TEMPORALIZADOR

8 BITS

Arcos Cerón Jorge Alberto.

Resumiendo nuestro trabajo hasta el momento, recordemos que iniciamos con la creación de un Puerto de 8 bits que posteriormente duplicamos y configuramos con la ayuda de un procesador ATMEGA8515 y con código C compilado en AVR adecuado de tal manera, que un Puerto sea una entrada (PuertoA) y el otro una salida (PuertoB).

Esto con el fin de incluir como siguiente pasó en nuestro sistema un registro de Memoria RAM e implementar otro código para que los datos ingresados por el Puerto A de entrada hagan la transición por dicha memoria antes de ser Leídos por el Puerto B, el resultado final de estos pasos lo observamos en la Figura1.

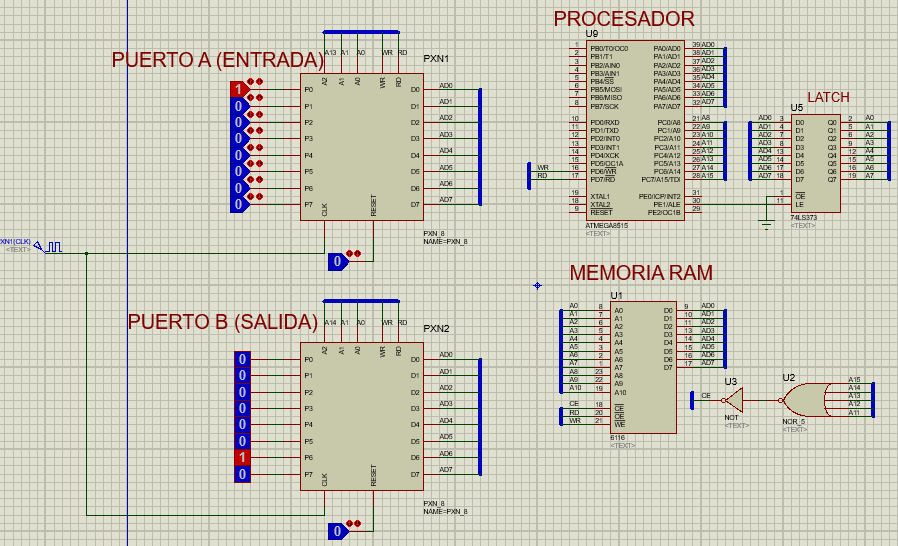


Figura 1, Puertos - Procesador – RAM

Partiendo de nuestro trabajo actual, que consta principalmente de Un Procesador, dos puertos E/S de 8 bits y una Memoria RAM, Existe la necesidad de Incluir un Temporalizador (TIMER) con el fin de completar nuestro sistema final que es el Micro-Controlador.

El TIMER cumplirá una gran función dentro de nuestro sistema final, ya que prácticamente todas las tareas o aplicaciones para un Microcontrolador tienen implícita la variable tiempo, inclusive llevar la fecha y hora actual; el Timer es quien le permitirá realizar dichas tareas con gran precisión.

El diagrama de Nuestro TIMER es el siguiente:

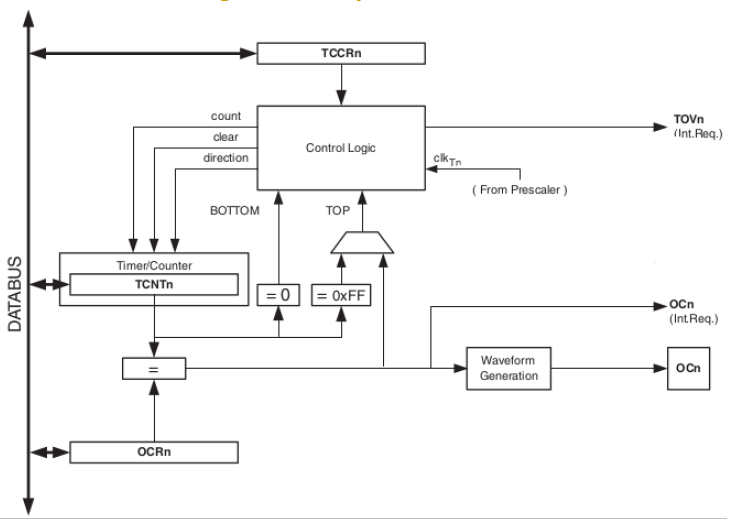


Figura 2, Diagrama Timer

Consta básicamente de 6 Regiones TCCRn, TCNTn, OCRn, Un Control Lógico, Un Pre-Escalador y un Generador de Onda.

