Publicaciones Técnicas
pubtecia@astro.unam.mx

Instituto de Astronomía
UNAM

Comunicación Interna CI-2012-06

aceptación: 27-06-2012
recepción: 04-11-2011

Fuente de voltaje programable.

F. Quirós, S. Zazueta, J.M. Murillo.
Instituto de Astronomía. Universidad Nacional Autónoma de México.
Km. 103 Carretera Tijuana-Ensenada, Ensenada, B.C., México.

RESUMEN:

En el presente trabajo se describe el desarrollo de una fuente programable de voltaje, con un intervalo de salida de 17.5 a 55 Vcd. Dicha fuente está orientada al control de ganancia de un detector intensificado y cuenta con conectividad vía puerto serie RS-232. Se anexan los datos técnicos del sistema así como los mandos de su conectividad.

Contenido

1. INTRODUCCIÓN

El presente proyecto se deriva de la necesidad de controlar la ganancia de un detector intensificado para lo cual utiliza una señal de voltaje en el intervalo de 17.5 a 55 volts.

Por lo anterior se desarrolló un sistema capaz de controlar esta señal, utilizando para su conectividad un puerto serie RS-232.

2. DESCRIPCIÓN GENERAL

El dispositivo se basa en un microcontrolador, un convertidor digital-analógico CDA de 12 bits de resolución y un amplificador de voltaje.

En la *Figura 1* se muestra el diagrama a bloques del dispositivo, el cual también cuenta con un convertidor analógico-digital CAD de 10 bits de resolución para crear una retroalimentación de la salida del sistema; dicha retroalimentación sólo es con fines de sensado.

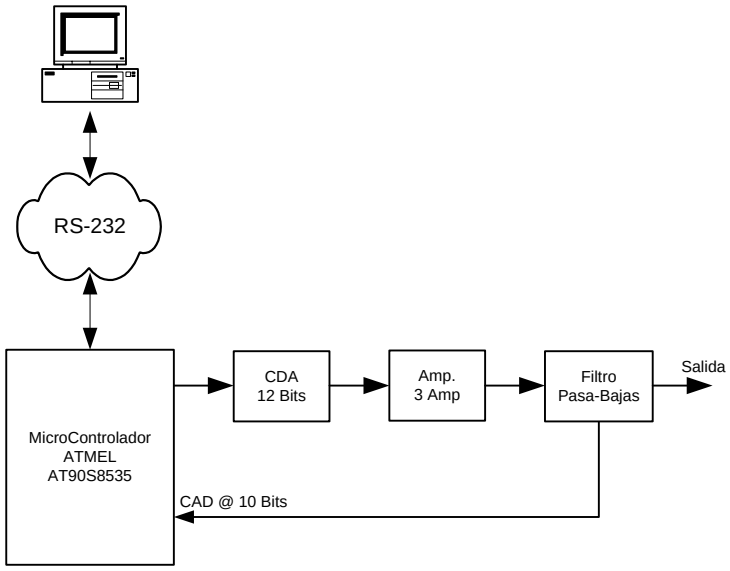

Figura 1: *Diagrama general del sistema.*

2.1 CONVERTIDOR DIGITAL-ANALÓGICO

Para generar la señal analógica se utilizó el convertidor digital-analógico CDA DAC7614 que tiene las siguientes características:

- 12 bits de resolución
- Modo de salida unipolar o bipolar
- Interfaz serie y
- Referencia interna

2.1.1 ASIGNACIÓN DE BITS DE CONTROL

La interfaz serie permite controlar al dispositivo con un número reducido de líneas de control, en la Tabla 1, se muestra la asignación de bits de control del microcontrolador hacia el convertidor digital-analógico CDA. Para mas detalles sobre las líneas de control, vea el Apéndice A.

TABLA 1

Asignación de bits de las señales del CDA.

