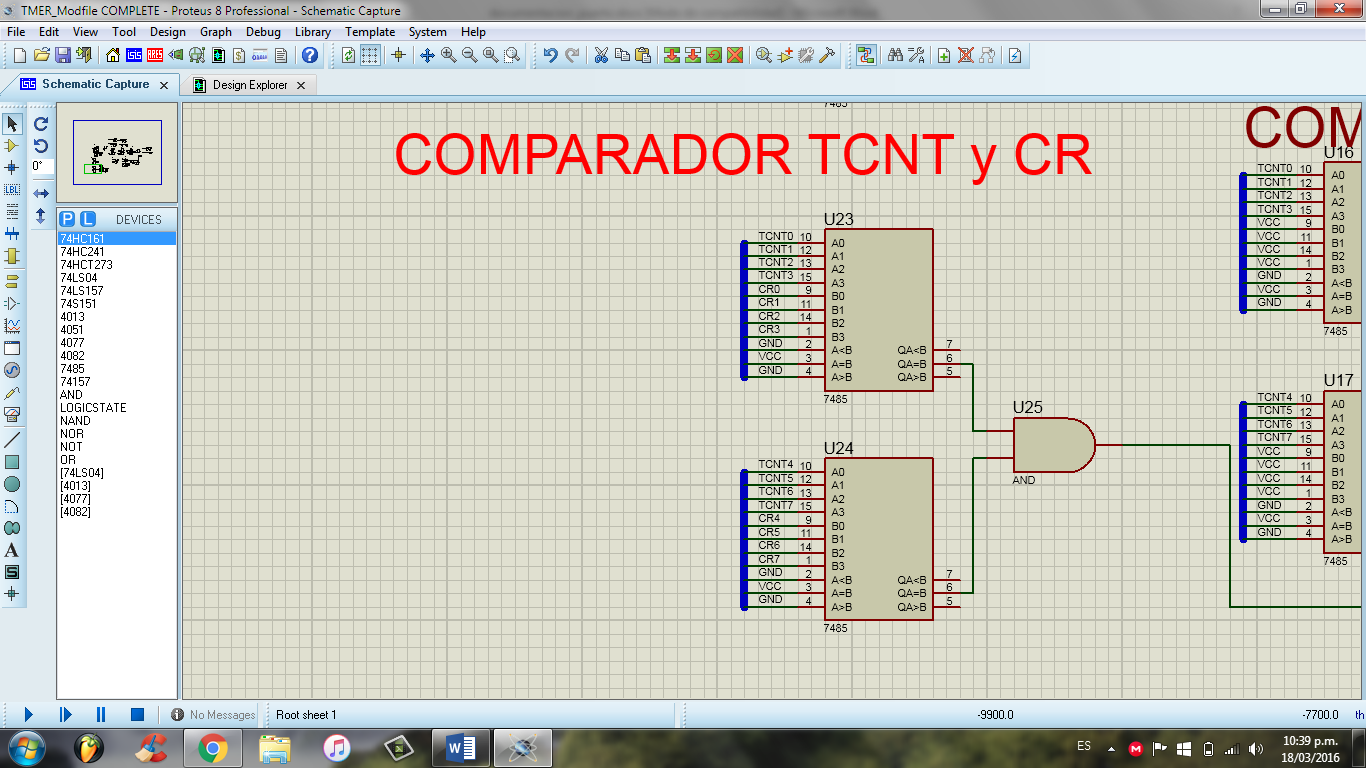


Fig 19. Circuito comparador del registro TCNT con el registro CR.

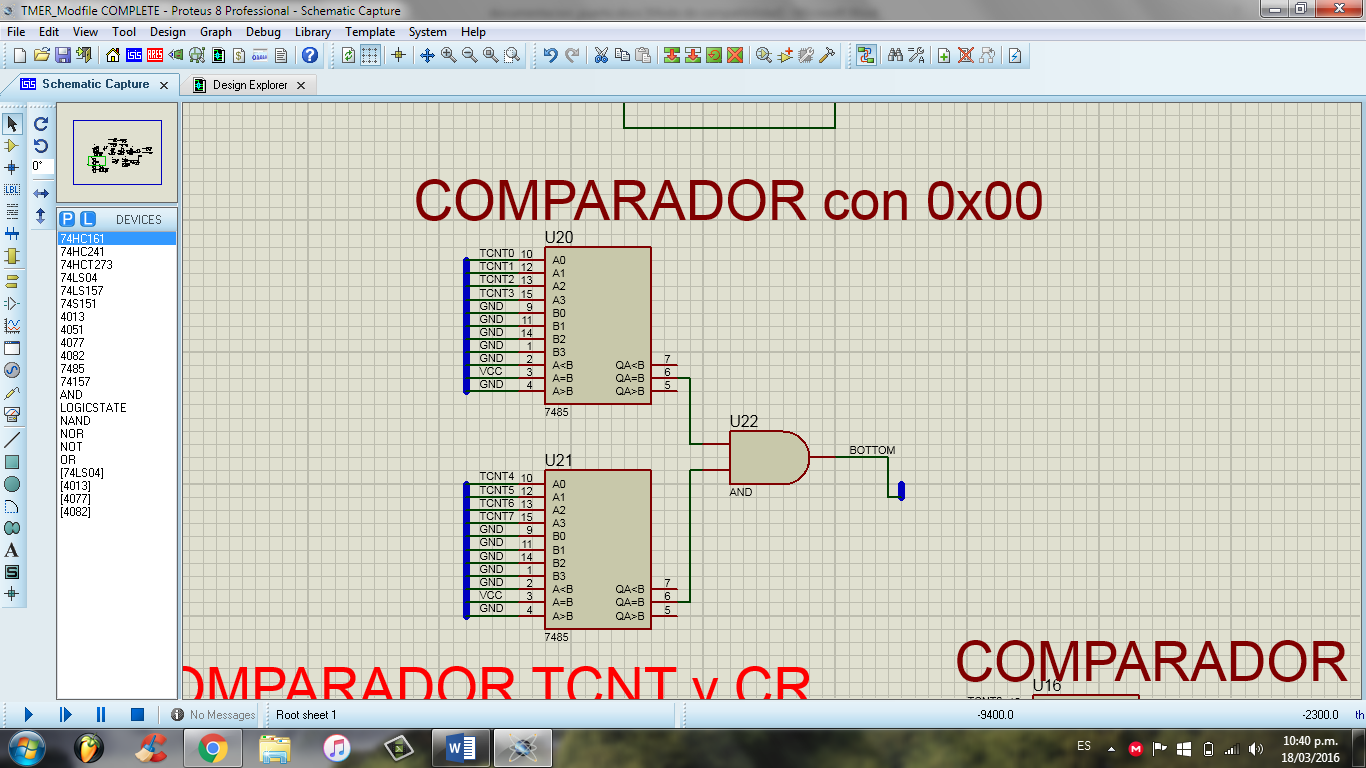


Fig 20. Circuito comparador del registro TCNT con 0.