**TCCRn:** Es el registro de control del Temporalizador, pues permite modificar el funcionamiento de este, ya que también puede ser usado como un Contador.

Su implementación en la herramienta Proteus es la mostrada en la Figura 3.

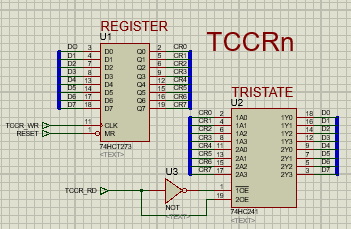


Figura 3, Registro TCCRn

Los 8 bits Ingresados inicialmente al registro TCCRn se distribuyen tal y como se observa en la Figura 2 pero que en nuestra implementación de proteus corresponden al bus de datos CR[0..7].

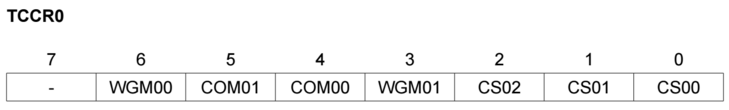
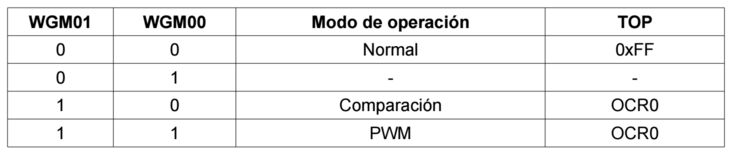


Figura 2, Distribucion de los bits en TCCRn

De acuerdo a esta información el Timer adoptara diferentes Configuraciones

Para los bits 3 y 6 denominados WGM01 y WGM00 o en nuestro caso CR3 y CR6

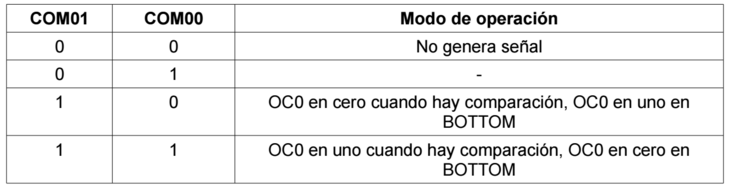


Obtendremos un contador de 0 a 256 que es el número mayor posible con 8 bits para nuestro primer caso ( 0 , 0 )

En el caso de escribir en ellos un (1 , 0) respectivamente nos resulta un contador de comparación que cuenta hasta el valor existente en el registro OCR0. Que describiremos más adelante.

Otro posible caso y el de nuestro interés es cuando escribimos en un 1 en los dos bits, así obtenemos una señal PWM que corresponde a una señal cuadrada modulada en el tiempo, que cambia su estado para cada Valor de TOP en este caso correspondo al valor de OCR0.

Los bits 4 y 5 denominados COM01 y COM00 conllevan la siguiente configuración para el Timer:



Donde BOTTOM es el valor mínimo que puede tomar OC0.

Los Bits 0,1 y 2 denotados CS00, CS01 y CS02 configuran la cantidad de Preescalado de la Señal de salida PWM.

Para las combinaciones posibles tenemos como Resultado lo descrito en la tabla de la Figura 4.

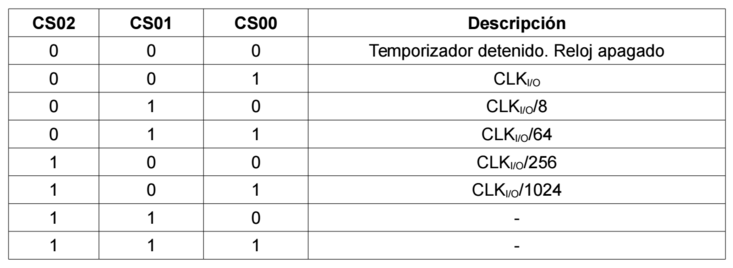
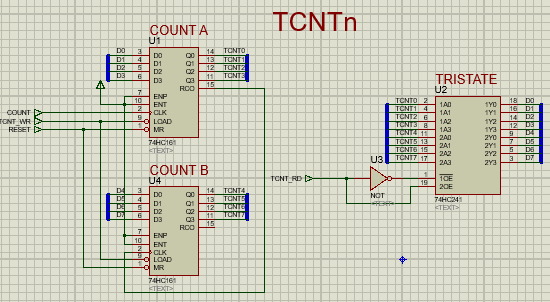


Figura 4, Tabla de Preescalado

Donde CLKio es la señal de Reloj Ingresada al sistema. Dicha señal es la base para el Preescalado.

