CONFIGURACION LCD Y TECLADO MATRICIAL EN ATMEL STUDIO.

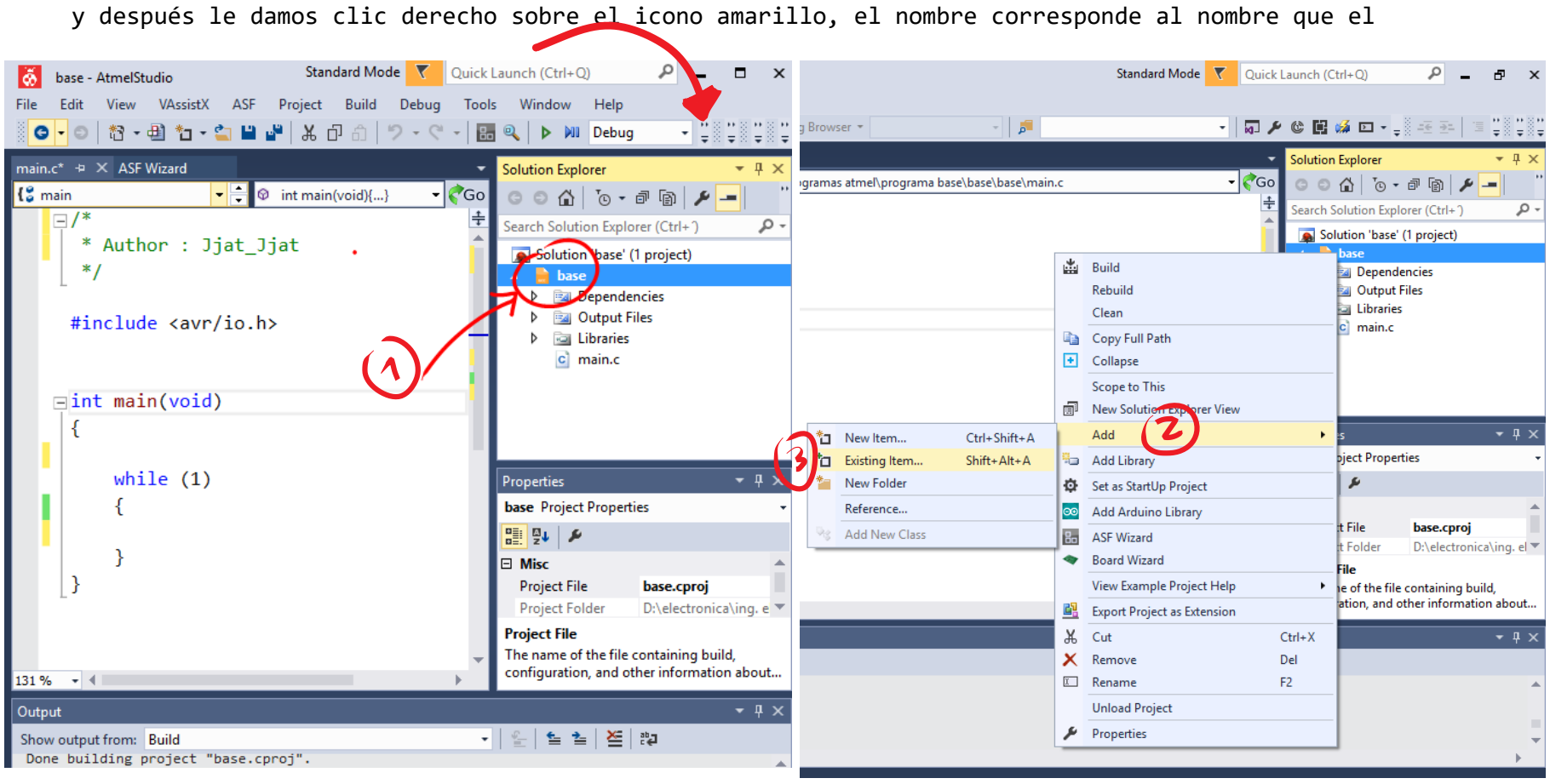
Nombre: Jaimen Aza-1526982

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:

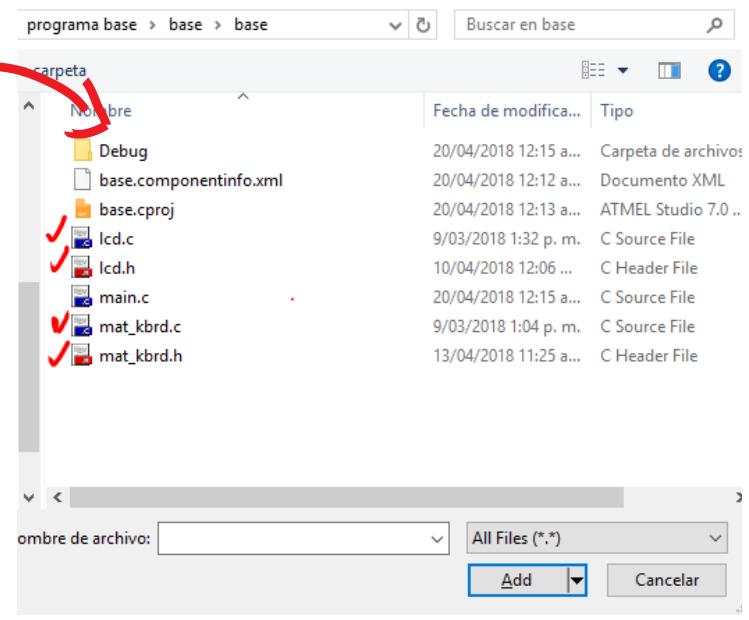
Luego de esto se procede a agregar las librerías en atmel studio, para ello se crea un nuevo proyecto, y después le damos clic derecho sobre el icono amarillo, el nombre corresponde al nombre que el



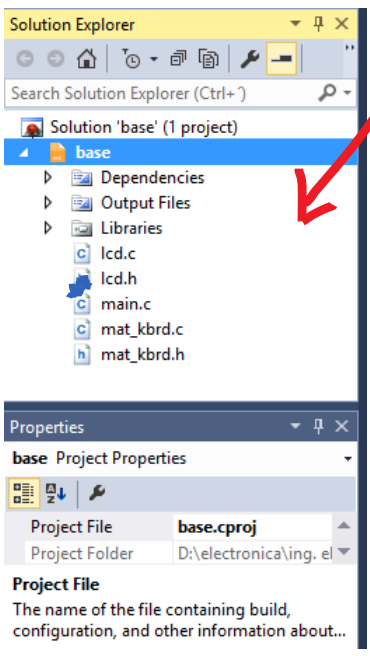
usuario haya asignado al proyecto, después damos clic en add y existing ítem, como se muestra en las siguientes imágenes:

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



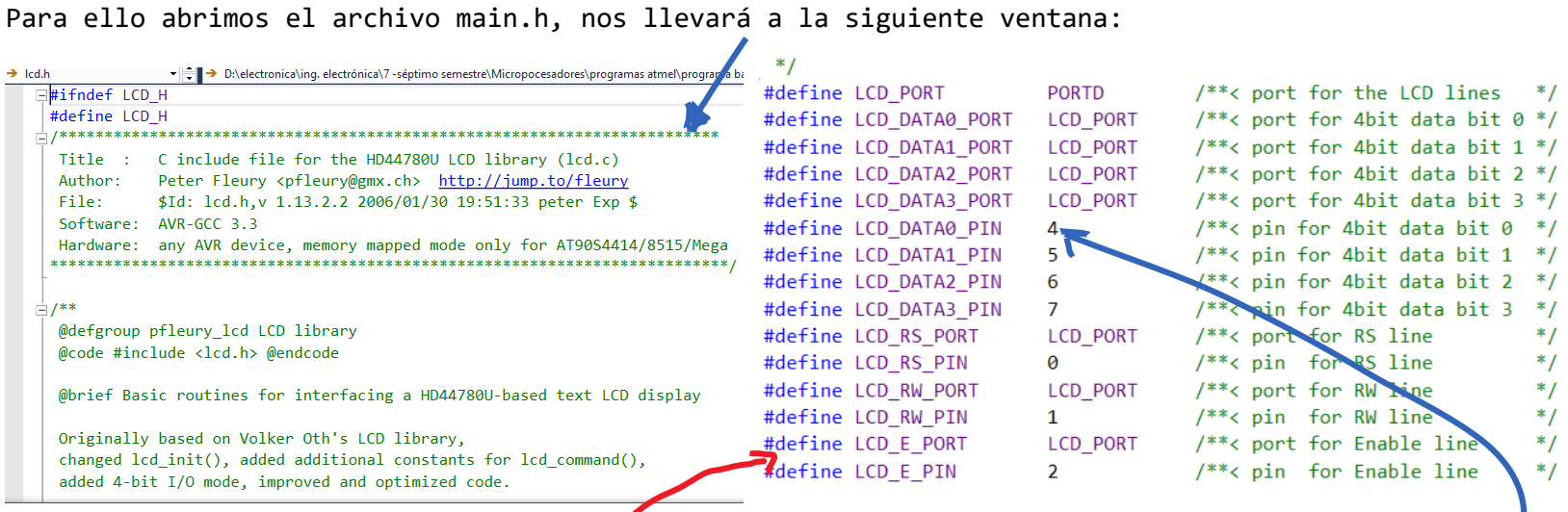
Una vez hecho esto se observará como aparecen en atmel



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:

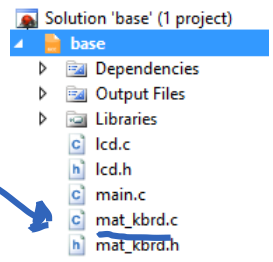
Configuración Puerto del LCD

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



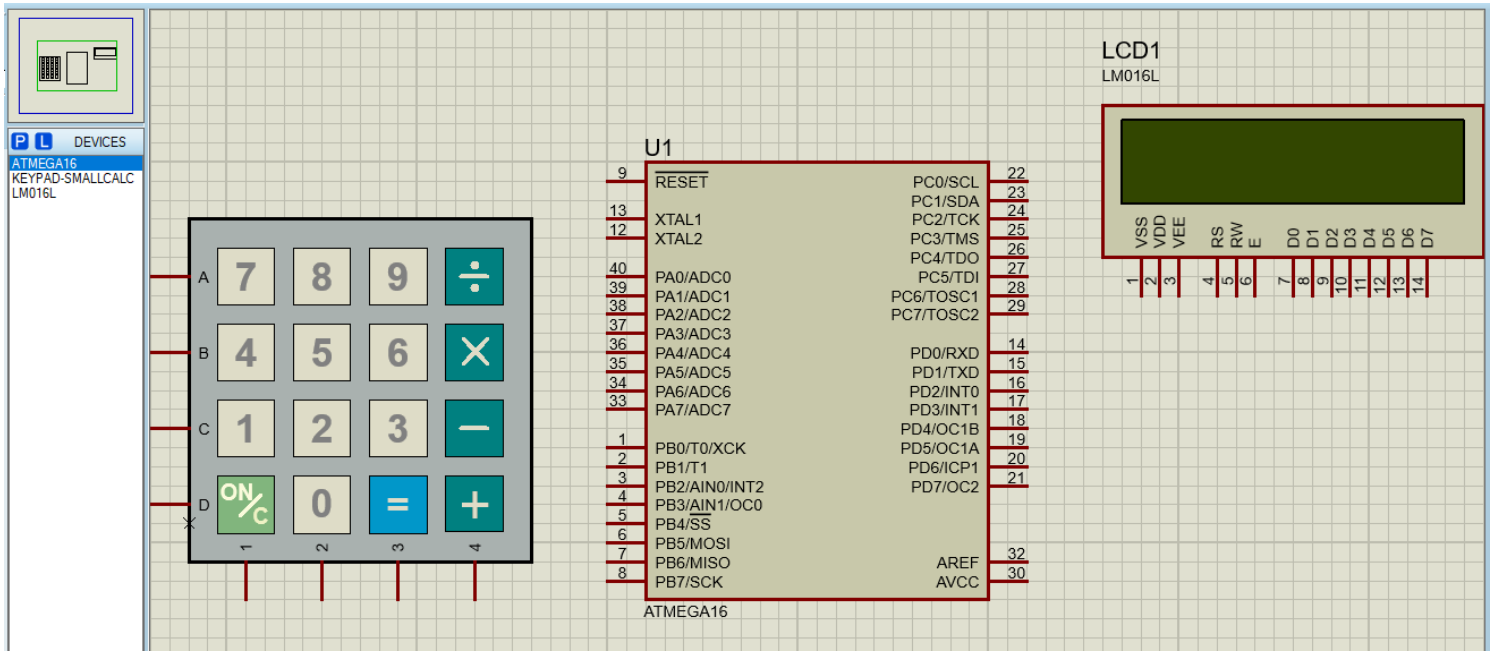
Para configurar el teclado inspeccionamos al archivo mat_kbrd.h

