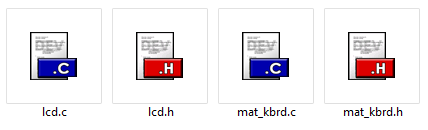
**CONFIGURACION LCD Y TECLADO MATRICIAL EN ATMEL STUDIO.**



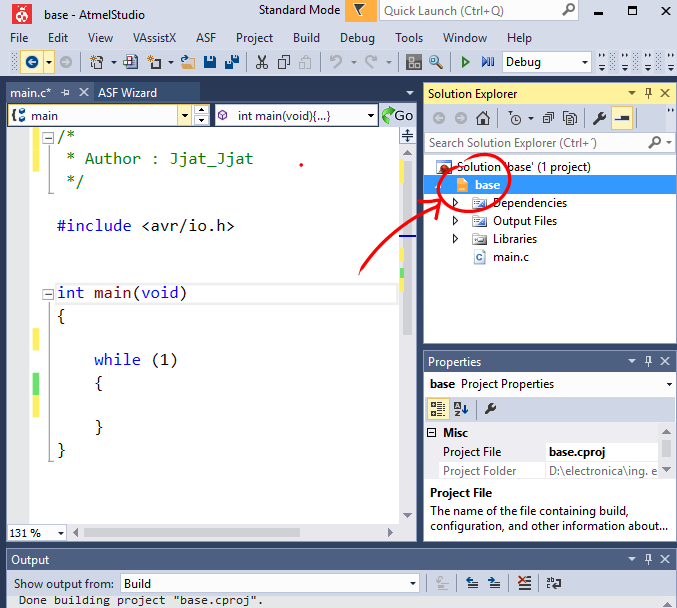
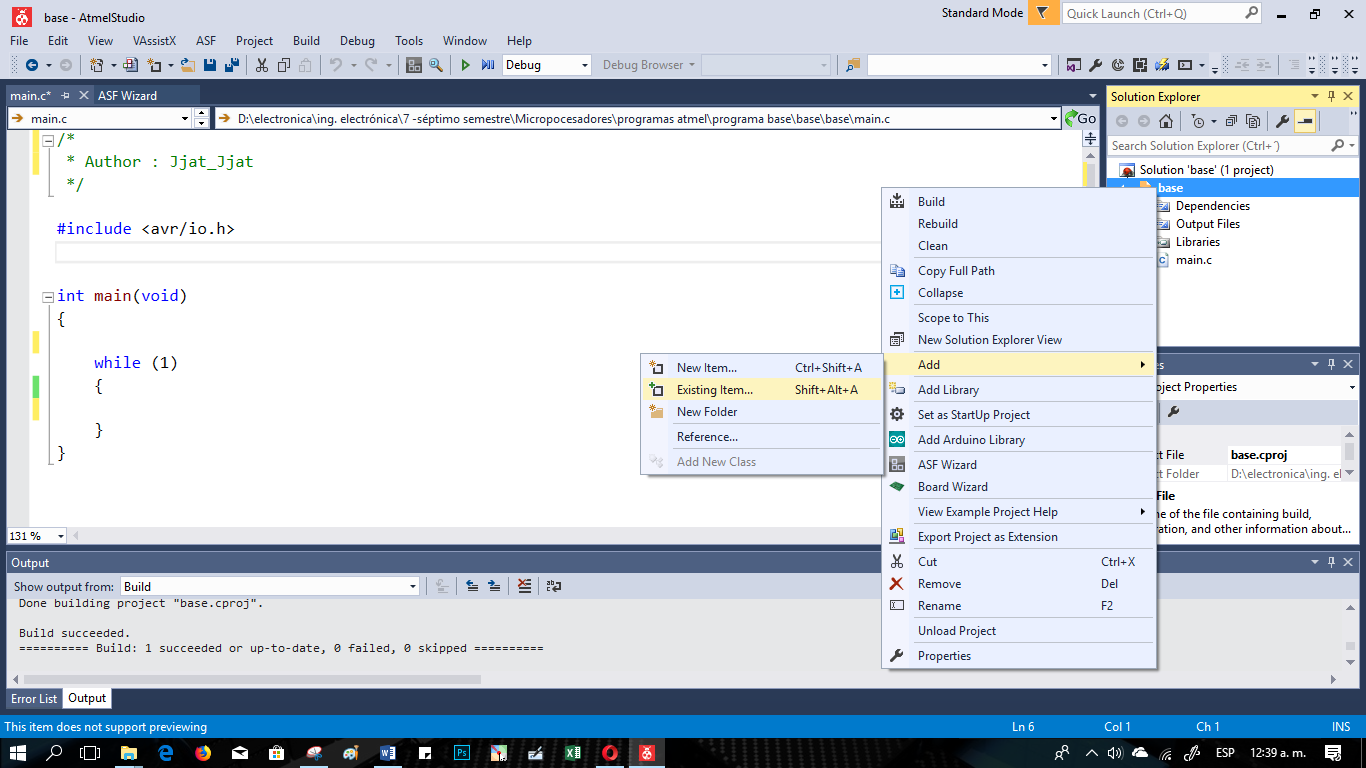
**Nombre: Jaimen Aza-1526982**