**TCNTn:** Este es un registro de Datos cuya función es almacenar la cuenta actual del Temporalizador.

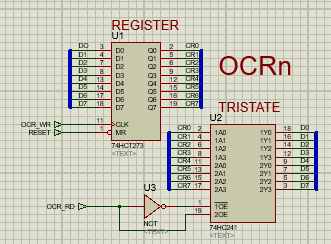
Su diseño en Proteus es el siguiente:



En el observamos que Ingresan los datos D[0..7] con la cuenta actual de nuestro Temporalizador, si existe la orden en TCNT\_WR, estos son almacenados en las variables TCNT[0..7], esperando la activación del pin TCNT\_RD que simboliza su lectura y remplazo en los antiguos datos D[0..7].

**OCRn:** Este también es un registro de Datos pero como lo mencionamos anteriormente, con la función de almacenar un Dato de Comparación, el cual se iguala con la cuenta Actual del Temporalizador con el fin de generar interrupciones y modificar el ancho de pulso de la señal PWM.

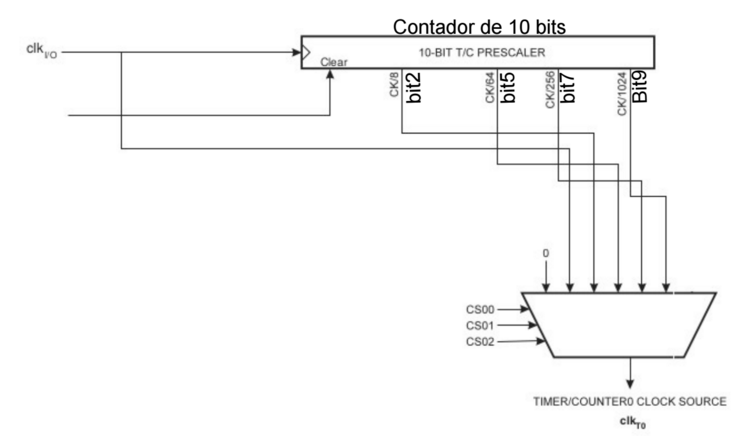
Su Implementación es:

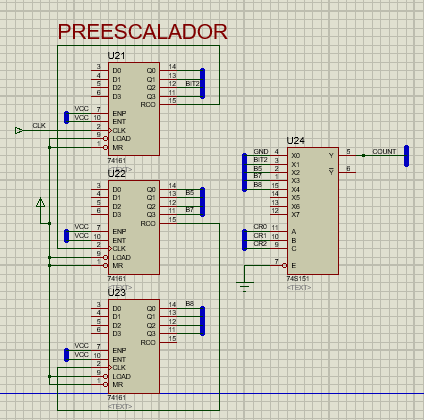


Notamos un comportamiento de almacenamiento similar al del registro TCCRn, con la diferencia en el uso de los datos almacenados.

**PRE-ESCALADOR**

Su diseño e implementación en proteus son los siguientes

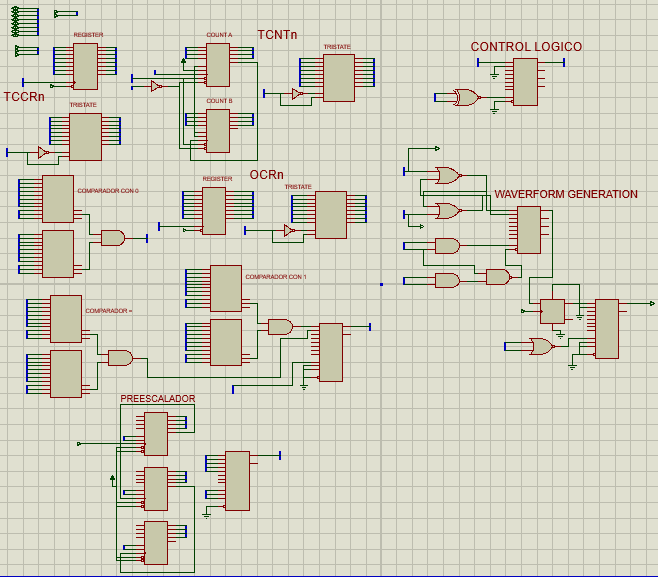




El cual como mencionamos y observamos en la Figura 4, su pre escalado depende de los bits 0,1 y 2 ingresados en el registro TCCRn y su configuración esta descrita en la misma.

