qwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnm

|  |
| --- |
| INFORME PARCIAL MICROPROCESADORES MCU AVR  PUERTO DE 8 BIT’S  04/03/2016  LIDER: Briant Moscoso C. cod 1094947405. DESARROLLADOR: Andre Felipe Arias P. cod. 1094945389. DOCUMENTADOR: Sebastian Diaz Velez. Cod. 1094942037. |

**PUERTO**

Para la construcción de nuestro **microprocesador mcu avr** debemos de realizar una serie de pasos, nuestro primer objetivo es la elaboración del puerto en este caso de 8 bit’s.

Para asegurar que el funcionamiento de este puerto sea adecuado, tuvimos que involucrarnos con la construcción de un solo pin en un ambiente simulado utilizando, la herramienta **isis Proteus.**

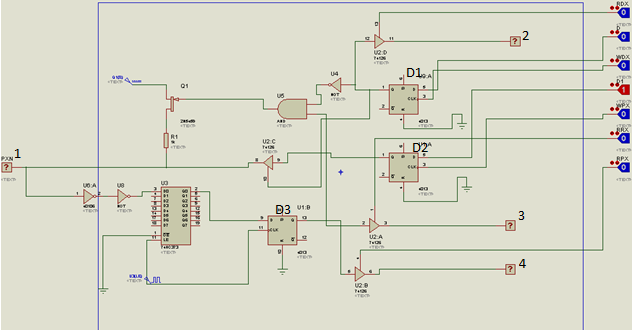


Figura 1. Infraestructura de un pin

En esta infraestructura utilizamos diferentes componentes como lo son:

**-Flip flop tipo D (D0)** = Este flip flop lo utilizamos para configurar este pin como salida (1) o como entrada (0).

**-Flip flop tipo D (D1)** = Este flip flop es el que recibe el dato del microprocesador y lo conduce al pin (este caso se da solo cuando el pin está configurado como salida (1)).

**-Flip flop tipo D (D2)** = Este flip flop es el que trae consigo el dato leído del pin y al actuar un reloj sobre este da a conocer el dato que trae al microprocesador.

**-Tri-estado** = permite que un puerto de salida tenga una alta impedancia de estado, además de los típicos niveles lógicos 0 y 1 esto garantiza que cuando haya alta impedancia no haya conducción y evitaos que se presente un corto circuito.

**-Compuerta NOT** = Esta compuerta convierte un 0 en binario en 1 binario y viceversa.

**-Schmitt trigger** = Esta aproxima un voltaje a su estado lógico.

**-Resistencia de Pull-up** = esta lo que hace es que cuando el pin este configurado como entrada el pin por defecto tenga un valor en este caso de 5v ósea un 1 lógico.

**-Compuerta AND** = Esta compuerta es prácticamente para activar la resistencia de pull-up, Teniendo en cuenta la configuración del pin.

**-Latch** = Asegura que haya sincronía con el reloj de la cpu y el flip flop (D2).

Tenemos también unos probadores lógicos esos son el pin (PXN), el que es para indicarle al microprocesador en que está configurado, si como entrada o salida (2), tenemos el de lectura (3) que viene del dato ingresado y el que le permite saber al microprocesador saber qué valor le envía al pin (4).

Para realizar la prueba de este pin utilizamos estados lógicos, utilizando una configuración que nos garantiza el correcto funcionamiento de este pin.

**PRUEBA PIN**

**Compuertas lógicas.**

Para garantizar que nuestro puerto funcione adecuadamente y que cuando configuremos el puerto como salida o entrada, únicamente este habilitado con la opción que seleccionemos y evitar posibles cortos en nuestro pin, para esto utilizamos compuertas lógicas.

A las compuertas lógicas le enviamos direcciones, donde cada dirección nos representa la tarea que deseamos ejecutar.

**WDX**: while DDRx. Este dato es el que nos permite configurar nuestro puerto como entrada o salida, ya que cuando este sea 1 lógico flanquea el flip flop D1 dando paso al dato que nos permite configurar el pin, si no se presenta el flanco no podemos configurar este pin.

**RDx**: Read DDRx. Es un buffer de 3 estados (U2:D) y dice al usuario el tipo de configuración que realizo previamente

**WPx**: write Port. Esta señal es la que configure el flip flop D2, que es el que utilizamos cuando configuramos nuestro puerto como salida. WPx es terminal que nos flanquee nuestro flipflop, y este hace posible que el dato que se escriba sea transmitido a la salida de nuestro pin.