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

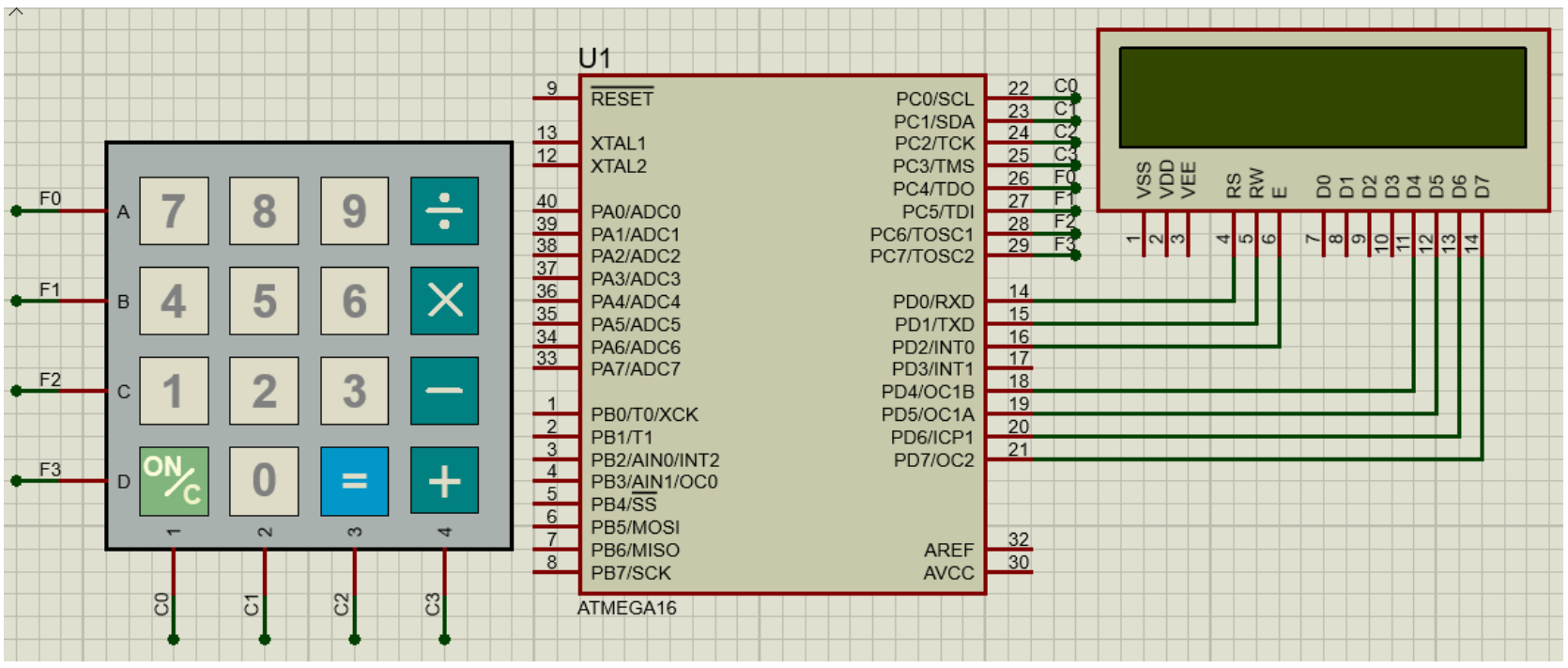
Para ello miramos las conexiones primero para el lCD haciendo clic en lcd.h y hacemos las respectivas conexiones:

```
#define LCD_PORT         PORTD        /**< port for the LCD lines   */
#define LCD_DATA0_PORT   LCD_PORT     /**< port for 4bit data bit 0 */
#define LCD_DATA1_PORT   LCD_PORT     /**< port for 4bit data bit 1 */
#define LCD_DATA2_PORT   LCD_PORT     /**< port for 4bit data bit 2 */
#define LCD_DATA3_PORT   LCD_PORT     /**< port for 4bit data bit 3 */
#define LCD_DATA0_PIN    4            /**< pin for 4bit data bit 0  */
#define LCD_DATA1_PIN    5            /**< pin for 4bit data bit 1  */
#define LCD_DATA2_PIN    6            /**< pin for 4bit data bit 2  */
#define LCD_DATA3_PIN    7            /**< pin for 4bit data bit 3  */
#define LCD_RS_PORT      LCD_PORT     /**< port for RS line         */
#define LCD_RS_PIN       0            /**< pin  for RS line         */
#define LCD_RW_PORT      LCD_PORT     /**< port for RW line         */
#define LCD_RW_PIN       1            /**< pin  for RW line         */
#define LCD_E_PORT       LCD_PORT     /**< port for Enable line     */
#define LCD_E_PIN        2            /**< pin  for Enable li
```

Ahora abrimos los archivos mat_kbrd.c y mat_kbrd.h y miramos las conexiones del teclado:

```
#define KBRD_PORT PORTC
#define KBRD_DDR DDRC
#define KBRD_PINPORT PINC

#define KBRD_C0_PORT KBRD_PORT
#define KBRD_C1_PORT KBRD_PORT
#define KBRD_C2_PORT KBRD_PORT
#define KBRD_C3_PORT KBRD_PORT
#define KBRD_F0_PORT KBRD_PORT
#define KBRD_F1_PORT KBRD_PORT
#define KBRD_F2_PORT KBRD_PORT
#define KBRD_F3_PORT KBRD_PORT
```

