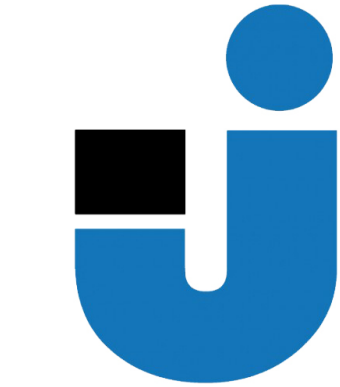


Universidad Nacional
ARTURO JAURETCHE

Programación en Tiempo Real

Trabajo Práctico nº 1

Interrupciones, reloj y adquisición de datos

Profesor
Diego Encinas

Autores
Cristian Sanabria
Gerónimo Bazán
Emiliano Salvatori

28 de septiembre de 2020

Índice

1. Introducción

En el siguiente informe se detalla lo realizado como parte del laboratorio de la materia **Programación en Tiempo Real** para la **Comisión nº 1**. El presente trabajo se basa en lo solicitado para la Práctica nº 1 sobre **Manejo de Interrupciones, reloj y adquisición de datos** realizando una investigación y simulación con PIT y con una placa adquisidora ADQ12.

Para ello se utiliza el entorno de programación Borland C sobre un sistema operativo Windows 10.

2. Cuestionario

2.1. Ejercicio nº 1

Realice una breve investigación sobre el funcionamiento del PIT (Programmable Interval Timer) 8253/8254. Tenga en cuenta los siguientes aspectos:

- Función y sub-funciones de cada registro
- Asociación de registros con dispositivos de la PC
- Ubicación en memoria de cada registro

2.2. Ejercicio nº 2

Escriba un programa que instale un manejador de interrupción y modifique la frecuencia de interrupción del reloj (PIT 8253/8254). Dicho programa debe recibir un parámetro que indique la frecuencia de interrupciones por segundo e imprimir cada segundo la cantidad de veces que se invocó el manejador de interrupciones. Después de 20 segundos el programa debe finalizar.

2.3. Ejercicio nº 3

Realice una investigación sobre la placa ADQ12.

1. Describa las características técnicas de la placa.
2. Describa las funciones y sub-funciones de los registros CTREG, INSTR, OUTBR, ADLOW, ADHIGH.
3. Indique el procedimiento de lectura de un valor analógico con interrupciones y sin interrupciones. Explique el porqué de las diferencias

2.4. Ejercicio nº 4

Un motor tiene conectado un sensor de temperatura que proporciona 0° para 0V y 120° para 5V. Este sensor mide la temperatura del motor. Cuando esta supera los 65° se activa el sistema de ventilación que está conectado a la salida digital 1. Si la temperatura llegara a alcanzar los 85° el motor debe apagarse utilizando el canal digital 0. Escriba un programa que adquiera periódicamente la temperatura y controle las salidas digitales de la forma planteada anteriormente. Grafique la señal analógica y las líneas de temperatura (utilizando graphics.h) en un sistema de ejes cartesianos donde el eje **x** indica el momento de la captura y el eje **y** indica el valor adquirido.

3. Resoluciones

3.1. Ejercicio nº 1

3.1.1. Funcionamiento del PIT 8253/8254

El 8253/54 es un chip temporizador que puede ser empleado como reloj de tiempo real, contador de sucesos, generador de ritmo programable, generador de onda cuadrada, etc

3.1.2. Diferencia entre 8253 y 8254

A continuación se visualizan las diferencias más notables entre ambos dispositivos:

PIT 8253	**PIT 8254**
Opera en una frecuencia que va desde 0 a los 2.6 MHz	Opera en una frecuencia que va desde 0 a 10 MHz
Usa la tecnología N-MOS	Utiliza la tecnología H-MOS
El comando Read-Back (*Leer hacia atrás*) no se encuentra habilitado	El comando Read-Back (*Leer hacia atrás*) se encuentra habilitado
Las lecturas y escrituras del mismo contador no se pueden intercalar.	Las lecturas y escrituras del mismo contador se pueden intercalar