**RRx**: Read Port Register. Esta señal nos controla el buffer de 3 estados U2: B y direccionamos esta señal cuando queremos leer el registro que escribimos previamente.

**RPx**: Read Port x Pin: Esta es la señal que debemos direccionar cuando deseamos configurar nuestro puerto como entrada.

Estas direcciones son expresadas en código binario, donde cada dirección únicamente nos representa una sola tarea.

Adicional a esto utilizamos dos datos adicionales RD y WR donde para ninguno de los casos ambos deben ser cero, si queremos configurar nuestro puerto como salida WR debe ser cero y DR 1, si queremos configurar nuestro puerto como entrada RD=0 y WR=1.

**Direcciones:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | A2 | A1 | A0 | WR | RD |
| WDx | 1 | 0 | 0 | 0 | 1 |
| WPX | 1 | 0 | 1 | 0 | 1 |
| RPx | 1 | 1 | 0 | 1 | 0 |
| RRx | 1 | 0 | 1 | 1 | 0 |
| RDX | 1 | 1 | 1 | 0 | 1 |

En la figura 2 se muestra cuando vamos a ejecutar la opción de configurar el puerto y vemos que para esa dirección solo se ejecuta esta tarea y solo tendríamos que esperar el flanco para que el último registro digitado en el flip flop D1 nos permita escribir o leer un determinado dato.