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

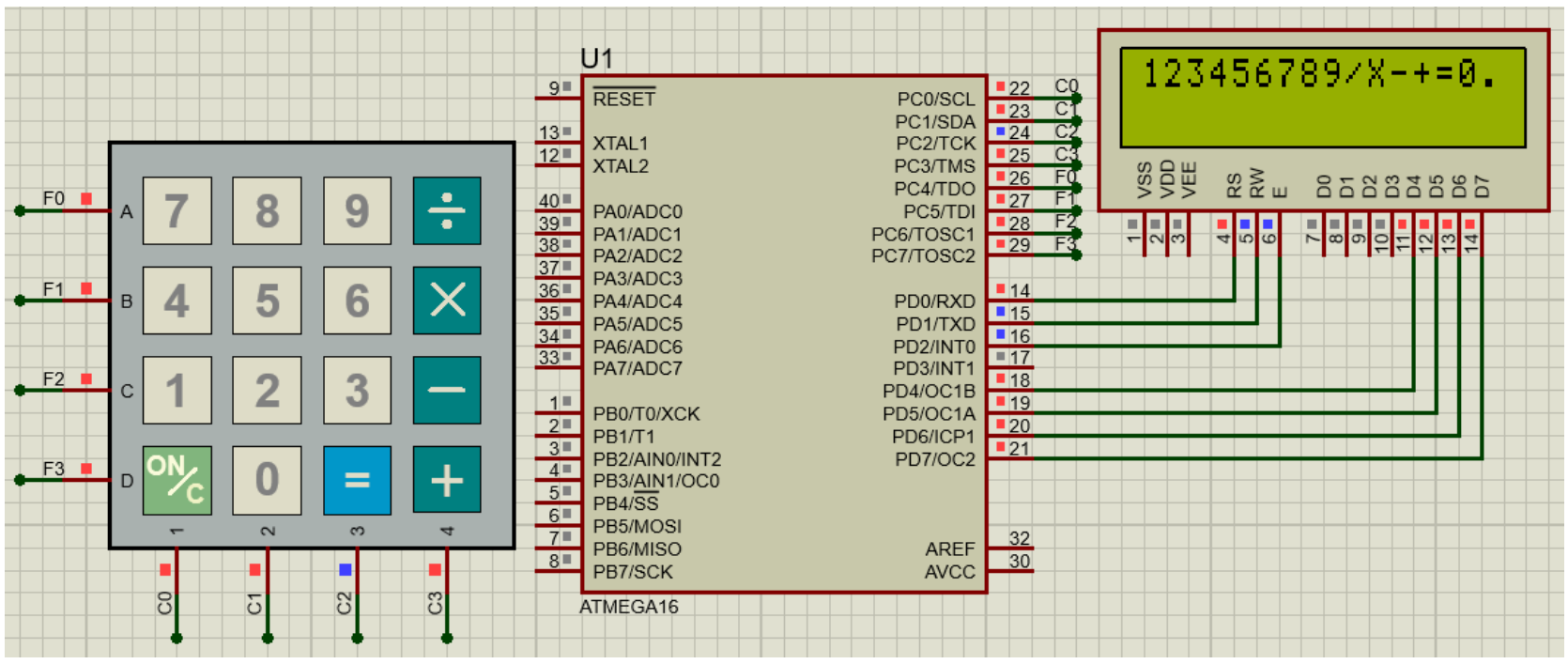
Los comandos básicos para el funcionamiento del LCD y teclado son:

```c
/*
 * Author : Jjat_Jjat
 */

#include <avr/io.h>
#include "lcd.h" //AGREGAR LIBRERIA LCD

int main(void)
{
    lcd_init(LCD_DISP_ON); //INICIALIZA LA LDC, EL PUERTO AL QUE ESTÃ CONECTADO PUERTO D
    lcd_home();            //ENVIAR EL PUNTERO A LA POSICION INICIAL
    lcd_clrscr();          //LIMPIAR LA PANTALLA (CLEAR SCREEN)
    lcd_gotoxy(0,0);       //MUEVE EL CURSOR A LA POSICIÃ“N X=i,Y=0
    lcd_putc();    //IMPRIME EL CARACTER

    kbrd_init(); //INICIA TECLADO
    kbrd_read(); //LEE EL TECLADO
    while (1)
    {

    }
}
```