3.1.3. Registros, funciones y sub-funciones

Los registros que posee este temporizador son 4, entre ellos hay 3 contadores y un registro de control de palabra. Los contadores son etiquetados como Contador 0, Contador 1 y Contador 2, estos son de 16 bits y pueden programarse para trabajar en modo binario o en modo BCD (Decimal codificado en binario). Los contadores cuentan desde el número que se les asigne hasta cero regresivamente, son independientes uno del otro, y pueden ser leídos fácilmente por el CPU. En el chip, cada contador tiene 3 pines asociados a él

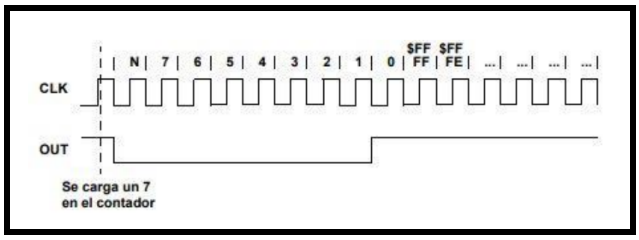
1. **CLK** (reloj): Es un pin de entrada por donde se recibe una señal de reloj para el contador. El contador contará hacia abajo decrementándose al ritmo de esta señal de sincronización.

2. **GATE** (puerta): Es un pin de entrada que recibe una señal para controlar el contador. Su significado varía de acuerdo al modo de operación.

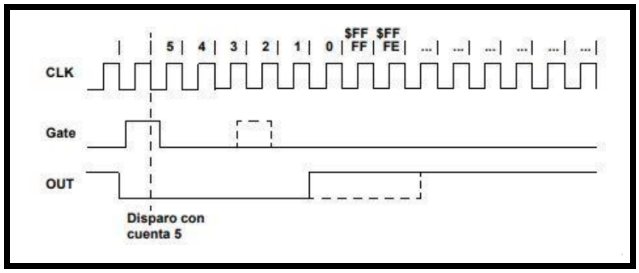
3. **OUT** (salida): Es un pin por donde sale una señal que se comporta de acuerdo al modo de operación con que fue programado el contador.

Por otro lado, el registro de control de palabra permite configurar el modo de operación de los contadores, estos modos pueden ser configurados en 5 formas diferentes:

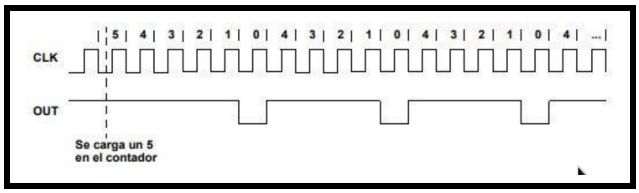
- **Modo 0 (Ciclo simple)**: Es empleado normalmente para contar sucesos, la salida se mantiene a nivel bajo hasta que finaliza la cuenta. Por ejemplo, se carga el valor 7 de conteo y el contador solo cuenta si GATE posee el valor 1



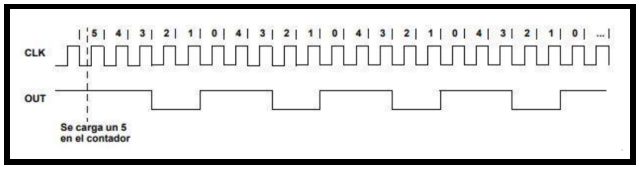
- **Modo 1 (Ciclo simple redisparable)**: La cuenta se inicia o reinicia en el primer flanco de bajada de la señal del reloj con GATE en estado alto. Por lo cual, un cambio en el valor de cuenta inicial no tiene efecto hasta que se produzca un redisparo. Por ejemplo, se carga el valor 5 de conteo y la salida del contador se mantiene a nivel bajo hasta que finaliza la cuenta.



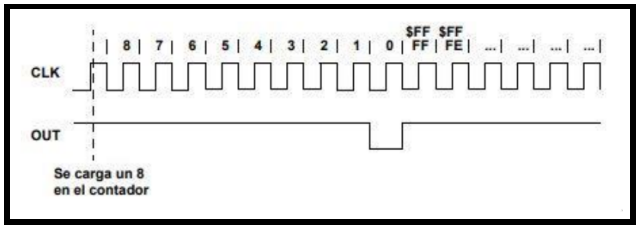
- **Modo 2 (Interrupción periódica)**: La salida se mantiene a nivel alto mientras la cuenta es distinta de 0 y a nivel bajo mientras la cuenta es 0. Por ejemplo, se carga el valor 5 de conteo y en la salida del contador se genera una onda periódica donde cada vez que se llegue al valor 0 comienza nuevamente con el conteo.