Implementada Cada Region procedemos a formar el cuerpo del Timer según las especificaciones del Diagrama visto en la Figura 2.

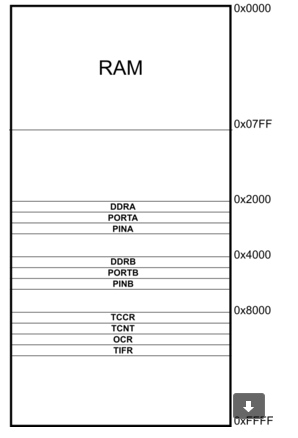
Siguiendo al pie sus Instrucciones y Conexiones obtenemos el siguiente Circuito



Teniendo ya el cuerpo de nuestro Timer, debemos incluir en el las compuertas que nos permitirán acceder a cada región para escribir o leer de ellas.

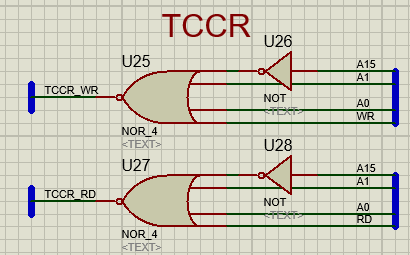
Para el diseño de estas compuertas debemos tener presentas las direcciones asignadas para cada Región a acceder.

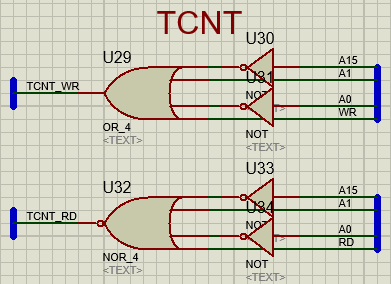
Para nuestro diseño hicimos la siguiente distribución de memoria

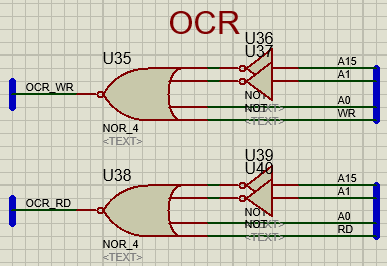


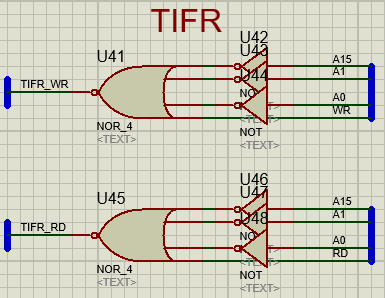
En el que se le asignó al Timer las direcciones 0x8000, 0x8001, 0x8002, 0x8003, 0x8004 para las regiones TCCR, TCNT, OCR y TIF respectivamente.

Con base a ello diseñamos a partir de compuertas lógicas NOR, NOT y OR de manera similar que para el diseño de los Puertos, las siguientes compuertas para acceder a cada una de estas regiones.







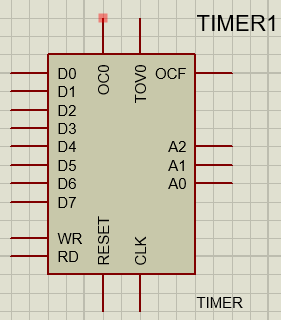


Procedemos a incluirlas en el cuerpo de Nuestro Timer y así completarlo.

Y como proceso final de diseño, creamos el componente que contenga el cuerpo de todo nuestro Timer, obteniendo como resultado, una pastillas con 8 Pines de Datos Bidireccionales D[0..7], 3 pines de entrada para las Direcciones A[0..2], un Pin de Reset, uno para la Entrada de Reloj, nuestro Pin de salida de la señal PMW denominado OC0 y dos Pines para señales Bandera OCF y TOV0 las cuales señalan los cambios de estado de cero a uno y de uno a cero de nuestra señal PWM.