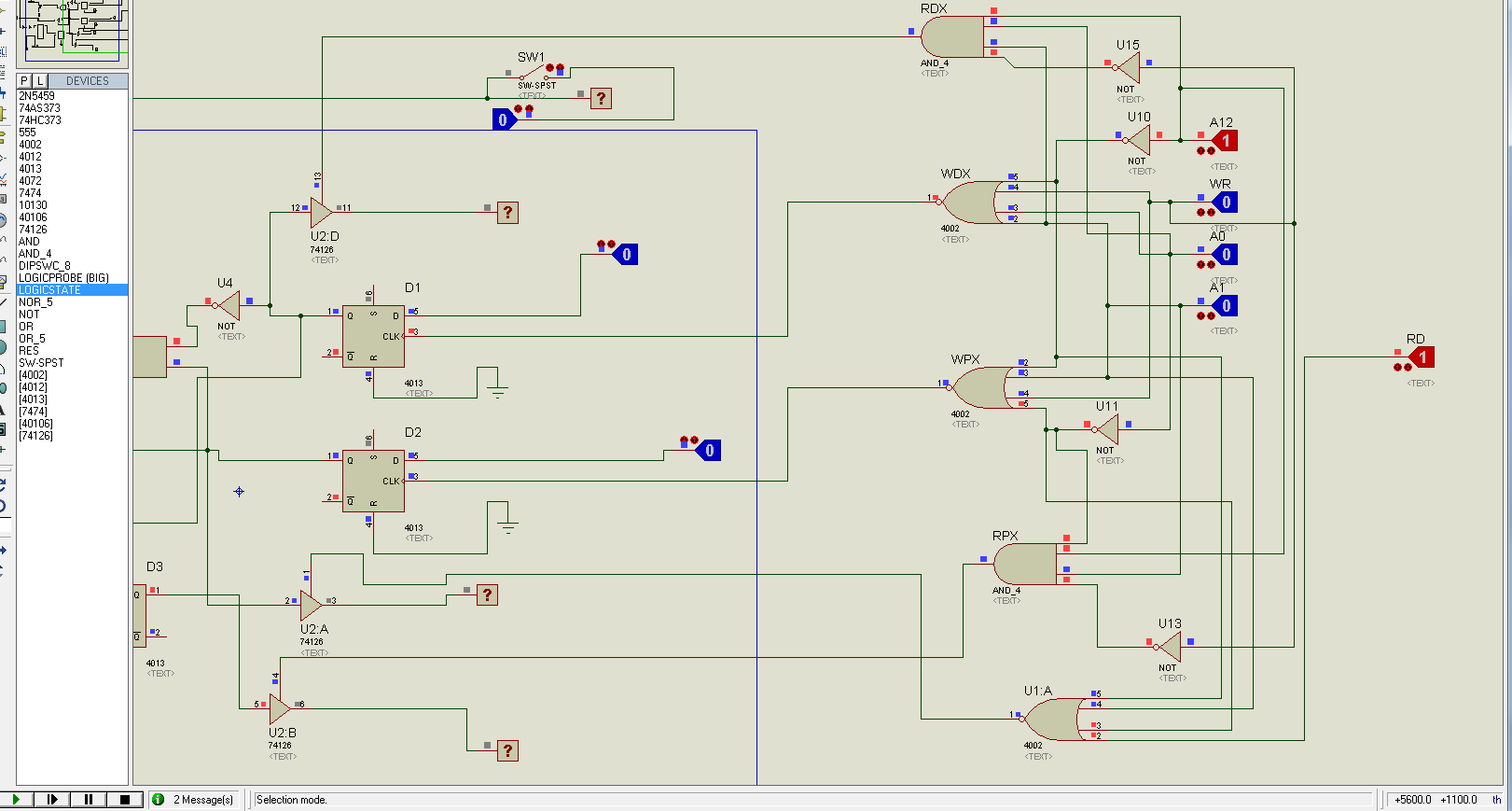


Figura 1.1 Prueba del pin con las compuertas.

Más adelante retomaremos este tema y veremos que lagunas entradas y unas salidas son unidas en un nodo bidireccional estas entradas son la entrada de los flip flop D1, D2, y la salida de los triestados. Estos se unen para garantizar que un solo dato este en por el sistema y no haya confusión.

**COMPONENTE**

Para crear el componente que queremos es necesario introducir este pin en un subcomponente por así decirlo.

**SUBCOMPONENTE:**

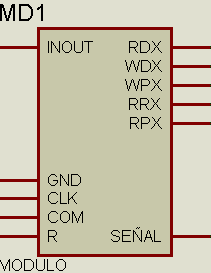


Figura 2 Subcomponente

En este componente tenemos toda la circuitería de la figura1. Donde tenemos dos bidireccionales que es el **INOUT** Y **SEÑAL**, **INOUT** es la entrada y salida del pin y **SEÑAL** es el puerto al microprocesador.

También tenemos lo común en cada componente que es:

**-GND:** la tierra que algunos componentes como la AND y los flip flop necesitan para poder trabajar correctamente.

**-COM:** Común, es el voltaje que activara la resistencia de pull-up dando un 1 lógico a la entrada del pin INOUT.

**-CLK:** Clock o reloj, es la frecuencia a la que trabajar el sistema, esta debe de ser sincronizada con el microprocesador, en este caso la trabajaremos 16Mhz.

**-R:** Reset, este cuando se activa dejaría todos los flip flop y el sistema en ceros 0.

Tenemos **RDX**, **WDX**, **WPX**, **RRX**, **RPX** que como anteriormente dijimos es un conjunto de bit’s que nos darán una dirección y con esto configuraremos el componente.

Ahora para darle sentido a toda esta circuitería y componentes, el subcomponente se convierte en una parte del componente al que queremos llegar, ahora unimos estos subcomponentes que manejan solo un dato y los convertimos en un componente que maneje 8 datos (8 bit’s) de esta forma:

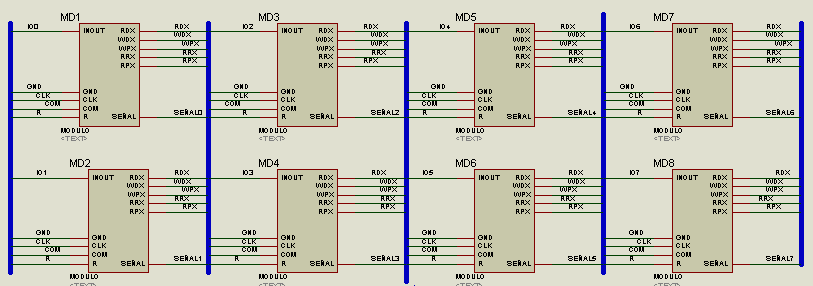
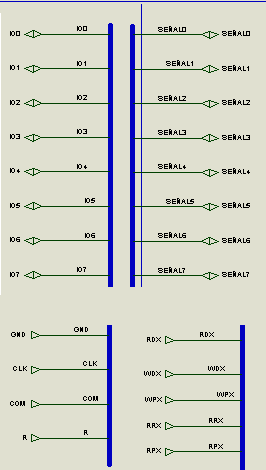


Figura 3 Subcomponentes unidos.

En esta parte de desarrollo del puerto, unimos cada subcomponente mediante las señales de entrada, y dejando las señales bidireccionales individualizadas dándoles su nombre en decimal de la siguiente manera INOUT (IO0,IO1,…,IO7) y SEÑAL (SEÑAL0,SEÑAL1,…,SEÑAL7). En este instante tenemos un puerto de 8 bit’s solo nos falta las entradas y las salidas.