Asignación de bits de control del µControlador	
PD.2	Reset (Rst)
PD.3	Línea de carga (LoadDAC)
PD.4	Selección de dispositivo (Chip-Select, CS)
PD.5	Señal de reloj (CLK)
PD.6	Línea de datos (SDI)

En la *Figura 2* se muestra la temporización necesaria de las líneas de control, los datos marcados como A1 y A0, controlan el canal de salida del CDA, en nuestro caso sólo utilizamos el canal 0; y la serie de datos D11 a D0 indican el dato a escribir en el CDA, el cual está en el intervalo de 0 a 4095.

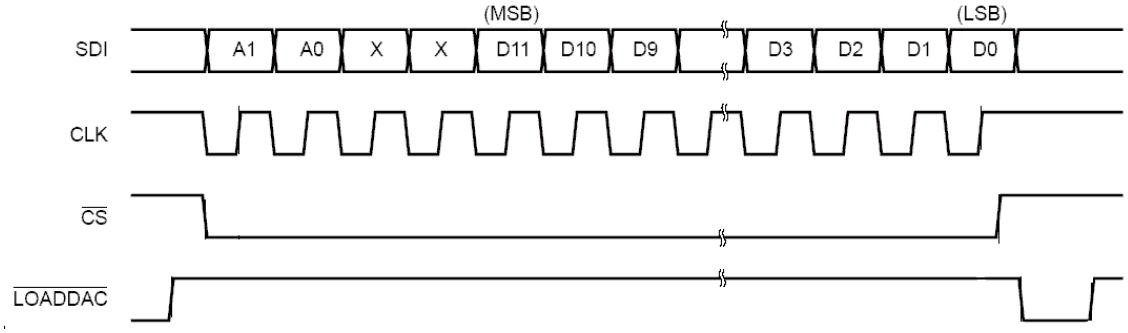

Figura 2: *Temporización para el convertidor digital-analógico.*

2.2 AMPLIFICADOR DE VOLTAJE

Para amplificar en voltaje y corriente la salida del convertidor digital-analógico se utilizó el amplificador operacional LM3875, utilizado normalmente en aplicaciones de amplificación de audio, que tiene las siguientes características:

- Alimentación unipolar o bipolar hasta 84 volts
- Hasta 56 Watts de potencia
- Corriente máxima de salida 6A y
- Relación señal a ruido > 96 dB

En la *Figura 3* se muestra la configuración del amplificador para generar un voltaje de salida de 17.5 a 55 Vcd. Dicha configuración es unipolar, esto es, utilizando sólo una fuente de voltaje.

