(19) OFICINA ESPAÑOLA DE PATENTES Y MARCAS

ESPAÑA

(11) Número de publicación: **2 120 826**

(21) Número de solicitud: 9402453

(51) Int. Cl.[6]: G08C 25/00

(12) PATENTE DE INVENCION B1

(22) Fecha de presentación: **24.11.94**

(43) Fecha de publicación de la solicitud: **01.11.98**

Fecha de concesión: **09.04.99**

(45) Fecha de anuncio de la concesión: **01.06.99**

(45) Fecha de publicación del folleto de patente: **01.06.99**

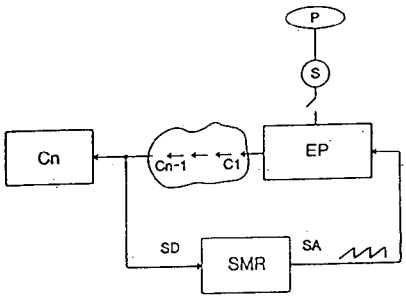
(73) Titular/es: **Universidad de Sevilla, Vicerrectorado de Investigación y Transferencia Tecnológica**
Valparaíso, 5 - 2ª Planta
41013 Sevilla, ES

(72) Inventor/es: **Luque Rodríguez, Joaquín y Escudero Fombuena, José Ignacio**

(74) Agente: **No consta**

(54) Título: **Sistema para la medida de retrasos en instrumentación distribuida.**

(57) Resumen:
Sistema para la medida de retrasos en instrumentación distribuida, que permite determinar los retrasos en la transmisión que afectan a los datos obtenidos mediante sistemas de instrumentación y control distribuidos. El sistema de medida de retrasos (SMR) propuesto es una unidad de proceso, gobernada por un software específico, que permite medir el tiempo que transcurre desde que un equipo de instrumentación primario (EP) lee el valor de un sensor S, medida directa de un proceso físico P, hasta que dicho valor llega, mediante comunicaciones intermedias, a un equipo concentrador final C_n; para ello el sistema de medida está conectado formando un bucle, cuya entrada es una señal digital SD de comunicación de datos (la misma que llega al concentrador), conectada internamente a un interfaz de comunicaciones, y cuya salida es una señal analógica SA de instrumentación en diente de sierra, obtenida internamente a partir de un convertidor digital/analógico, que se inyecta en el equipo de instrumentación primario (EP). Es de aplicación a cualquier sistema de instrumentación y/o control distribuido, como por ejemplo el telecontrol de redes de energía eléctrica.



ES 2 120 826 B1

DESCRIPCION

Sistema para la medida de retrasos en instrumentos distribuida.

Objeto de la invención

La presente invención se refiere a un sistema que permite determinar los retrasos que afectan a los datos obtenidos mediante sistemas de instrumentación y control distribuidos. Estos retrasos, asociados a la velocidad de variación de las magnitudes físicas, hacen que los datos considerados se encuentren afectados por un error que es conveniente considerar en cualquier sistema de instrumentación y control. Es de aplicación, con ligeras modificaciones para cada caso particular, a prácticamente cualquier sistema de instrumentación y/o control distribuido; en concreto se describe una arquitectura adaptada a un sistema de telecontrol de redes de energía eléctrica.

Estado de la técnica

En buen número de procesos industriales, a poco que la complejidad del mismo lo justifique, se realiza un seguimiento de las magnitudes físicas más significativas que permitan, por un lado observar y conocer su evolución, y por otro actuar sobre las variables de control del mismo, de forma manual o automática, para conseguir operar el proceso de manera óptica desde un punto de vista técnico y económico. Surgen con ello los equipos de instrumentación (con o sin control asociado) que captan los datos de diversas magnitudes físicas, los presentan a operadores humanos, los procesan y, eventualmente los transmiten a otros equipos. Dada la creciente complejidad técnica de la mayoría de los procesos industriales, los criterios de explotación técnica y económica cada vez más exigentes, y la disponibilidad de equipos de instrumentación a precios más reducidos, existe una tendencia a aumentar el número de equipos de instrumentación (y control) presentes en un preceso productivo.

Con frecuencia, por razones operativas y por su diferente ubicación física, los equipos de instrumentación se hallan físicamente distribuidos en un área geográfica que, dependiendo de la naturaleza del proceso, puede abarcar una simple planta industrial, una ciudad, una región o todo el planeta. Obviamente la información suministrada por los diferentes equipos de instrumentación, distribuidos de esta manera, ha de ser concretada en uno o varios equipos concentradores, constituyendo de esta forma un "sistema de instrumentación (y control) distribuido". En estos sistemas los datos captados por los sensores y equipos de instrumentación de primer nivel, son transmitidos con distintos criterios a equipos concentradores de nivel superior. Análogamente, los datos de equipos concentradores pueden transmitirse a otros concentradores del mismo o superior nivel, constituyendo así el sistema de instrumentación distribuido.

Las medidas que se manejan en un sistema de instrumentación (y control) distribuido están afectadas por un error que depende de dos factores: la evolución de las magnitudes físicas y los retrasos de transmisión. La determinación del primero de estos factores, no presenta dificultades, existiendo buen número de equipos en el mercado que permiten el seguimiento de la evolución de una magnitud física. Más dificultades entraña, sin embargo, el segundo de los factores: el retraso. Hay que señalar que este retraso no es debido únicamente a los tiempos de transmisión por los medios de comunicación sino que incluyen también los tiempos de procesos en los distintos equipos del sistema distribuido, así como las esperas, cíclicas o aleatorias, antes de que un dato se transmita de un equipo a otro. El sistema de medida que se describe en la presente Memoria permite, de una forma sistemática, determinar estos retrasos de transmisión.

