qwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnm

|  |
| --- |
| INFORME PARCIAL MICROPROCESADORES MCU AVR  ATMEGA 8515 & MEMORIA RAM  11/03/2016  LIDER: Briant Moscoso C. cod 1094947405. DESARROLLADOR: Andres Felipe Arias P. cod. 1094945389. DOCUMENTADOR: Sebastian Diaz Velez. Cod. 1094942037. |

**ATMEGA 8515 & MEMORIA RAM**

En el desarrollo del microprocesador tuvimos que implementar diferentes componentes de acuerdo a las necesidades de este, en este caso tenemos la novedad que a este sistema de puerto y microprocesador debemos de adjuntar una memoria RAM la cual debe de guardar algunos datos en las direcciones que escojamos o el usuario escoja. Para lograr esto debemos sincronizar el microprocesador en este caso el Atmega 8515 con los puertos y esta memoria RAM, esta configuración la lograremos usando una muy buena herramienta llamada CODEBLOCK’s, en esta configuramos el microprocesador y le daremos las ordenes necesarias para un caso en particular, después de una compilación en esta herramienta y adjuntando el .HEX al Atmega tendremos la configuración implantada en este, dando paso a las respectivas pruebas y desarrollo de este proyecto, podremos configurar este microprocesador las veces que sean necesarias, ya que con solo compilar este programa quedara configurado el microprocesador.

A continuación tenemos una vista en concreto del circuito ya dicho:

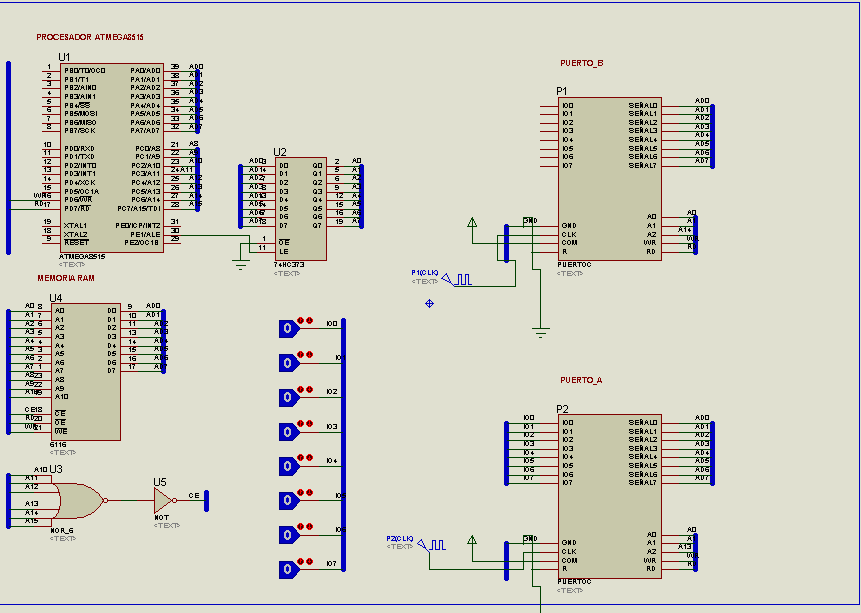


Figura 1. Microprocesador.

Podemos observar que tenemos 2 puertos los cuales depende de la configuración pueden quedar uno como entrada y el otro como salida, en este caso el **puerto A** es al que le ingresamos los datos y el microprocesador los toma (los lee) y depende de lo que queramos el micro los puede guardar en un respectivo lugar de la RAM y después si es necesario obtener estos datos para llevar a cabo una determinada acción, una de las acciones más comunes y que hemos visto hasta el momento es mostrarlo en el **puerto B** en este caso como salida del microprocesador.

**MEMORIA RAM**

**DEFINICION**: La memoria RAM es conocida como **memoria volátil** lo cual quiere decir que **los datos no se guardan de manera permanente**, es por ello, que cuando deja de existir una fuente de energía en el dispositivo la información se pierde. Asimismo, la memoria RAM puede ser reescrita y leída constantemente. Podemos decir que es una memoria tipo buffer la cual solo esta activa cuando el dispositivo esta encendido, y la información que se guarda en esta desaparece al instante que el dispositivo deje de funcionar.