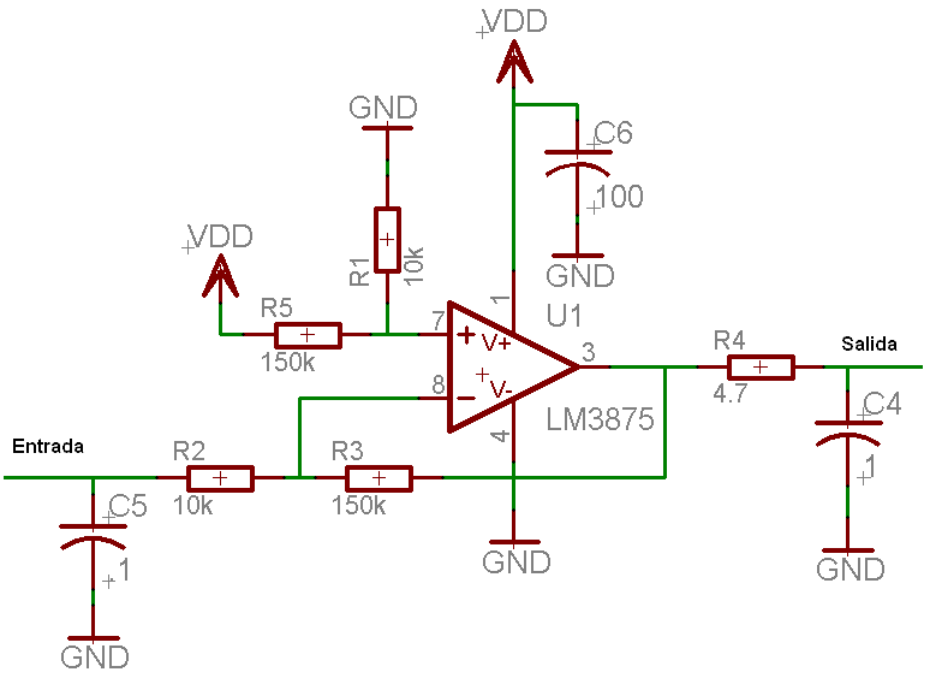

Figura 3: *Amplificador unipolar.*

Analizando el diagrama de la *Figura 3* obtenemos:

$$V_{SALIDA} = V_{DD} - 15*V_{ENTRADA}$$

Si la alimentación V_{DD} es 55 volts, y el intervalo de salida del convertidor digital-analógico es de 0-2.5 volts, obtenemos que el intervalo de salida del amplificador es de:

$$\Delta \text{ Salida} = [V_{DD}, V_{DD} - (15)*(2.5)] = [V_{DD}, V_{DD} - 37.5] = [55, 17.5] \text{ Volts}$$

En términos del número digital (ND) que se le envía al sistema, tenemos:

$$V_{ENTRADA} = (ND.\ 2.5) / (4095) \text{ Volts}$$

De la ecuación anterior obtenemos que el ND en función del voltaje de salida es:

$$ND = (V_{SALIDA} - V_{DD})*4095/37.5$$

2.3. MANDOS SERIE

El dispositivo cuenta con interconectividad vía puerto serie RS-232, por lo que los mandos de control deben de integrar un cierto protocolo adoptado por el IA-UNAM.

Todos los caracteres que manejamos en la comunicación serie son "mayúsculas"; y las características de la comunicación son: 9600 bps, 8 bits de datos, paridad none y 1 bit de paro. Finalmente los mandos series a los que responde se muestran en la Tabla 2.

TABLA 2

Mandos serie a los que responde el dispositivo.

S\n	Pide el estado de la fuente.
V_{ND}\n	Cambia el voltaje de salida con el número ND [4095, 0].

En ambos mandos el dispositivo responde con:

$$S= ND\ V_{OUT}\backslash n,$$

Por ejemplo, si enviamos el mando V0512\n, obtendremos:

$$S=0512\ 22.18\backslash n$$

Donde:

ND (Número Digital), es el número digital enviado como salida al DAC, este valor va en el intervalo de 0 a 4095, con lo cual obtenemos una resolución 0.0091 volts.

V_{OUT} es el voltaje de retroalimentación que lee el microcontrolador con su convertidor analógico-digital integrado.

NOTA:

Hay que tomar en cuenta que un ND de valor 0, genera un voltaje de 0 volts a la salida del CDA, y a su vez V_{DD} a la salida del amplificador, y un ND de 4095 genera una salida de 2.5 volts y 17.5 volts a la salida del amplificador.

3. PRUEBAS DE ESTABILIDAD

En la *Figura 4* se muestran algunos resultados obtenidos; la metodología para obtener estos resultados fue:

a) Se envió un voltaje de salida diferente cada 2 segundos,
b) Se leyó el estado del sistema cada 200 milisegundos.

Se generó un archivo con una serie de datos con el voltaje de referencia y una serie de datos con el voltaje de salida, medido por el microcontrolador.

Los datos obtenidos presentan cierta incertidumbre, ya que el CAD del microcontrolador es de baja resolución (10 bits). En la *Figura 4* se muestran los resultados.