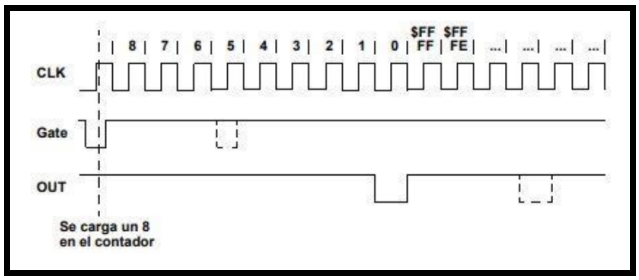
- **Modo 3 (Generador de onda cuadrada)**: Emite una onda cuadrada repetida una y otra vez, con una frecuencia regulada por el contador. Este modo es similar al modo 2. Sin embargo, la duración de los pulsos de reloj alto y bajo de la salida serán diferentes a los del modo 2, ya que, la salida permanecerá en alto durante los primeros ciclos y en bajo durante los últimos ciclos. Por ejemplo, se carga el valor 5 de conteo y en la salida de contador se genera una onda cuadrada donde cada vez que se llegue a la mitad del conteo la onda de salida cambia su valor



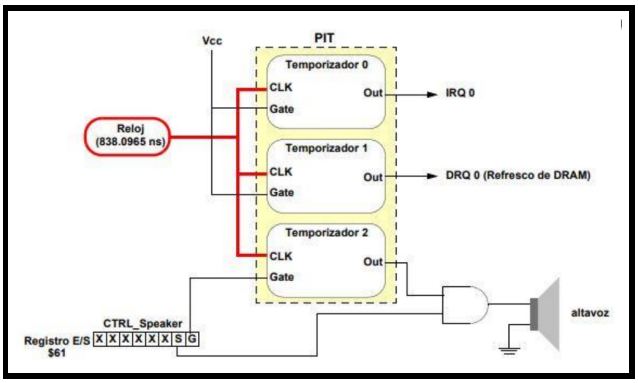
- **Modo 4 (Interrupción sobre fin de cuenta)**: La salida se mantiene a nivel alto mientras que el conteo llegue al nivel nulo. Por ejemplo, se carga el valor 8 de conteo y en la salida de contador se mantiene en alto hasta que finaliza el conteo



- **Modo 5 (Interrupción redisparable)**: Emite un único pulso al terminar el contador, La puerta GATE se usa para iniciar el conteo regresivo para el disparo del pulso. Este modo es similar al modo 4. Sin embargo, el proceso de conteo es disparado por la entrada GATE, por lo cual se entiende que es redisparable, ya que, por cualquier flanco positivo en GATE. Por ejemplo, se carga el valor 8 de conteo y en la salida del contador se mantiene a nivel alto hasta que el valor sea nulo, es decir, 0



3.1.4. Asociación de registros con dispositivos de la PC



Como se puede observar en la imagen los registros se relacionan con 3 dispositivos de la PC.

- **Contador 0**: Está conectado a IRQ 0 (ligado a la INT 8, que a su vez invoca a INT 1Ch); este contador está programado por defecto con el valor cero (equivalente a 65536), por lo que la cadencia de los pulsos es de 1.193.180/65.536 = 18,2 veces por segundo, valor que determina la precisión del reloj del sistema, ciertamente demasiado baja. Se puede modificar el valor de recarga de este contador en un programa, llamando a la vieja INT 8 cada 1/18,2 segundos para no alterar el funcionamiento normal del ordenador, ya que sincroniza todos los componentes por este medio. Si bien no es conveniente instalar programas residentes que cambien permanentemente esta especificación, los programas del usuario esperan encontrarse el temporizador a la habitual y poco útil frecuencia de 18,2 interrupciones/segundo.