**Librerías:**

Para empezar es importante tener primero que todo las siguientes librerías:

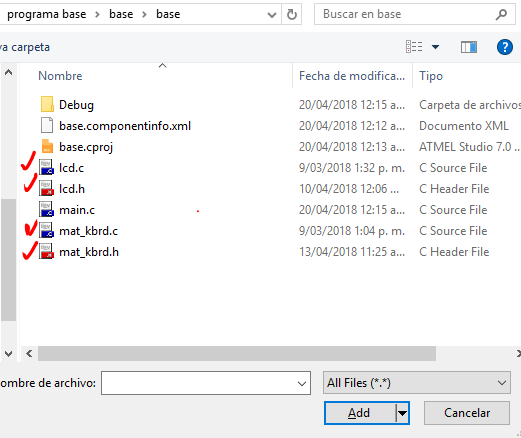


estas librerías deben estar guardas en la misma carpeta donde se encuentra el main.c de nuestro proyecto:

Luego de esto se procede a agregar las librerías en atmel studio, para ello se crea un nuevo proyecto, y después le damos clic derecho sobre el icono amarillo, el nombre corresponde al nombre que el usuario haya asignado al proyecto, después damos clic en add y existing ítem, como se muestra en las siguientes imágenes:

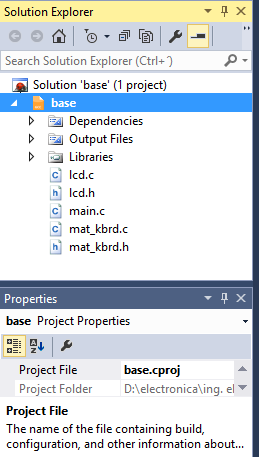


Ahora buscamos las 4 librerias donde ya las habiamos guardado, es decir donde se encuentra el proyecto:

Se selecciona y se agrega las 4 librerias ya mencionadas



Una vez hecho esto se observará como aparen en atmel



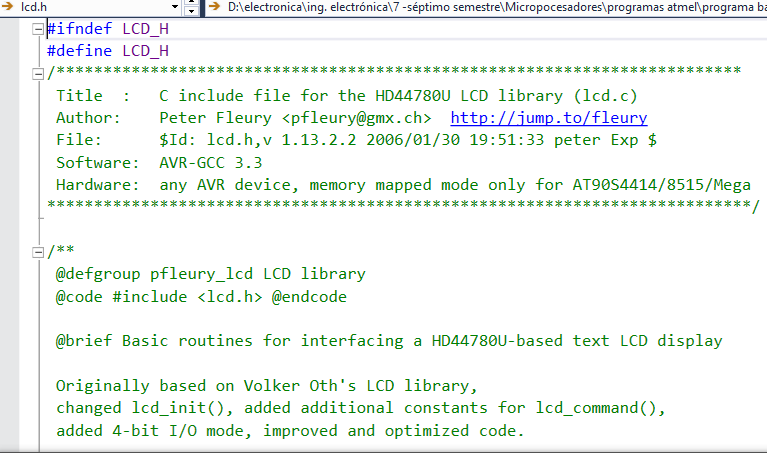
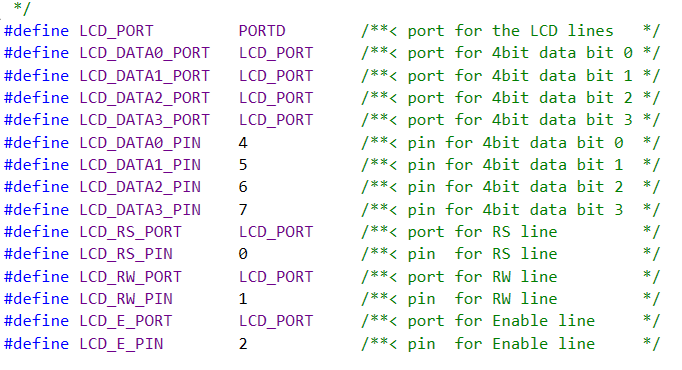


Una vez hecho esto ya tenemos listo nuestro atmel para empezar a trabajar con el lcd y el teclado, para ver sus conexiones y funcionamiento básico se va ser una simulación en proteus, empezaremos mostrando los las configuraciones que hay que hacerle al a las librerías del LCD:

**Configuración Puerto del LCD**

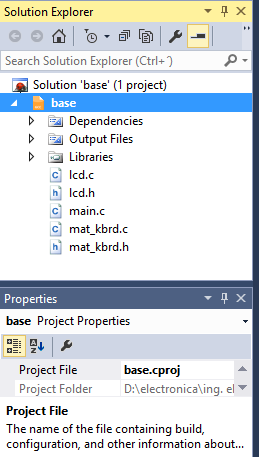
Para ello abrimos el archivo main.h, nos llevará a la siguiente ventana:





Aquí buscamos la parte en la que se le define el puerto a usar para las conexiones, en esto caso está en el puerto D, lo podemos cambiar al puerto que necesitemos, además en las líneas de código hay comentarios que nos da información de que es cada pin. Estos números corresponden al número del pin del puerto que estemos usando.