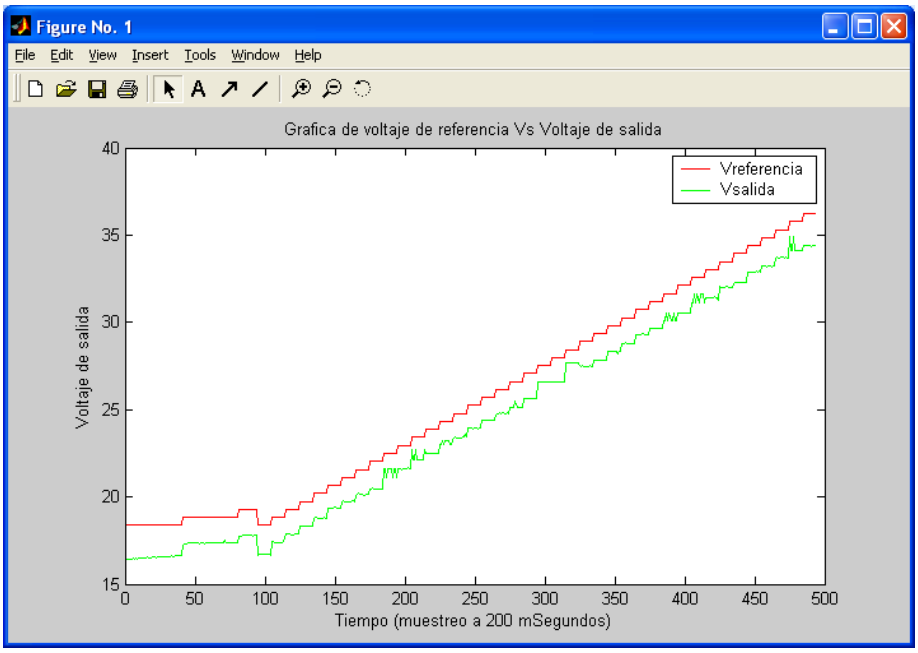

Figura 4: *Gráfica de voltaje de salida contra retroalimentación del microcontrolador.*

La desviación estándar de la diferencia de las dos señales es de 0.2286 volts, la cual se debe a la baja resolución que tiene el convertidor analógico-digital integrado en el microcontrolador.

Para el análisis de los datos, se utilizó el paquete de análisis de datos MATLAB 6.5 y se creó el siguiente programa (fuente.m):

```
load fuente.txt
ref=fuente(:,1);
sal=fuente(:,2);
vi=ref*2.5/4095;
vout=55-15*vi;
plot(vout,'r');
hold on;
plot(sal,'g');
diff=vout-sal;
desviación=std(diff)
```

Donde fuente.txt es el archivo de datos.