Tenemos dentro del componente las señales y las salidas como ya habíamos visto.

Cabe aclarar que las entradas GND, CLK, COM, R, RDX, WDX, WPX, RRX y RPX son iguales para los subcomponentes MD.

Figura 4 Entradas y salidas componentes (interno)

Ahora bien, como el microprocesador no envía las entradas RDX, WDX, WPX, RRX y RPX solo manda direcciones, es indispensable dentro de este mismo componente instalar un sistema que nos permita saber que tarea ejecutar y ejecutarla correctamente, por ende implantamos en este las compuertas lógicas antes mencionadas, para que el componente no tenga problema en ejecutar las acciones que el microprocesador ordena.

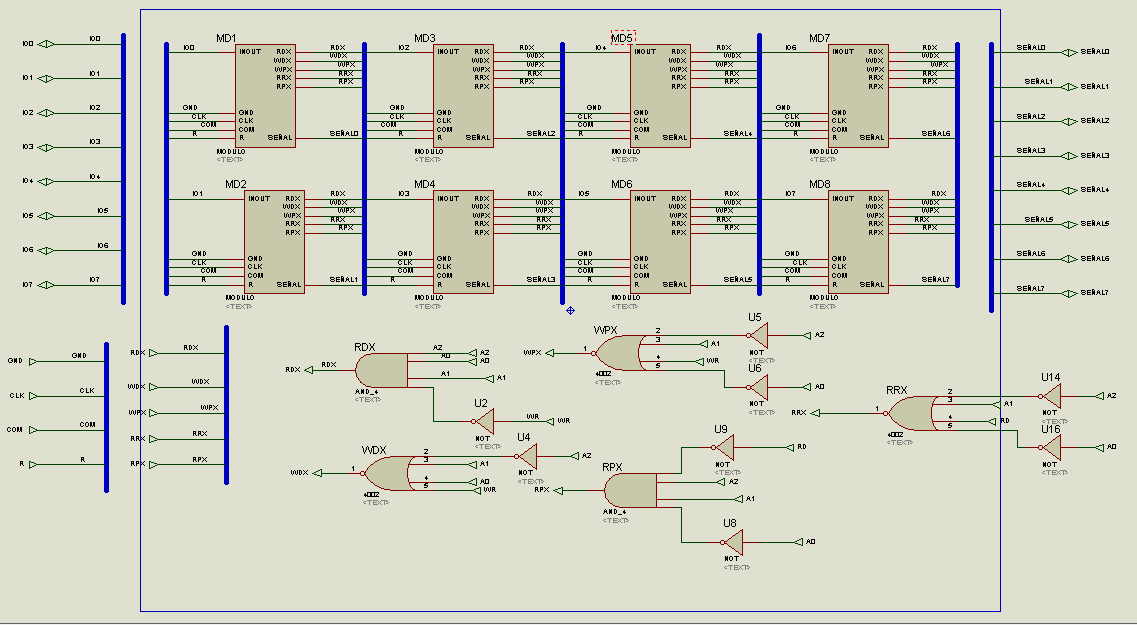


Figura 5 Componente vista interna

En la figura 5 tenemos ya la circuitería del componente internamente, dando paso a poder encapsular todo esto en el componente que realmente nos interesaba. A continuación veremos el prototipo final de este componente.

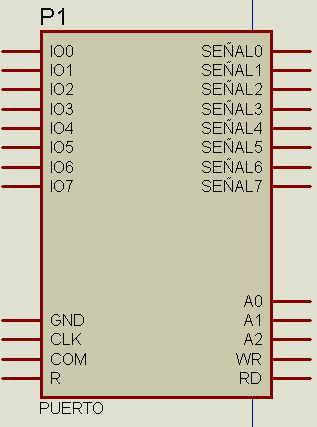


Figura 6 Componente

Este es nuestro producto final donde se encuentra todo el ponderado de la circuitería hecha anteriormente. Este componente es capaz de entender las ordenes del microprocesador, con las entradas A0, A1, A2, WR, RD, con estos datos el microprocesador se configura como lectura o escritura, tenemos los IO0, IO1, ..., IO7, estos son bidireccionales y son los que van al exterior del puerto, también sirven para leer los datos que le queremos dar a conocer al microprocesador. Las SEÑALES son las provenientes del microprocesador y son el camino de los IO’s, estas van conectados al microprocesador.