**Configuración Puerto del teclado**

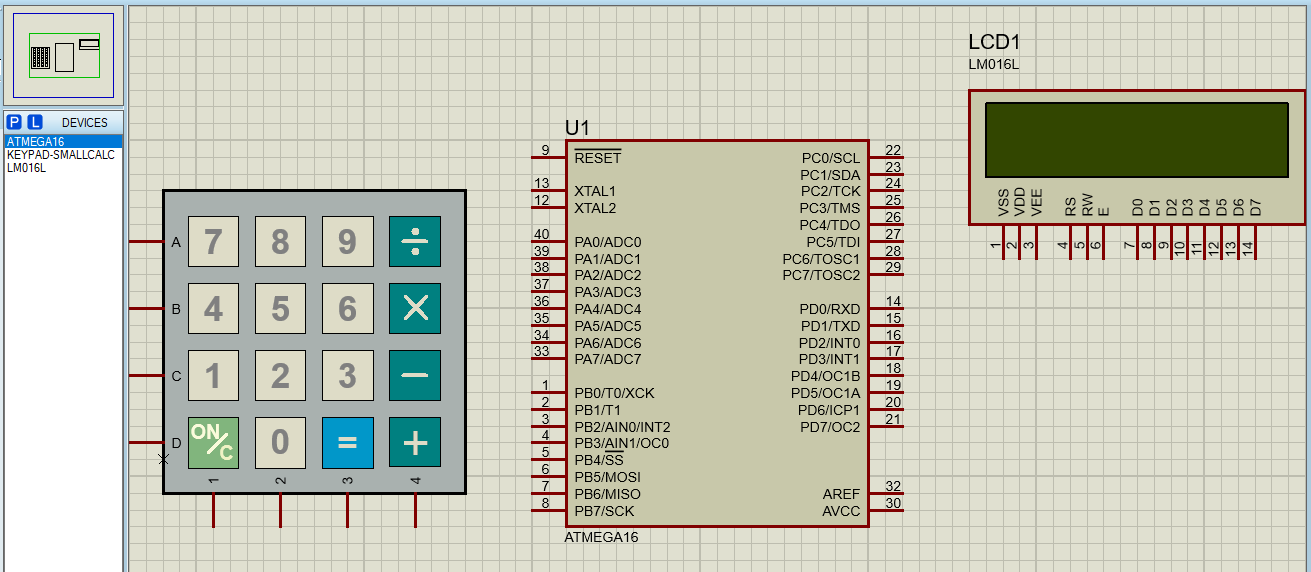
Para configurar el teclado inspeccionamos al archivo mat\_kbrd.h

al abrirlo nos daremos cuenta que hay varias configuraciones tanto como para el puerto   
  
como para cada una de las conexiones, esto será explicado más adelante en conjunto con   
  
la simulacion de proteus.



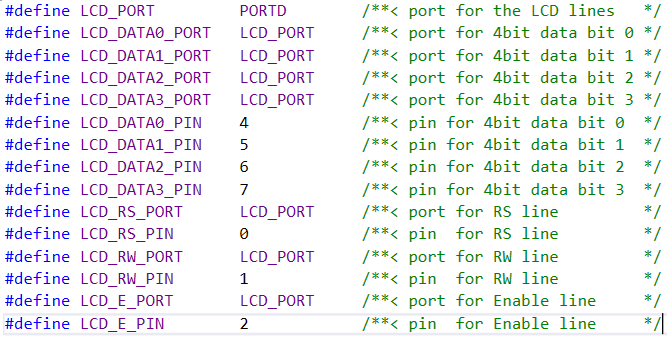
**Simulación en Proteus:**

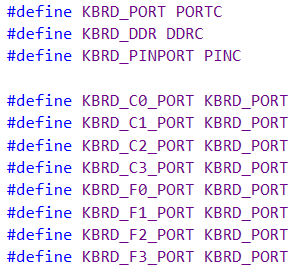
Primero buscamos las tres cosas que se necesita, en este caso se usó el atmega16, el LCD, y el teclado:



Procedemos hacer las respectivas conexiones, esto teniendo en cuenta los puertos que se les asignaron a cada dispositivo en atmel

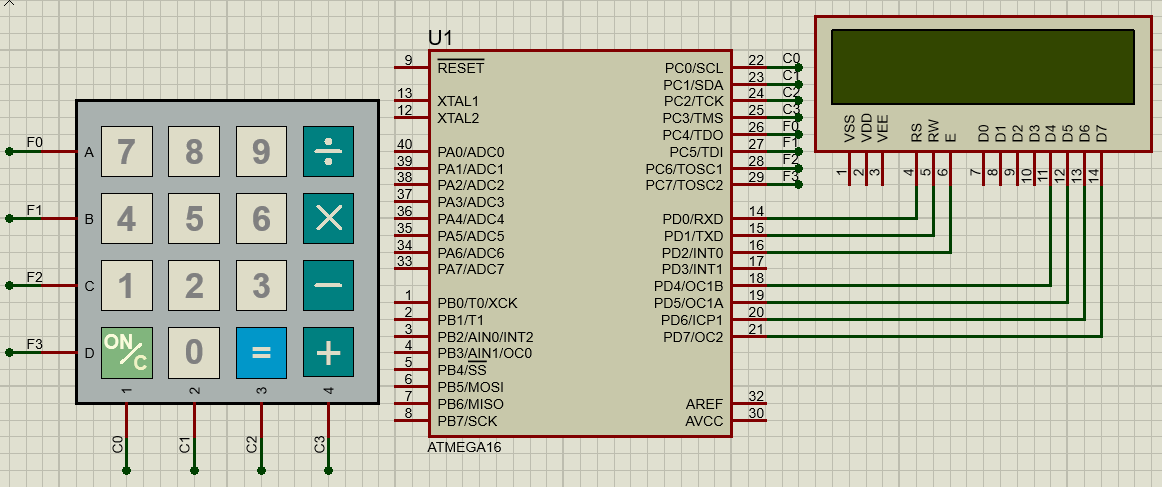
Para ello miramos las conexiones primero para el lCD haciendo clic en lcd.h y hacemos las respectivas conexiones:





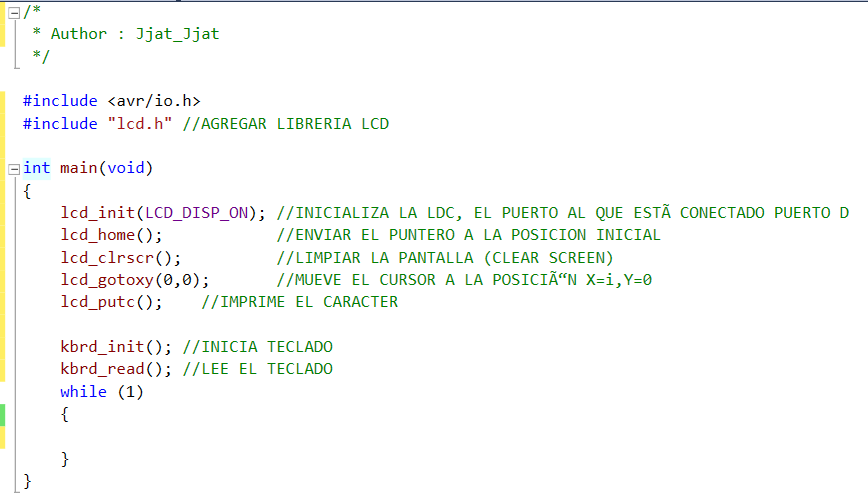
Ahora abrimos los archivos mat\_kbrd.c y mat\_kbrd.h y miramos las conexiones del teclado:

Se puede observar que está asignado el puerto C, también se ve que C0 a C3  
  
corresponden a las 4 columnas y F0 a F3 corresponden a las 4 filas y los pines van  
  
desde el PINC0 hasta el PINC7 respectivamente, por lo tanto las conexiones quedarán  
  
así:



**Programa en Atmel**

Los comandos básicos para el funcionamiento del LCD y teclado son:



**Ejemplo programa en atmel:**

/\*

\* Author : Jjat\_Jjat

\*/

#define *F\_CPU* 16000000UL //freq 16 MHz

#include <avr/io.h>

#include <util/delay.h>

#include "lcd.h"

int main(void)

{

char key;

int i ;

lcd\_init(LCD\_DISP\_ON); //INICIALIZA LA LDC, EL PUERTO AL QUE ESTÃ CONECTADO PUERTO D

kbrd\_init(); //INICIA TECLADO

lcd\_home(); //ENVIAR EL PUNTERO A LA POSICION INICIAL

lcd\_puts(" PRUEBA"); //IMPRIME EL CARACTER

*\_delay\_ms*(50);

lcd\_clrscr(); //LIMPIAR LA PANTALLA (CLEAR SCREEN)

while (1)

{

key = kbrd\_read(); //LEE EL TECLADO

if (key != 0){

lcd\_gotoxy(i,0); //MUEVE EL CURSOR A LA POSICIÃ“N X=i,Y=0

lcd\_putc(key); //IMPRIME EL CARACTER

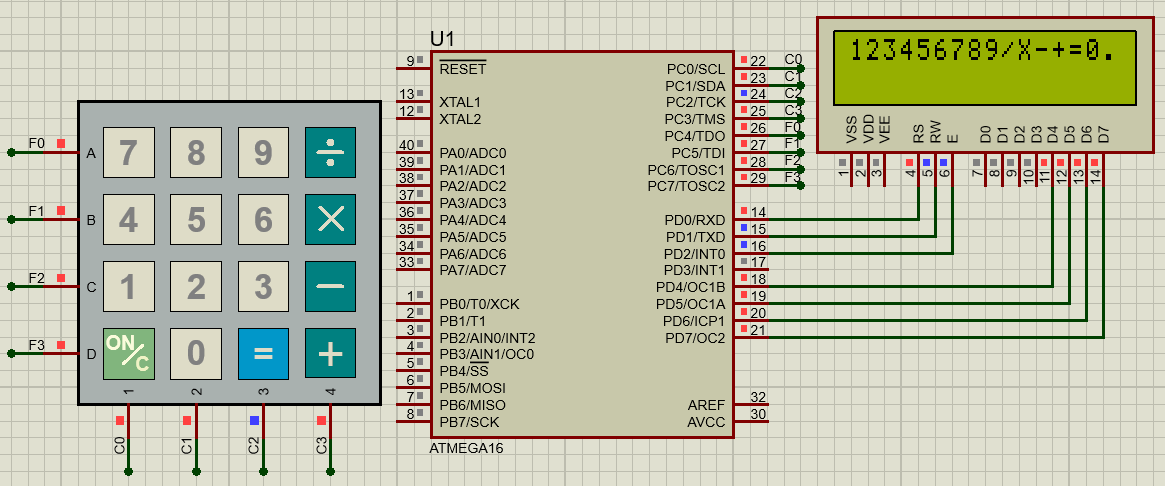
i++;