4. APÉNDICE A: DIAGRAMAS ELECTRÓNICOS

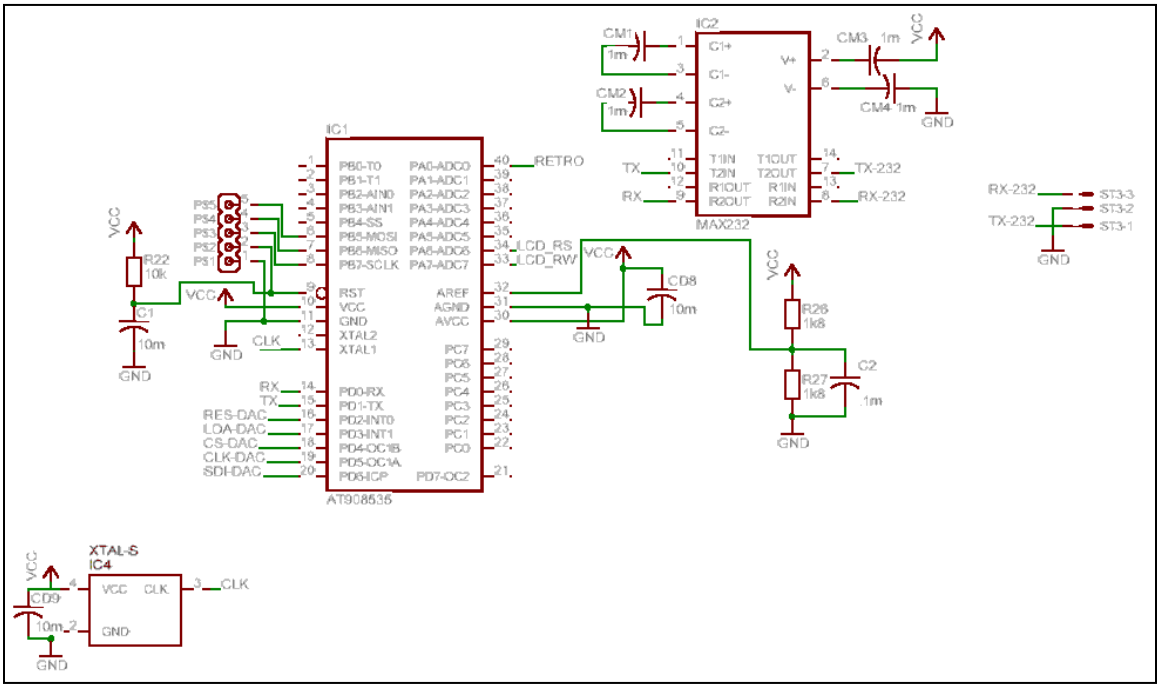

Figura A.1: *Diagrama esquemático de etapa digital.*

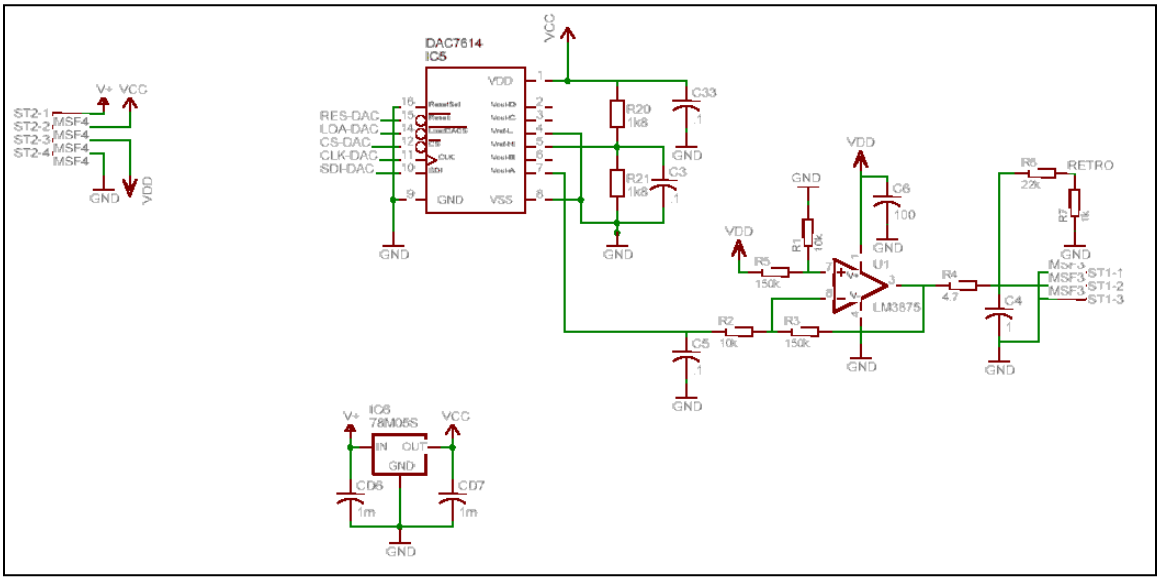

Figura A.2: *Diagrama esquemático de etapa analógica.*

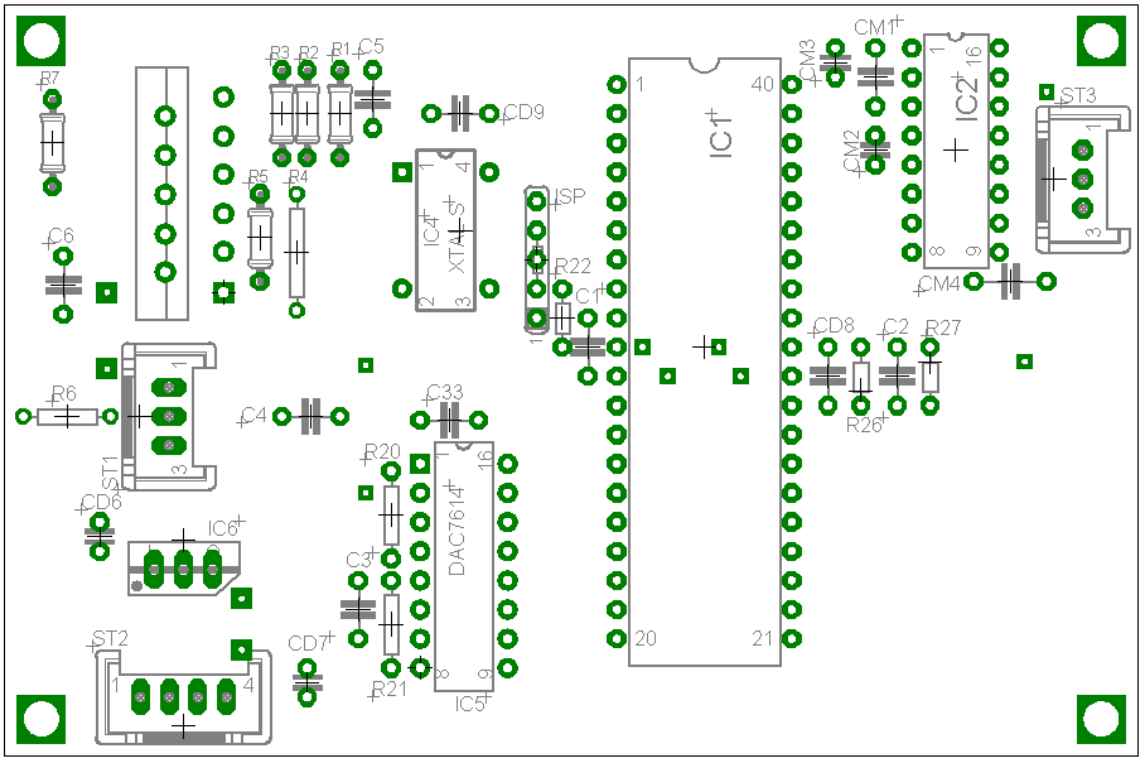

Figura A.3: *Distribución de componentes.*

5. APÉNDICE B: PROGRAMA EN LENGUAJE C PARA EL MICROCONTROLADOR

El programa se realizó en lenguaje C, compilado con ICC-AVR Version 6.29.

```
// Fuente controlada y programable 17.5 - 55 Vcd
// Utiliza un CAD del micro, y un CDA externo (ADC7614)
// Programador:      Fernando Quiros P
// Inicio:           3-octubre-2011
// Modificaccion:    3-octubre-2011

#include <io8535v.h>
#include <macros.h>
#include <stdio.h>
#include <stdlib.h>
#include "fuenteprog.h"

#pragma interrupt_handler UART_RX_interrupt:12 UART_TX_interrupt:13

/* Static Variables */
static unsigned char UART_RxBuf[UART_RX_BUFFER_SIZE];
static volatile unsigned char UART_RxHead;
static volatile unsigned char UART_RxTail;
static unsigned char UART_TxBuf[UART_TX_BUFFER_SIZE];
static volatile unsigned char UART_TxHead;
static volatile unsigned char UART_TxTail;

static volatile unsigned char hay_msg;
unsigned int vout = 4095;

/* initialize UART */
void InitUART( unsigned char baudrate ){
    UBRR = baudrate;                        // set the baud rate

    UCR = ( (1<<RXCIE) | (1<<RXEN) | (1<<TXEN) );
    UART_RxTail = 0;                        // flush receive buffer
    UART_RxHead = 0;
    UART_TxTail = 0;
    UART_TxHead = 0;
    hay_msg = 0;
}
/* interrupt handlers */
void UART_RX_interrupt( void ){
    unsigned char data;
    unsigned char tmphead;
    data = UDR;                             // read the received data
    if (data == '\n') hay_msg = 1;
                                            // calculate buffer index
    tmphead = ( UART_RxHead + 1 ) & UART_RX_BUFFER_MASK;
    UART_RxHead = tmphead;                  // store new index
    if ( tmphead == UART_RxTail ){          // ERROR! Receive buffer overflow
    }
```

```
        UART_RxBuf[tmphead] = data;             // store received data in buffer
}
void UART_TX_interrupt( void ){
        unsigned char tmptail;
        if ( UART_TxHead != UART_TxTail ){              // calculate buffer index
                tmptail = ( UART_TxTail + 1 ) & UART_TX_BUFFER_MASK;