- **Contador 1**: Controla el refresco de memoria en todas las máquinas, su valor normal para el divisor es 18; aumentándolo se puede acelerar el funcionamiento del ordenador, con el riesgo de un fallo en la memoria, detectado por los chips de paridad que provoca generalmente el bloqueo del equipo. De todas maneras, en los PC/XT se puede aumentar entre 19 y 1000 sin demasiados riesgos, acelerándose en ocasiones hasta casi un 10 % la velocidad de proceso del equipo. En los AT la ganancia de velocidad es mucho menor y además este es un punto demasiado sensible que conviene no tocar para no correr riesgos, aunque se podría bajar hasta un valor 2-17 para ralentizar el sistema.

- **Contador 2**: Puede estar conectado al altavoz del ordenador para producir sonido; alternativamente puede emplearse para temporizar. Es el único contador que queda realmente libre para el usuario, lo que suele dar quebraderos de cabeza a la hora de producir sonido.

3.1.5. Ubicación en memoria de cada registro

Dirección E/S	Tipo de Operación	Función
$40	Escritura	Establece el valor inicial del temporizador 0.
$40	Lectura	Lee cuenta y/o el estado del temporizador 0.
$41	Escritura	Establece el valor inicial del temporizador 1.
$41	Lectura	Lee cuenta y/o el estado del temporizador 1.
$42	Escritura	Establece el valor inicial del temporizador 2.
$42	Lectura	Lee cuenta y/o el estado del temporizador 2.
$43	Escritura	Establece Modo de Control.
$43	Lectura	Operación NO permitida.

3.2. Ejercicio nº 2

A continuación se visualiza el código que se solicita en el **Ejercicio nº 2**

```
1  #include<stdio.h>
2  #include<conio.h>
3  #include<dos.h>
4  #include<math.h>
5  #include<stdlib.h>
6
7  #define LOW_BYTE(n) (n & 0x00ff)
8  #define HIGH_BYTE(n) ((n>>8)&0x00ff)
9
10 /* Se la declara como unsigned log ya que solo
11    podra tomar valores entre 0 y 65535 */
12 unsigned long int frecteclado = 0;
13
14 /* Se declaran variables que se utilizaran para realizar
15    el conteo de interrupciones en 20 segundos*/
16 int x=0;
17 int cont=0;
18 int flag=0;
19 int y=0;
20
21 void interrupt (*restablecedor) ();
22 void SetFrecuencia(int TeclFrec, int *contViejo, int *
       contViejo2);
23
24 /*Se crea la funcion que se encarga de instalar el
       manejador
25   de interrupciones y modificar la frecuencia del mismo*/
26 void SetFrecuencia(int TeclFrec,int *contViejo, int *
       contViejo2) {
27
28     /* La frecuencia base del interruptor es 1.193 MHz*/
```