}

}

}

**Resultado:**



**ADJUNTO LINK VIDEO EN EL CUAL SE EXPLICA LA CONFIGURACION DEL LCD Y TECLADO ANTERIORMENTE VISTA, EN LA DESCRIPCIÓN DEL VIDEO TAMBIÉN SE ENCUENTRAN LAS LIBRERIAS:**

**Librerías:**

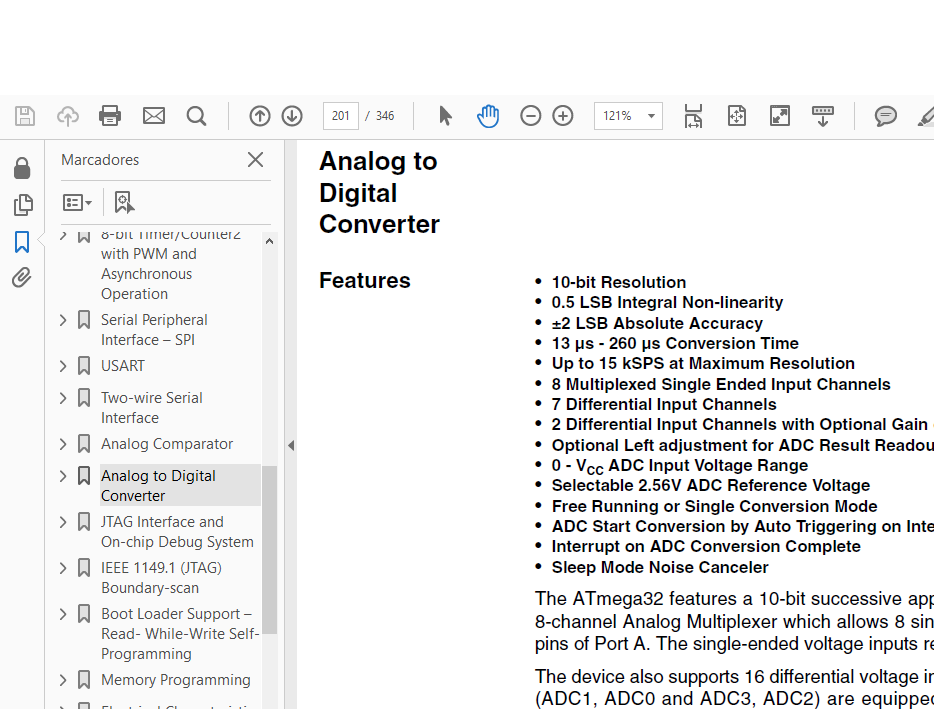
[**https://drive.google.com/open?id=1MtY01k\_nbq7YpKUlMS71EjD3PvGPX\_HA**](https://drive.google.com/open?id=1MtY01k_nbq7YpKUlMS71EjD3PvGPX_HA)

**Link Video:**

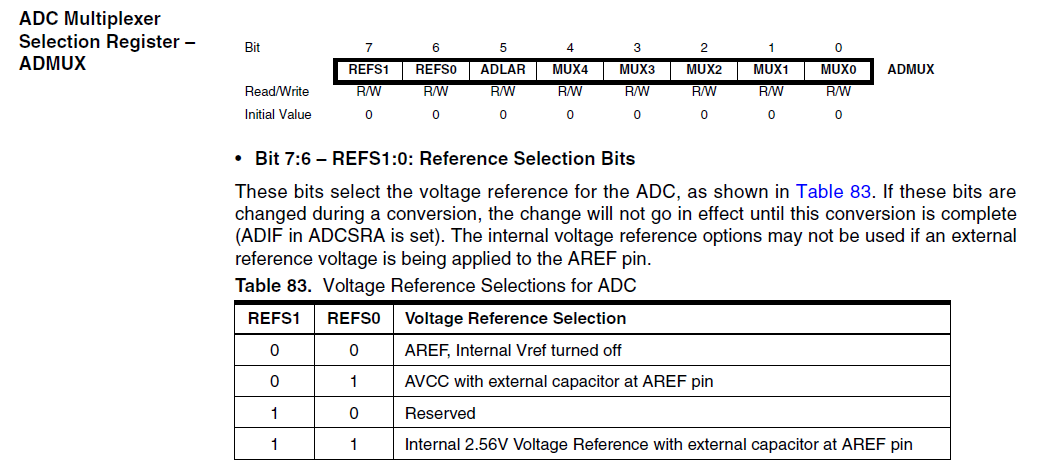
[**https://www.youtube.com/watch?v=KHeUGyIqHjo**](https://www.youtube.com/watch?v=KHeUGyIqHjo)