                UART_TxTail = tmptail;                  //store new index
                UDR = UART_TxBuf[tmptail];              //start transmition
        }
        else {
                UCR &= ~(1<<UDRIE);                     //disable UDRE interrupt
        }
}
/* Read and write functions */
unsigned char ReceiveByte( void ){
        unsigned char tmptail;
        while ( UART_RxHead == UART_RxTail ) ;                  //wait for incomming data
        tmptail = ( UART_RxTail + 1 ) & UART_RX_BUFFER_MASK;    //calculate buffer index
        UART_RxTail = tmptail;                                  //store new index
        return UART_RxBuf[tmptail];                             //return data
}
void TransmitByte( unsigned char data ){
        unsigned char tmphead;
                                                                //calculate buffer index
        tmphead = ( UART_TxHead + 1 ) & UART_TX_BUFFER_MASK;
        while ( tmphead == UART_TxTail )                        //wait for free space in buffer
                ;
        UART_TxBuf[tmphead] = data;                             // store data in buffer
        UART_TxHead = tmphead;                                  // store new index
        UCR |= (1<<UDRIE);                                      // enable UDRE interrupt
}
unsigned char DataInReceiveBuffer( void ){
        return ( UART_RxHead != UART_RxTail ); //returno(FALSE) if the receive buffer is empty
}
void delay_ms(int i){
int j,k;
        for(j=0;j<i; j++)
                for(k=0;k<900; k++){}
}
// Lee el CAD, retorna el resultado
unsigned int read_adc(unsigned char adc_input){
        ADMUX=adc_input;
        ADCSR|=0x40;                                            // Inicia la conversión
        while ((ADCSR & 0x10)==0);                              // Espera a que termine
        ADCSR|=0x10;
        return ADC;                                             //dato de 10 bits [0,1024]
}

// Escribe dato a DAC 7614
void write_dac(unsigned int salida){
unsigned char dato;
  clear_dac_cs();
  clear_dac_clk();
  clear_dac_sdi();              //selecciona el CDA (A1)
```

```
set_dac_clk();                  //da clock
clear_dac_clk();
clear_dac_sdi();        //selecciona el CDA (Ao)
set_dac_clk();                  //da clock
clear_dac_clk();
clear_dac_sdi();        //este dato no-importa