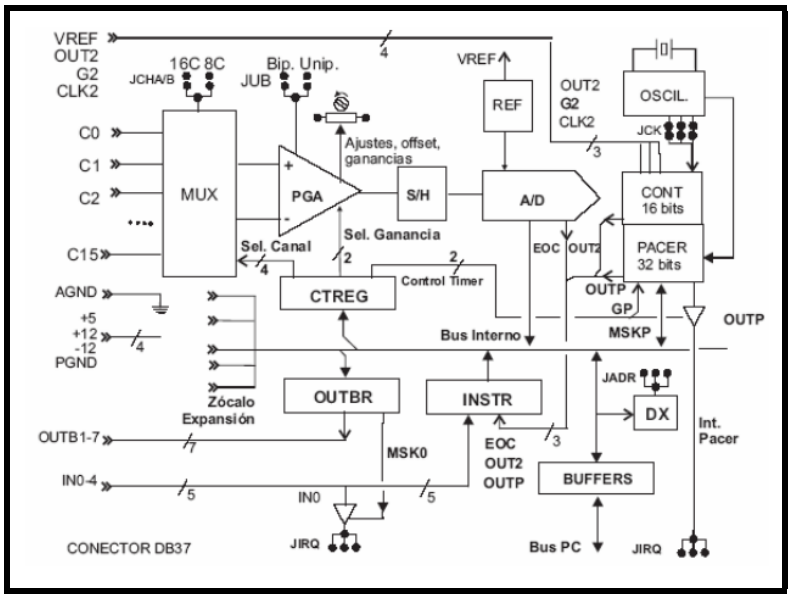
```
29      long int frec=1193180;
30
31      /* Modificar la frecuencia del temporizador */
32      frec = frec / TeclFrec;
33      outportb(0x43,0x36); /* escribe el byte del control
            de palabras */
34
35      /* Se guarda la parte baja del contador viejo */
36      *contViejo = inportb(0x40);
37
38      /* Se guarda la parte alta del contador viejo */
39      *contViejo2 = inportb(0x40);
40
41      /* Se escribe el nuevo contador bajo  */
42      outportb(0x40,LOW_BYTE(frec));
43
44      /* Se escribe el nuevo contador alto */
45      outportb(0x40,HIGH_BYTE(frec));
46      clrscr();
47
48  }
49
50  void interrupt mimanejador() {
51      flag=1;
52  }
53
54  /* Funcion principal */
55  void main(void) {
56
57      int contViejo, contViejo2;
58      printf("Ingrese divisor de frecuencia deseada entre
            19 y 65535: ");
59      scanf("%d", &frecteclado);
60
61      if (frecteclado >= 19 && frecteclado <= 65535){
62
63           SetFrecuencia(frecteclado,&contViejo, &
                contViejo2);
64
65      } else {
66
67          printf("Ingrese nuevamente la frecuencia entre 19
                y 65535: ");
68          scanf("%d", &frecteclado);
69
70          if (frecteclado >= 19 && frecteclado <= 65535){
71
72                   SetFrecuencia(frecteclado,&contViejo, &
                        contViejo2);
73
74          } else {
75
76                  printf("INCREIBLE, pero le erro dos veces
                        \n");
77                  printf("Presione cualquier tecla para
                        finalizar");
78                  getch();
```

```
79                  exit(0);
80          }
81      }
82
83      /* Se salva el estado del manejador antes de la
            modificacion */
84      restablecedor=getvect(0x1C);
85
86      /* Se coloca el nuevo estado */
87      setvect(0x1C,mimanejador);
88
89      while ( cont<20 ) {
90
91          if (flag==1) {
92                  y++;
93                  x++;
94                  flag=0;
95
96                  /* Se escribe la cantidad de
                        interrupciones por segundo*/
97                  printf("Interrupcion: %d\n",y);
98
99                  /* Se cuenta la cantidad de segundos
                        transcurridos
100                    por cada bloque de iterrupciones*/
101                 if (y==frecteclado) {
102                         y=0;
103                         cont++;
104                         printf("Segundo: %d\n", cont);
105                 }
106          }
107
108     }
109
110     printf("\n");
111     printf("Cantidad de segundos en total: %d\n",cont);
112     printf("Cantidad de interrupciones totales: %d\n",x);
113
114     /* Antes de terminar el programa, se restaurar
115       el estado origina del manejador */
116     setvect(0x1C,restablecedor);
117
118     /* Se restauran los contadores*/
119     outportb(0x40,contViejo);
120     outportb(0x40,contViejo2);
121
122     printf("Presione cualquier tecla para finalizar");
123     getch();
124 }
```

3.3. Ejercicio nº 3

3.3.1. Placa ADQ12

La ADQ12 es un sistema modular de adquisición y control. La placa cuenta con dos secciones, una analógica y otra digital, como puede apreciarse en la figura siguiente:



3.3.2. Descripción técnica del Hardware

Conversor AD 12 bits:

- Canales: 16 desbalanceados, 8 diferenciales.
- Tiempo de conversión: 10μSeg.
- Ganancia software programable: 4 rangos.
- Entradas bipolares: $\pm$5, $\pm$2, $\pm$1, $\pm$0.5 Voltios.
- Entradas unipolares: +5, +2, +1, +0.5 Voltios.

Entradas - Salidas Digitales:

- Port de salida de 8 bits, programable bit a bit.
- Port de entrada de 5 bits.

Timers

- Contador de 16 bits, reloj interno ó externo.
- Pacer de 32 bits, reloj 16μS a 2h: 23m, soft seleccionable.