**CONFIGURACIÓN REGISTROS PARA EL USO DE LOS ADC (ANALOG TO DIGITAL CONVERTER)**

En este caso se van a configurar los ADC del uC atmega 32 es lo mismo para otros atmega, basta con mirar el datasheet de cada atmega e ir a la sección de **Analog to Digital Converter:**



Luego de esto nos dirigimos hacia **ADC Multiplexer Selection Register-ADMUX**:



En esta sección lo primero que hay que hacer es poner un voltaje de referencia que me sirva para hacer la conversión, este voltaje puede ser externo como puede ser interno, para escoger hay que poner en 1 o 0 a los bits 6 y 7 del ADMUX es decir a REF1 y a REF0, como muestra la anterior tabla, en este caso se usará el voltaje de referencia interno que tiene el uC, por lo tanto dejamos ponemos ceros tanto en REF1 como en REF0:

void Init( void )

{

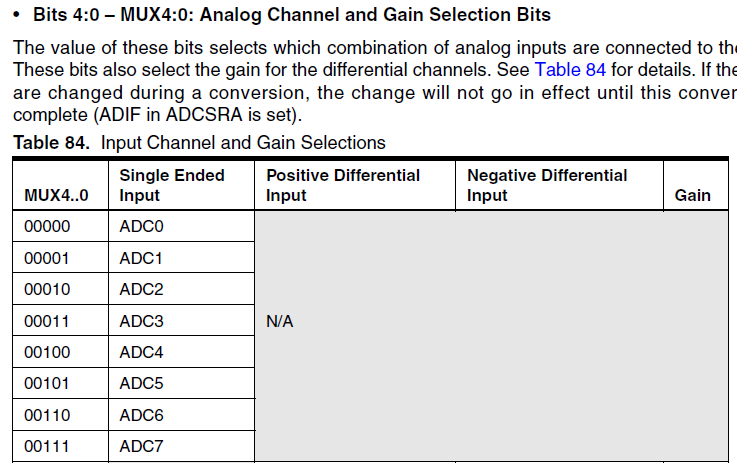
//referencia Vcc

ADMUX &= ~(1 << REFS0);

ADMUX &= ~(1 << REFS1);

}

Luego vamos a escoger que canal queremos usar para recibir nuestra señal análoga:



En el atmega 16,32,64 el puerto A corresponde a todos los ADC, en este caso se va escoger el ADC2 que corresponde Al pin A2 del uC, para ello llevamos al MUX1 a 1:

void Init( void )

{

//referencia Vcc

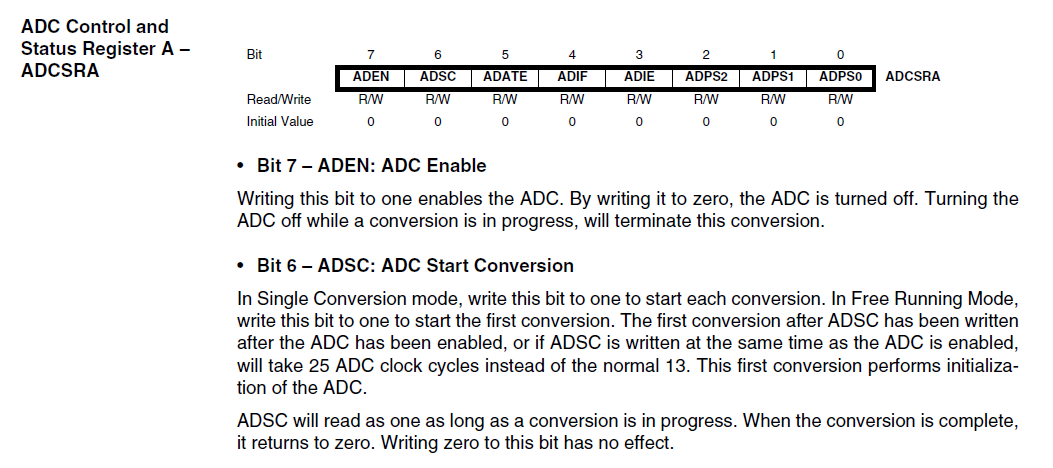
ADMUX &= ~(1 << REFS0);

ADMUX &= ~(1 << REFS1);

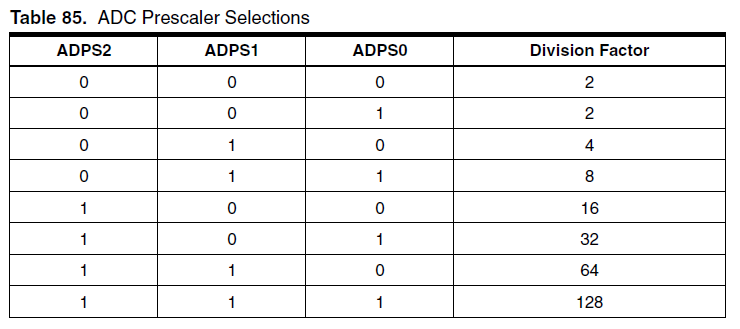
ADMUX |= (1<<MUX1);

}

Ahora vamos al registro de control:

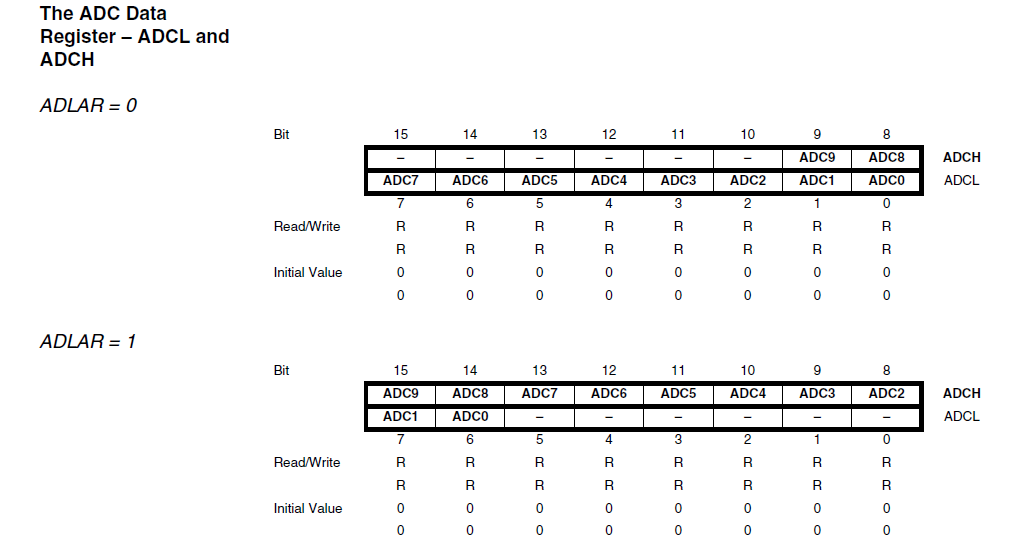


Los bits de este registro son muy importantes, empezando con el bit 7 **ADEN,** hay que poner en 1 al bit 7 para habilitar el ADC y hay que poner el bit 6 **ADSC** en 1 para empezar la conversión, así mismo para **ADATE** que habilita un disparo automático (auto-trigger) **ADIE** para habilitar una interrupción, **ADPS2, ADPS1, ADPS0,** para seleccionar el preescaler:



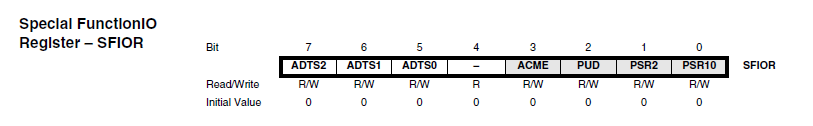
Del preescaler y del reloj que estemos usando depende la frecuencia de muestreo de nuestra señal, por ejemplo si estamos usando un reloj de 8Mhz y escogemos un preescaler de 64, nuestra frecuencia de muestreo será:

Ahora Vamos al registro de los datos del ADC:

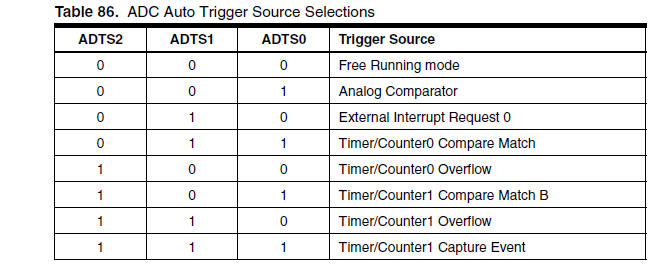


Aquí se tiene dos opciones que son poner el **ADLAR** en uno o ponerel **ADLAR** en cero, esto implica trabajar con una resolución (que tan preciso se quiere el dato) de 10 bit o de 8 bits respectivamente, en este registro es donde se van almacenar los datos que se reciben por las entradas del ADC.

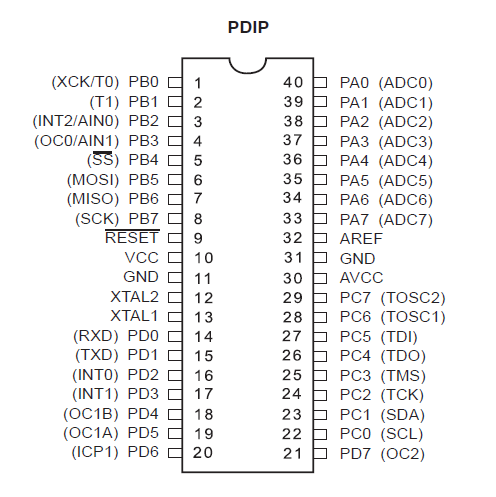
Por ultimo en el caso del Atmega 32 hay un registro denominado **SFIOR(**Special FuncionIO Register**)**



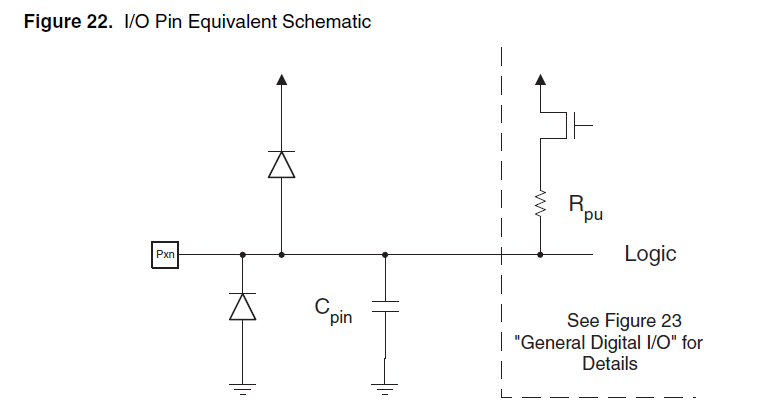
Es un registro de 8 bits, en el cual se puede escoger el modo del ADC, el recomendado es el modo Free Running que implica poner los bits 7,6 y 5 en cero, en este modo la conversión se activará por el trigger y una conversión seguirá inmediatamente termine una.