Ejemplo programa en atmel:

```c
/*
 * 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:

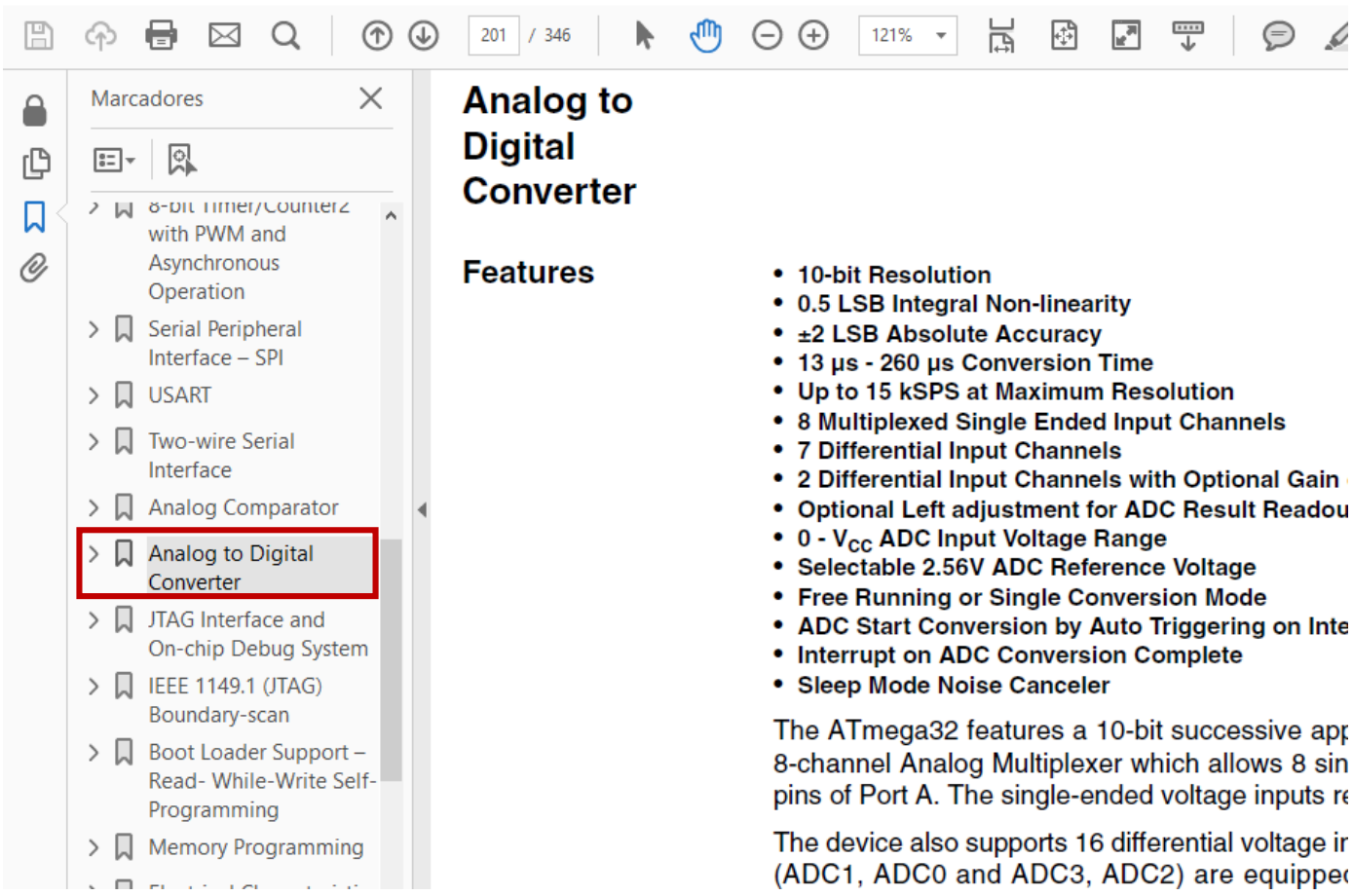
https://drive.google.com/open?id=1MtY01k_nbq7YpKUlMS71EjD3PvGPX_HA

Link Video:

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:**

ADC Multiplexer Selection Register – ADMUX

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – REFS1:0: Reference Selection Bits**

These bits select the voltage reference for the ADC, as shown in Table 83. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set). The internal voltage reference options may not be used if an external reference voltage is being applied to the AREF pin.

Table 83. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, Internal Vref turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

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:

- **Bits 4:0 – MUX4:0: Analog Channel and Gain Selection Bits**

The value of these bits selects which combination of analog inputs are connected to th
These bits also select the gain for the differential channels. See Table 84 for details. If the
are changed during a conversion, the change will not go in effect until this conver
complete (ADIF in ADCSRA is set).

Table 84. Input Channel and Gain Selections

MUX4..0	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
00000	ADC0	N/A		
00001	ADC1			
00010	ADC2			
00011	ADC3			
00100	ADC4			
00101	ADC5			
00110	ADC6			
00111	ADC7			

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:

ADC Control and Status Register A – ADCSRA

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ADEN: ADC Enable**

Writing this bit to one enables the ADC. By writing it to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress, will terminate this conversion.

- **Bit 6 – ADSC: ADC Start Conversion**

In Single Conversion mode, write this bit to one to start each conversion. In Free Running Mode, write this bit to one to start the first conversion. The first conversion after ADSC has been written after the ADC has been enabled, or if ADSC is written at the same time as the ADC is enabled, will take 25 ADC clock cycles instead of the normal 13. This first conversion performs initialization of the ADC.

ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. Writing zero to this bit has no effect.

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:

Table 85. ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

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:

$$\frac{1}{64} * 8000000 = 125Khz$$

Ahora Vamos al registro de los datos del ADC:

The ADC Data Register – ADCL and ADCH

ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	ADC9	ADC8	ADCH
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Aquí se tiene dos opciones que son poner el **ADLAR** en uno o poner el **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**)**

Special FunctionIO Register – SFIOR

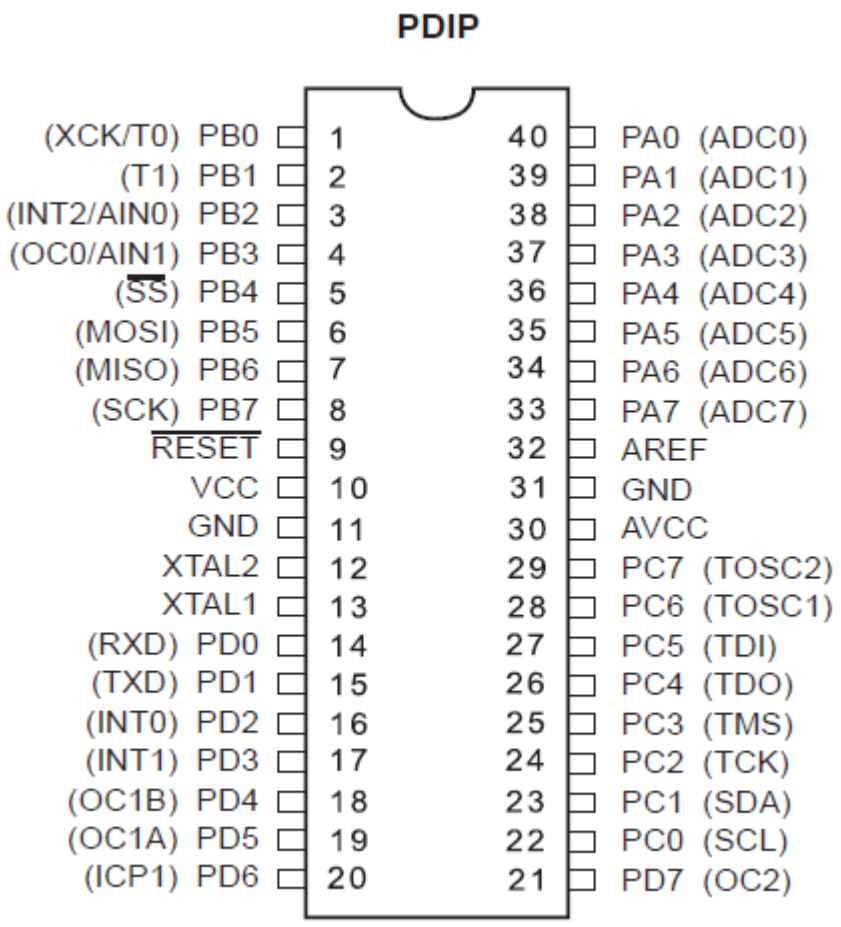
Bit	7	6	5	4	3	2	1	0	
	ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10	SFIOR
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

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.

Table 86. ADC Auto Trigger Source Selections

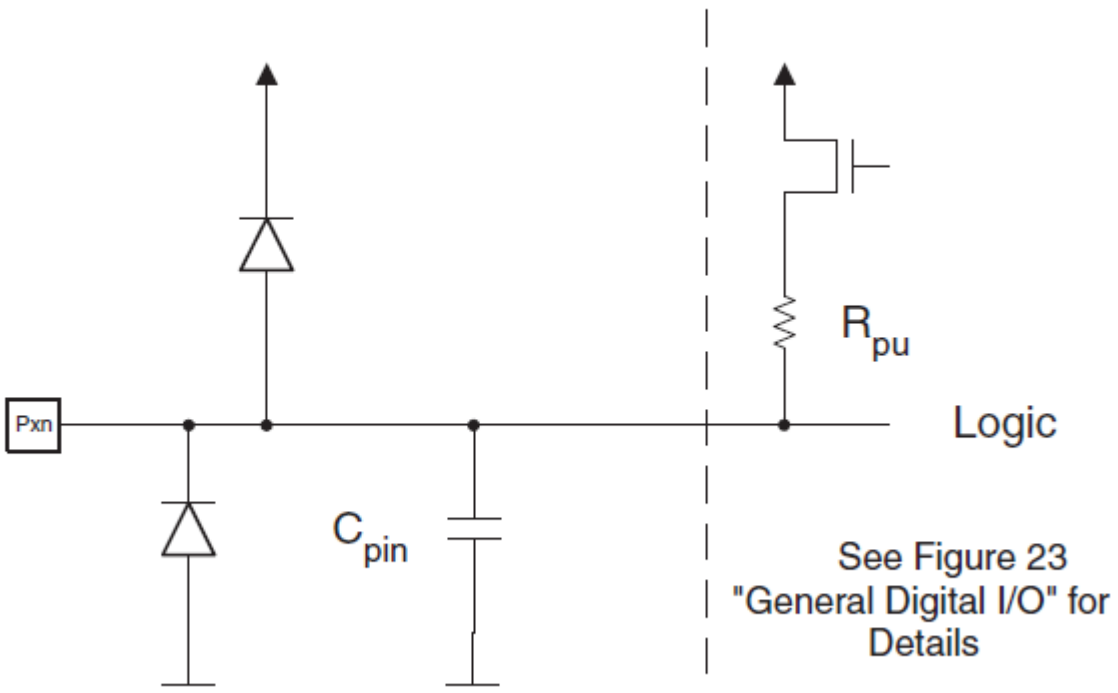
ADTS2	ADTS1	ADTS0	Trigger Source
0	0	0	Free Running mode
0	0	1	Analog Comparator
0	1	0	External Interrupt Request 0
0	1	1	Timer/Counter0 Compare Match
1	0	0	Timer/Counter0 Overflow
1	0	1	Timer/Counter1 Compare Match B
1	1	0	Timer/Counter1 Overflow
1	1	1	Timer/Counter1 Capture Event

CONFIGURACIÓN Y MANEJO DE ENTRAS Y SALIDAS DIGITALES ATMEGA 32



CIRCUITO EQUIVALENTE A LOS PINES DE ENTRADAS Y SALIDAS:

Figure 22. I/O Pin Equivalent Schematic



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:

Table 22. Port A Pins Alternate Functions

Port Pin	Alternate Function
PA7	ADC7 (ADC input channel 7)
PA6	ADC6 (ADC input channel 6)
PA5	ADC5 (ADC input channel 5)
PA4	ADC4 (ADC input channel 4)
PA3	ADC3 (ADC input channel 3)
PA2	ADC2 (ADC input channel 2)
PA1	ADC1 (ADC input channel 1)
PA0	ADC0 (ADC input channel 0)

PUERTO B:

Table 25. Port B Pins Alternate Functions

Port Pin	Alternate Functions
PB7	SCK (SPI Bus Serial Clock)
PB6	MISO (SPI Bus Master Input/Slave Output)
PB5	MOSI (SPI Bus Master Output/Slave Input)
PB4	$\overline{SS}$ (SPI Slave Select Input)
PB3	AIN1 (Analog Comparator Negative Input) OC0 (Timer/Counter0 Output Compare Match Output)
PB2	AIN0 (Analog Comparator Positive Input) INT2 (External Interrupt 2 Input)
PB1	T1 (Timer/Counter1 External Counter Input)
PB0	T0 (Timer/Counter0 External Counter Input) XCK (USART External Clock Input/Output)

PUERTO C:

Table 28. Port C Pins Alternate Functions

Port Pin	Alternate Function
PC7	TOSC2 (Timer Oscillator Pin 2)