Interrupciones Enmascarables (hard y soft):

- Dos entradas de interrupciones enmascarables.
- Nivel de IRQ seleccionable (7, 5, 4, 3, 2/9)

3.3.3. Registro de Control CTREG

Este registro permite definir las funciones programables que dispone la placa **ADQ12-B**. Las funciones y los nombres de cada bit son los que siguen.

Bit	Nombre	Función
Bit 7	MSKP	**1**: Habilita la interrupción del PACER. **0**: Enmascara la interrupción del PACER.
Bit 6	GTP	**1**: Dispara el PACER. **0**: Bloquea el PACER.
Bit 5-4	AX1 - AX0	Control de los 4 rangos de ganancia del PGA **00**: Fondo de escala 5.0V **01**: Fondo de escala 2.0V **10**: Fondo de escala 1.0V **11**: Fondo de escala 0.5V
Bit 3-0	MX3 - MX0	Selección de los canales analógicos

Cuando se opera en modo diferencial, los bits 2 a 0 seleccionan uno entre 8 de los canales analógicos, en el presente caso el bit 3 no cumple función alguna. Cuando se opera en modo desbalanceado, los bits 3 a 0 seleccionan 1 entre 16 canales analógicos. El registro de control es forzado a cero luego de un encendido ó un reset del computador (ALT-CTRL-DEL no fuerza un reset)

3.3.4. Puerto de Entrada y Registro de Estado STINR

Este registro combina dos funciones: los 5 bits de menor peso sirven para el ingreso de señales digitales y se hallan disponibles en el conector DB37, IN0 a IN4.

La entrada IN0 tiene además capacidad de interrupción. Los tres bits de mayor peso revelan el estado del conversor A/D pacer y contador.

Bit	Nombre	Función
Bit 7	OUT2	Indica el estado de salida del Timer2, CONT2.
Bit 6	OUTP	Indica el estado de salida del PACER.
Bit 5	EOC	**1**: Fin de conversión. **0**: Conversión en curso.
Bit 4	IN4	Estado de la línea de entrada IN4
Bit 3	IN3	Estado de la línea de entrada IN3
Bit 2	IN2	Estado de la línea de entrada IN2
Bit 1	IN1	Estado de la línea de entrada IN1
Bit 0	IN0	Estado de la línea de entrada IN0

3.3.5. Puerto de salida OUTBR

OUTBR es un puerto de salida de 8bits. La programación de los estados de salida se realiza en forma individual, bit a bit.

Siete de ellos se hallan disponibles en el conector IOC (DB37): del OUTB1 al OUTB7. El bit cero, OUTB0, sirve a efectos de enmascarar la entrada de interrupción IN0. Las salidas OUTBR son forzadas a cero tras el encendido o mediante un Reset del computador.

El control de los bits se realiza enviando a la dirección de OUTBR un byte con las siguientes características: los tres bits de menor peso (B2, B1, B0) señalan uno de los 8 bits de salida, en tanto el valor del bit 3 será colocado en la salida señalada. Al definir un determinado bit de salida, los restantes mantienen su valor. Los bits 7 a 4 no desempeñan función alguna.

Función	**B3**	**B2**	**B1**	**B0**
OUTB5 toma el valor 1	1	1	0	1
OUTB6 toma el valor 0	0	1	1	0
OUTB6 toma el valor 1	1	1	1	0
OUTB0 toma el valor 0	0	0	0	0

3.3.6. Conversor Analógico - Digital A/D

La placa **ADQ12** utiliza un conversor AD de 12 bits modelo ADC912A ó similar. La obtención del valor convertido se deberá realizar en dos lecturas sucesivas, primero el byte alto ADHIG, y luego el bajo ADLOW. En el byte alto son significativos los 4 bits de menor peso, los restantes tendrán el valor cero.

Una señal de fin de conversión (EOC) en estado alto, indica que se cuenta con un nuevo valor digitalizado. El estado de la señal EOC puede obtenerse del bit 5 del registro STINR. El inicio de la conversión se realizará por lectura del byte bajo, ADLOW. Son posibles dos procedimientos para la adquisición, estos se describen en el próximo punto.