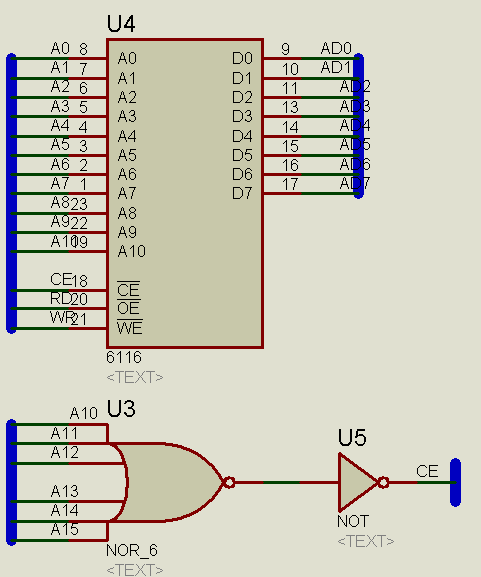


Figura 2. Memoria RAM.

En este caso utilizamos una memoria RAM con la referencia 6116 en ISIS Proteus, cabe aclarar que esta memoria no siempre debe de estar guardando datos por ende la línea CE, es la que autoriza la entrada y salida de datos de esta RAM, si esta está en 0 el microprocesador no podrá acceder a esta y de lo contrario sí.

Esta memoria necesita las direcciones en las cuales va indicado el respectivo lugar que se van a guardar los datos, estas direcciones dadas por el microprocesador van conectadas a las líneas A0,A1, …, A10. Ahora la incógnita que surge es de donde salen A0, A1, …, A7. Estas direcciones salen del latch ya que estas direcciones si las genera el microprocesador pero en unos pines compartidos con los datos que salen y entran al microprocesador, por ende debemos de separar los datos de las direcciones, A continuación observaremos el latch.

**LATCH**

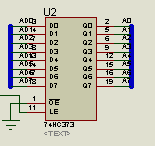


Figura 3. Latch.

Podemos observar que al lado izquierdo entran los datos provenientes del microprocesador y al lado derecho ya salen las direcciones que van a la memoria RAM, esto se debe a que el microprocesador da la orden al latch por medio de enable (LE), que de paso a estas direcciones e inmediatamente cuando el LE se active pasan hacer las direcciones y no datos.

Ya teniendo esto pasaremos a la parte de la programación del microprocesador, en esta parte es donde definimos el comportamiento de cada componente de este circuito.

Usando la herramienta CODEBLOCK’s

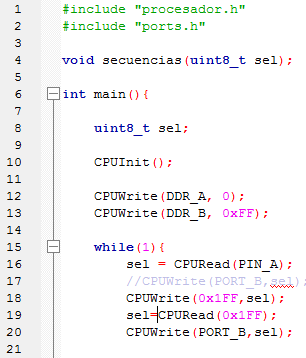


Figura 4. Programación CODEBLOCK’s

En esta parte del desarrollo del proyecto tuvimos que programar en la ya nombrada herramienta, en este pantallazo tenemos que configuramos que el **puerto A** es la entrada que va a leer el microprocesador “CPUWrite (DDR\_A, 0);” y el **puerto B** es la salida del microprocesador “CPUWrite(DDR\_B, 0xFF);”.

En el proceso tenemos que definir la variable en la que vamos a guardar los datos obtenidos del **puerto A** y esta la nombramos en este caso (uint8\_t sel;) ya que son 8 bits los que vamos a guardar, como ya habíamos dicho esta variable guardara los bits que ingresamos en el **puerto A**. En la siguiente línea de código tenemos “CPUWrite(0x1FF,sel);” Esta línea lo que hace es escribir en el espacio 0x1FF de la memoria RAM el valor que tiene la variable “sel”, siguiendo con el orden la otra línea “sel=CPURead(0x1FF);” sobre-escribe el valor de la variable “sel” que debería ser el mismo valor de la misma, y en esta línea es cuando comprobamos si la memoria RAM está funcionando ya que esta dirección deberá tener el mismo valor de “sel” de lo contrario esta variable quedaría con un valor desconocido, después para ver resultados la siguiente línea “CPUWrite(PORT\_B,sel);” da la orden de mostrar en el puerto B el valor que está guardado en “sel” dando a revelar si la memoria esta guardando el valor correcto. Por lo tanto el mismo valor ingresado en el **puerto A** lo veremos en el **puerto B.**