**CONFIGURACIÓN Y MANEJO DE ENTRAS Y SALIDAS DIGITALES ATMEGA 32**



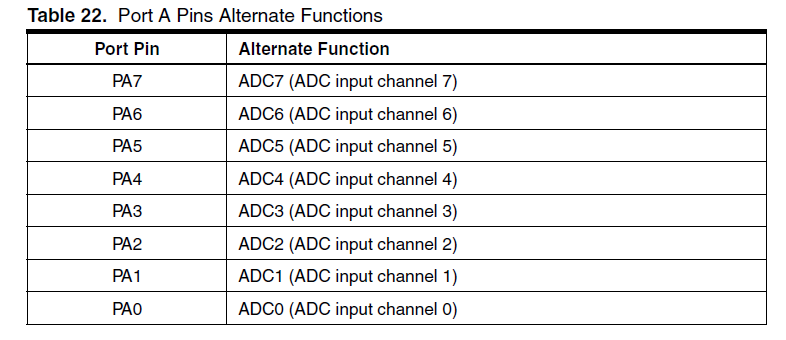
**CIRCUITO EQUIVALENTE A LOS PINES DE ENTRADAS Y SALIDAS:**

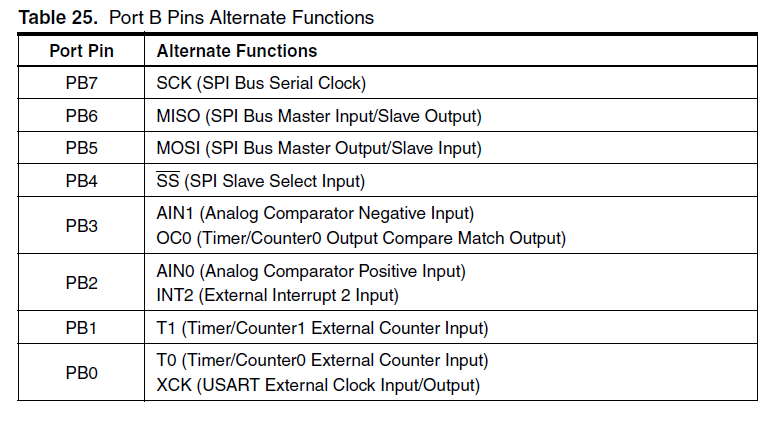


EL atmega 32 cuenta con 40 pines, 32 de los pines se reparten en 4 puertos A,B,C y D.

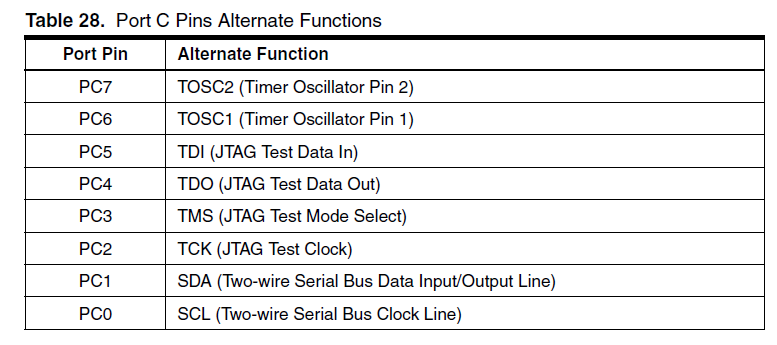
Es importante recalcar que los pines tienen funciones alternativas, a continuación se presentan las funciones de cada uno de los los puertos y sus funciones:

**PUERTO A:**

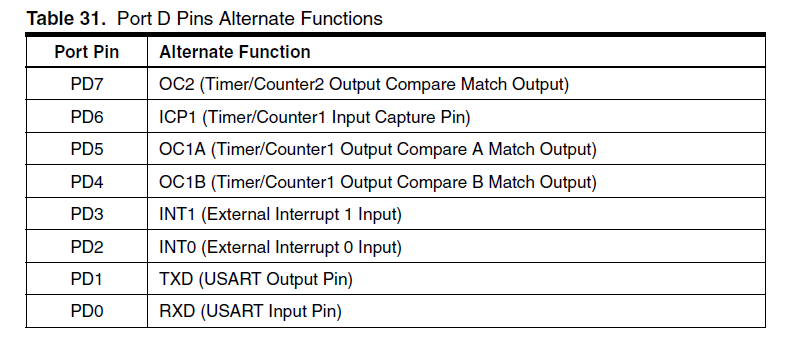
**PUERTO B:**



**PUERTO C:**



**PUERTO D:**



**REGISTRO PARA**