PC6	TOSC1 (Timer Oscillator Pin 1)
PC5	TDI (JTAG Test Data In)
PC4	TDO (JTAG Test Data Out)
PC3	TMS (JTAG Test Mode Select)
PC2	TCK (JTAG Test Clock)
PC1	SDA (Two-wire Serial Bus Data Input/Output Line)
PC0	SCL (Two-wire Serial Bus Clock Line)

PUERTO D:

Table 31. Port D Pins Alternate Functions

Port Pin	Alternate Function
PD7	OC2 (Timer/Counter2 Output Compare Match Output)
PD6	ICP1 (Timer/Counter1 Input Capture Pin)
PD5	OC1A (Timer/Counter1 Output Compare A Match Output)
PD4	OC1B (Timer/Counter1 Output Compare B Match Output)
PD3	INT1 (External Interrupt 1 Input)
PD2	INT0 (External Interrupt 0 Input)
PD1	TXD (USART Output Pin)
PD0	RXD (USART Input Pin)

REGISTRO PARA

Port A Data Register – PORTA

Bit	7	6	5	4	3	2	1	0	
	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	PORTA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port A Data Direction Register – DDRA

Bit	7	6	5	4	3	2	1	0	
	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	DDRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port A Input Pins Address – PINA

Bit	7	6	5	4	3	2	1	0	
	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	PINA
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Port B Data Register – PORTB

Bit	7	6	5	4	3	2	1	0	
	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port B Data Direction Register – DDRB

Bit	7	6	5	4	3	2	1	0	
	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port B Input Pins Address – PINB

Bit	7	6	5	4	3	2	1	0	
	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Port C Data Register – PORTC

Bit	7	6	5	4	3	2	1	0	
	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	PORTC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port C Data Direction Register – DDRC

Bit	7	6	5	4	3	2	1	0	
	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	DDRC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port C Input Pins Address – PINC

Bit	7	6	5	4	3	2	1	0	
	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	PINC
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Port D Data Register – PORTD

Bit	7	6	5	4	3	2	1	0	
	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port D Data Direction Register – DDRD

Bit	7	6	5	4	3	2	1	0	
	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Port D Input Pins Address – PIND

Bit	7	6	5	4	3	2	1	0	
	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	