El tiempo de conversión de la unidad conversora AD es de 10,0 μSeg. Este tiempo es el que media entre el inicio de la conversión y el instante en que EOC indica que el dato se halla disponible. Para determinar la velocidad de adquisición, es decir si el número de muestras por segundo sobre un determinado canal, intervienen además los siguientes factores:

1. Lectura y conformación del dato (luego que EOC = 1).
2. Almacenamiento del dato en memoria. Incremento de punteros y decremento de contadores.
3. Transformación del dato en valores equivalentes de tensión (variable real en presición simple).
4. Presentación numérica y/ó gráfica en tiempo real.

3.4. Ejercicio nº 4

A continuación se visualiza el código que se solicita en el **Ejercicio nº 4**

```c
#include <dos.h>
#include <stdio.h>
#include <graphics.h>
#include <stdlib.h>
#include<math.h>
#include <time.h>
#include <conio.h>

/*Defino para los contadores del temporizador*/
#define LOW_BYTE(n) (n & 0x00ff)
#define HIGH_BYTE(n) ((n>>8)&0x00ff)

/*Variables para dibujar*/
int xa,xn,ya,yn,k=0;
int flag=0,cont=0, y=0;
unsigned char A, B, C;
float VA;

/*Registros de la placa ADQ12*/
unsigned int DIRBAS, OUTBR, STINR, CTREG, ADHIG, ADLOW,
    VD;

/*Se crea un restablecedor para poner los valores como
  estaban en un principio del interruptor*/
void interrupt (*restablecedor) ();

void SetFrecuencia(int TeclFrec, int *contViejo, int *
    contViejo2);

/*Se crea la funcion que se encarga de instalar el
    manejador
  de interrupciones y modificar la frecuencia del mismo*/

void SetFrecuencia(int TeclFrec,int *contViejo, int *
    contViejo2)
{
    long int frec=1193180; /* La frecuencia base del
        interruptor es 1.193 MHz*/

    /* Modificar la frecuencia del temporizador */
    frec = frec / TeclFrec;

    /* Escribe el byte del control de palabras */
    outportb(0x43,0x36);

    /* Se guarda la parte baja del contador viejo */
    *contViejo = inportb(0x40);

    /* Se guarda la parte alta del contador viejo */
    *contViejo2 = inportb(0x40);

    /* Se escribe el nuevo contador bajo */
    outportb(0x40,LOW_BYTE(frec));

    /*Se escribe el nuevo contador alto */
    outportb(0x40,HIGH_BYTE(frec));
    clrscr();
```

```
53  }
54
55  /*Defino mi manejador para controlar las interrupciones*/
56  void interrupt mimanejador()
57  {
58      flag=1;
59  }
60
61
62  /*Prototipos de las funciones*/
63  void calculartemp(void);
64  void grafico(void);
65
66  /*Calculamos la temperatura*/
67  void calculartemp(){
68
69          /*Inicia la conversion en parte baja AD */
70          C = inportb (ADLOW);
71
72          /*Espera la senial del bit 5(EOC) en 1 del STINR
73          que indica el fin de la conversion*/
74          do {
75                  C = inportb (STINR);
76
77                  /* En la variable C carga el bit EOC */
78                  C = C && 0x20;
79
80          }while (C == 0x0); /* Continua leyendo hasta que
                EOC = 1*/
81
82          /* Una vez finalizada la conversion,
83          retoma la lectura del valor convertido
84          de 12 bits 4 en high y 8 en low*/
85          A = inportb (ADHIG);
86          B = inportb (ADLOW);
87          /* Conforma un nro de 12 bits.*/
88          VD = A * 256 + B;
89
90          /*Pasamos VD a voltios.*/
91          VA = (float) 5 / 4096 * VD;
92          /*printf ("\n \n%1.3f Voltios \n \n \n", VA);*/
93  }
94
95
96  /*Funcion para graficar*/
97  void grafico(){
98
99     char ey,ex;
100    int i,j;
101    int gd = DETECT, gm;
102    initgraph(&gd, &gm, "..\\BGI");
103
104    setlinestyle(SOLID_LINE,0,2);
105
106    line(100,460,100,60);
107    line(100,460,600,460);
108    line(90,70,100,60);
```

```
109     line(110,70,100,60);
110     line(590,450,600,460);
111     line(590,470,600,460);
112
113     outtextxy(85,35,"VOLTAJE/TEMPERATURA");
114     outtextxy(613,452,"SEGUNDOS");
115
116     setlinestyle(DOTTED_LINE,0,2);
117
118     for(i=460,j=0;i>=60;i-=80,j++){
119          line(100,i,600,i);
120          itoa(j,&ey,10);
121          outtextxy(85,i-5,&ey);
122     }
123
124     for(i=100;i<=600;i+=25){
125          line(i,460,i,60);
126          itoa(k,&ex,10);