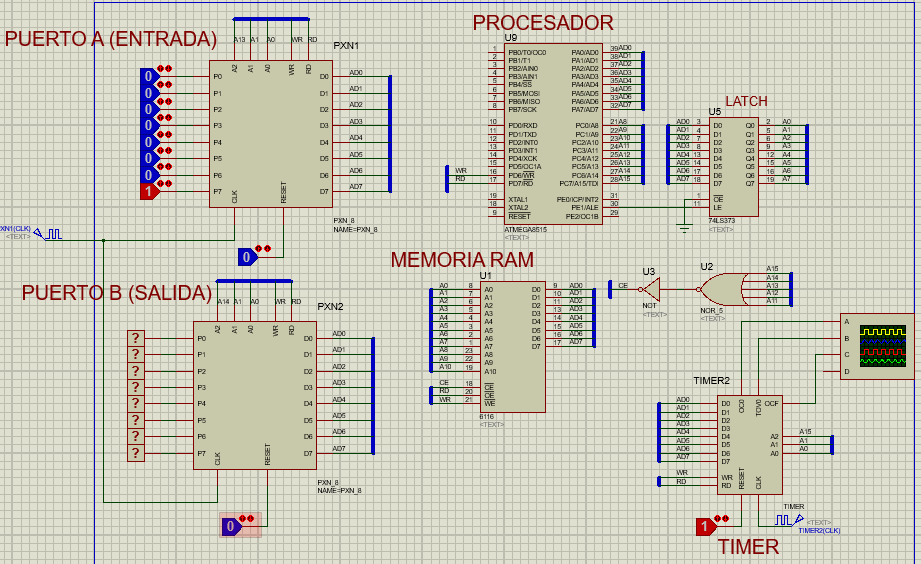
Por ultimo dos Pines también de entrada para los mandos de Escribir (WR) y leer (RD).

Este componente se puede observar a continuación:



Finalmente incorporamos nuestro componente TIMER al circuito que tenemos actualmente expuesto en la Figura 1.

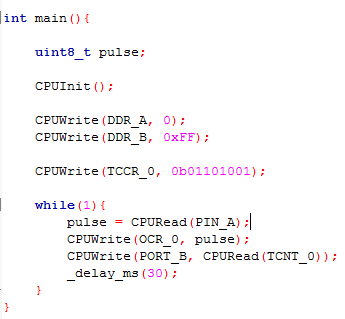
Al hacer este procedimiento y realizar las respectivas conexiones entre ellos obtenemos el siguiente sistema.



Al que le hemos colocado un Osciloscopio en la salida OC0, TOV0 y OCF para visualizar nuestra señal PWM y sus respectivas Banderas de cambio de estado.

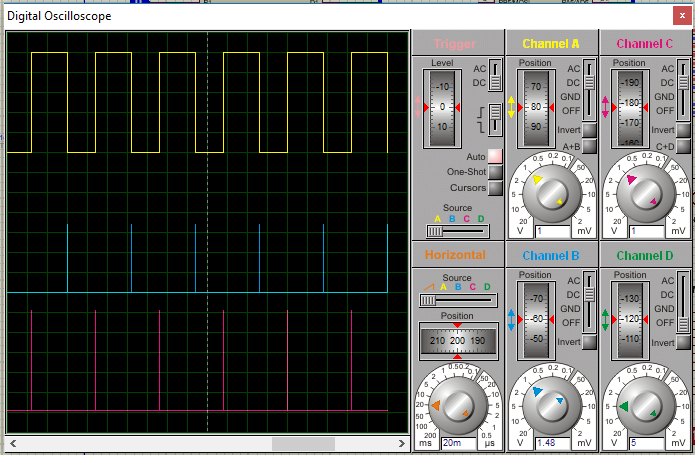
Finalmente y al diseñar un código adecuado en C y compilarlo en AVR para nuestro procesador, en el cual le ingresamos al registro TCCN el vector [01101001] al OCR0 un pulso y también hemos escrito en nuestro Puerto B de salida el conteo que se produce en el registro TCNT0.

Esto es básicamente lo que compone nuestro código de prueba para el Timer



Así y a partir de este código nuestro Microcontrolador queda configurado para resultar una señal PWM con un Preescalamietno del mismo Tiempo de la señal de entrada de Reloj que en nuestro caso tiene una frecuencia de 16k, esto sucede porque le asignamos al registro TCCR0 los valores 0 0 1 para sus 3 últimos Bits (CS02, CS01 y CS00) lo cual determina un Preescalamiento de CLKio, esto se evidencia en la tabla de la Figura 4 expuesta anteriormente.

Si Iniciamos el funcionamiento de nuestro sistema Microcontrolador Obtenemos el resultado esperado una señal PWM, con pre escalamiento de 16k con sus respectivas banderas de cambio de estado.



La señal amarilla es nuestra señal PWM modulada, la señal azul es nuestro TOP que me indica cuando la señal cambia de su estado de mayor valor a su estado mínimo, la señal roja muestra el comportamiento de BOTTOM el cual tiene el comportamiento inverso a TOP, me indica cuando la señal PWM cambia del estado de valor Minimo.