set_dac_clk();                  //da clock
clear_dac_clk();
clear_dac_sdi();        //este dato no-importa
set_dac_clk();                  //da clock
clear_dac_clk();

dato = (unsigned char) (salida/256);
if(dato & 0x08) set_dac_sdi();                  //dato 11 (msb)
else            clear_dac_sdi();
set_dac_clk();                                  //da clock
clear_dac_clk();
if(dato & 0x04) set_dac_sdi();                  //dato 10
else            clear_dac_sdi();
set_dac_clk();                                  //da clock
clear_dac_clk();
if(dato & 0x02) set_dac_sdi();                  //dato 09
else            clear_dac_sdi();
set_dac_clk();                                  //da clock
clear_dac_clk();
if(dato & 0x01) set_dac_sdi();                  //dato 08
else            clear_dac_sdi();
set_dac_clk();                                  //da clock
clear_dac_clk();

dato = (unsigned char) (salida & 0x00ff);
if(dato & 0x80) set_dac_sdi();                  //dato 07
else            clear_dac_sdi();
set_dac_clk();                                  //da clock
clear_dac_clk();
if(dato & 0x40) set_dac_sdi();                  //dato 06
else            clear_dac_sdi();
set_dac_clk();                                  //da clock
clear_dac_clk();
if(dato & 0x20) set_dac_sdi();                  //dato 05
else            clear_dac_sdi();
set_dac_clk();                                  //da clock
clear_dac_clk();
if(dato & 0x10) set_dac_sdi();                  //dato 04
else            clear_dac_sdi();
set_dac_clk();                                  //da clock
clear_dac_clk();
if(dato & 0x08) set_dac_sdi();                  //dato 03
else            clear_dac_sdi();
set_dac_clk();                                  //da clock
clear_dac_clk();
if(dato & 0x04) set_dac_sdi();                  //dato 02
else            clear_dac_sdi();
set_dac_clk();                                  //da clock
```

```
  clear_dac_clk();
  if(dato & 0x02) set_dac_sdi();                    //dato 01
  else            clear_dac_sdi();
  set_dac_clk();                                    //da clock
  clear_dac_clk();
  if(dato & 0x01) set_dac_sdi();                    //dato 00
  else            clear_dac_sdi();
  set_dac_clk();                                    //da clock

  set_dac_cs();
  set_dac_cs();

  clear_dac_load();
  set_dac_load();
}

main() {