127          outtextxy(i,465,&ex);
128          k++;
129     }
130
131     setcolor(RED);
132     setlinestyle(CENTER_LINE,0,2);
133 }
134
135
136 void main(void){
137
138          /* variables necesarias para los contadores del
                SetFrecuencia*/
139          int contViejo, contViejo2;
140
141          //Direccion base del ADQ12
142          DIRBAS = 0x300;
143
144          //Port de salida de 8 bits
145          OUTBR = DIRBAS + 4;
146
147          //Port de entrada 5 bits + reg status
148          STINR = DIRBAS + 0;
149
150          //Reg control funciones programables
151          CTREG = DIRBAS + 0;
152
153          //Conversor analogico-parte alta
154          ADHIG = DIRBAS + 9;
155
156          //Conversor analogico-parte baja
157          ADLOW = DIRBAS + 8;
158
159          /* Se setea la frecuencia para el temporizador*/
160          SetFrecuencia(19,&contViejo, &contViejo2);
161
162          /* Se salva el estado del manejador antes de la
                modificacion */
163          restablecedor=getvect(0x1C);
```

```
164
165         /* Se coloca el nuevo estado */
166         setvect(0x1C,mimanejador);
167
168         /* Selecciona: canal 1, FE: +5V*/
169         outportb (CTREG,0);
170
171         /*escribe el byte en el port CTREG el byte 0,
172         seleccion de canal analogico bit 0 a 2 (
                diferencial)
173         seleccion de canal analogico bit 0 a 3 (
                desbalanceado)*/
174
175         calculartemp();
176         grafico();
177
178         /*inicio grafico*/
179         xa=100;
180         ya=(int)(-80*VA)+460;
181
182         /*el programa se reproduce en 40 segundos,
                pasados los 40 segundos
183           se cierra, puede ser cualquier cantidad de
                 segundos
184           que acepte el temporizador*/
185
186         while ( cont<40 ){
187         if(flag==1){/*Si cumplo un segundo*/
188
189                 y++; //incremento el tiempo en el que se
                        reproduce la linea
190                 calculartemp();/*Calculo VA*/
191
192                 /*Se controla que la linea roja no pase
                        la pantalla, una vez que pasa la
                        pantalla se refrezca nuevamente el
                        grafico*/
193                 if(xa==600){
194                         clearviewport();
195                         xa=100;
196                         grafico();
197                         }
198         /*Se controla que cada cierto tiempo se dibuje la
                linea roja sobre el grafico*/
199                 if (y==19){
200                         y=0;
201                         cont++;
202                         xn=xa+25;
203                         yn=(int)(-80*VA)+460;
204                         line(xa,ya,xn,yn);
205                         xa=xn;
206                         ya=yn;
207                         }
208                 if(VA>3.54){//si la temperatura es mayor
                        a 85 grados se prende la salida
                        digital 0
209                         outportb(OUTBR, 0x08);
```

```
210                     }
211                     else{
212                             outportb(OUTBR, 0x00);
213                             //si no, se apaga
214                     }
215                     if(VA>2.71){//si la temperatura es mayor
                            a 65 grados se prende la salida
                            digital 1
216                             outportb(OUTBR, 0x09);
217                     }else{
218                             outportb(OUTBR, 0x01);
219                             //si no, apago el bit 1
220                     }
221                     flag=0;
222
223                     }
224             }
225
226             /*Restablezco Timer original*/
227             setvect(0x1C,restablecedor);
228
229             /*se restauran los contadores*/
230             outportb(0x40,contViejo);
231             outportb(0x40,contViejo2);
232
233             closegraph();
234             printf("presione una tecla para salir");
235             getch();
236             exit(0);
237
238     }
```