 unsigned int dato_tmp;
 float decimo, tmp;
 unsigned char buf[32];
 unsigned int i;
 PORTA=0xF0;                        // Puerto A.0-A.3 de entrada y A.4-A.7 de salida
 DDRA=0xF0;
 PORTB=0xFF;                        // Puerto B de salida
 DDRB=0xFF;
 PORTC=0xFF;                        // Puerto C de salida
 DDRC=0xFF;
 PORTD=0x00;                        // Puerto D de salida
 DDRD=0xFF;

 ADCSR=0x86;                        // ADC initialization, ADC Clock frequency: 133.734 kHz
 delay_ms(5);
 set_dac_load();                    // Inicializa el CDA DAC7614
 set_dac_cs();
 set_dac_clk();
 set_dac_reset();
 clear_dac_reset();
 set_dac_reset();

InitUART( 38 );                     //baudrate to 9600 bps using a 6MHz crystal
 _SEI();                        // enable interrupts => enable UART interrupts

sprintf(buf,"Fuente 0-55 Vcd v1.0\n");  //al inicio manda quien es
for(i=0;i<22;i++)
  TransmitByte( buf[i]);

 while (1) {
  write_dac(vout);

//rutina de pto serie
  if( hay_msg == 1){                        //llego un mando
    hay_msg = 0;
    for(i=0;i<16;i++){                      //barre el buffer
      if( DataInReceiveBuffer() != 0){      //indica que hay caracter
```

```
            buf[hay_msg] = ReceiveByte();
                    hay_msg++;
            }
           }
           hay_msg = 0;
       if( buf[0] == 'S'){                                    //manda el set point
            dato_tmp = read_adc(0);
            tmp = (float)(dato_tmp)/17.803;
         decimo = tmp - ((int) tmp);
         if(decimo < 0) decimo = -1*decimo;
         sprintf(buf,"S=%d %d.%d\n       ", vout, (int)tmp, (int)(100*decimo));
         for(i=0;i<16;i++)
            TransmitByte( buf[i]);
       }
       if( buf[0] == 'V'){                              //Cambia el set point
         sprintf(buf,"%c%c%c%c", buf[1],buf[2],buf[3],buf[4]);
            vout = atoi(buf);
            if(vout>4095) vout=4095;
            write_dac(vout);
            delay_ms(50);

            dato_tmp = read_adc(0);
            tmp = (float)(dato_tmp)/17.803;
         decimo = tmp - ((int) tmp);
         if(decimo < 0) decimo = -1*decimo;
         sprintf(buf,"S=%d %d.%d\n       ", vout, (int)tmp, (int)(100*decimo));
         for(i=0;i<16;i++)
            TransmitByte( buf[i]);
       }
      }
      delay_ms(250);
     }
     return 0;
    }
```