Descripción general de la invención

El sistema de medida de retrasos (SMR) propuesto, es un dispositivo capaz de medir el tiempo que ha pasado desde que se tomó en campo una medida determinada, hasta el instante actual, en que esa medida se utiliza en algún ordenador del sistema distribuido con fines de control del mismo.

Para ello el dispositivo genera una señal analógica (SA) que posee forma de diente de sierra. Esta forma de la señal, cuyos parámetros son manejables por el usuario para cada caso, permite que, conocidos dos valores de la misma, se pueda calcular con exactitud el intervalo de tiempo que los separa.

Esta señal (SA) se inyecta en el equipo primario (EP) en sustitución de las medidas reales que el sensor (S) del equipo lleva a cabo. El equipo primario (EP) toma ese valor de la señal analógica (SA) como si fuera el valor medio de una magnitud física y lo envía a través de los canales de comunicación del sistema distribuido, como señal digital (SD).

De esta manera, el propio sistema de medida de retrasos (SMR) analiza la línea de comunicaciones que une el sistema distribuido con el ordenador en estudio. Cuando detecta un mensaje con datos, almacena el valor actual de la señal analógica (SA) que se está inyectando en esos momentos en el equipo primario

(EP) y compara ese valor con el transmitido a través del sistema de medida de retrasos (SMR) a través de la señal digital (SD). Dada la forma de la señal analógica (SA) inyectada es fácil obtener el tiempo transcurrido entre los dos valores, que al fin y al cabo es el retraso con el que las medidas llegan a ese ordenador concreto del sistema distribuido.

Descripción detallada de la invención

Es consustancial a los sistemas de instrumentación (y control) distribuido, dada su arquitectura, que los datos que posee un equipo concentrador determinado estén afectados por un retraso, en base a los cuales se tomarán probablemente decisiones de actuación sobre el proceso. De ahí que el valor de la magnitud física que posee un equipo concentrador no corresponde con su valor actual, sino con el valor que tenía cuando el equipo de instrumentación primario realizó la medición. Este retraso se denomina técnicamente "edad" del dato y es una medida indirecta de su error. La magnitud física medida está variando en el tiempo de modo que el valor actual no coincide con el valor que tenía cuando se realizó la medición (figura 1). El error será mayor o menor dependiendo de la rapidez de la variación y de la edad del dato.

En un sistema típico de instrumentación distribuido el conjunto de elementos que forman la cadena a través de la cual se obtiene una medida de una magnitud física está compuesta por (figura 2.a):

- el proceso físico estudiado (P);
- el sensor de medida (S) que realiza la conversión de la magnitud física medida a una magnitud eléctrica (tensión o intensidad);
- el equipo primario (EP) de instrumentación que acepta la señal eléctrica del sensor y la hace disponible para almacenamiento, presentación o transmisión a otros equipos;
- uno o varios concentradores (C_n, C_{n-1},...., C_1) que reciben el valor de la magnitud medida mediante comunicación de datos entre procesadores, típicamente (aunque no exclusivamente) mediante una comunicación asíncrona según la norma RS-232.

Esta arquitectura puede ser representada de forma simplificada tal como aparece en la figura 2.b.

El sistema de medida de retrasos que se propone en esta invención permite medir el tiempo que transcurre desde que un equipo de instrumentación primero (EP) lee el valor de un sensor S, hasta que dicho valor llega, mediante comunicaciones intermedias C_1...C_{n-1}, a un equipo concentrador de nivel n, C_n. Para ello el sistema de medida de retraso (SMR) debe conectarse a la instrumentación distribuida tal como aparece en la figura 3. Como puede verse, en esta configuración el equipo de instrumentación primario no lee la información del sensor sino una información que le suministra el sistema de medida en forma de señal analógica de tensión o intensidad. Por otra parte, el sistema de medida de retraso (SMR) se conecta también a la línea de comunicaciones que suministra los datos al concentrador de nivel n (C_n), pudiendo leer la información que circula por dicho canal de comunicaciones. Por tanto, el sistema de medidas se conecta con la instrumentación distribuida mediante:

- una entrada, en forma de señal digital de comunicación de datos SD, conecta internamente a un interfaz de comunicaciones;
- una salida, en forma de señal analógica de instrumentación SA, obtenida internamente a partir de un convertidor digital/analógico D/A.

La arquitectura del sistema de medida propuesto presenta una gran flexibilidad y modularidad, por lo que puede ser fácilmente adaptado a las características de los distintos sistemas de instrumentación distribuidos. Básicamente está compuesto por (figura 4):

- una unidad central de proceso (CPU) de propósito general, los requisitos de proceso de este sistema no son excesivamente pesados por lo que puede utilizarse una del tipo 80286 o similar,
- una memoria (Mm) de 1 MByte;
- uno o varios adaptadores de periféricos (A) que, en función de las prestaciones que se soliciten al sistema de medida, puede incluir típicamente disco magnético, disquete, pantalla y teclado;
- una interfaz de comunicaciones (IC) según las características de la línea de comunicación de datos de la que deba recibir información;
- un convertidor D/A cuyas características de resolución (típicamente 12 bits), señal de salida (típicamente de 0 a 10 Voltios), tolerancias, etc. se adaptarán al sistema de instrumentación distribuida al que se acopla;

- un conjunto de programas (Pr) que, utilizando los recursos físicos del equipo de medida, permiten realizar la medida del retraso operando tal como se describe más adelante, y la almacenan, procesan y presentan al operador del sistema de medida.

Explicación de las figuras

Figura 1: Error debido al retraso en función de la "edad" del dato y de la variación temporal de la magnitud física medida.

E Error de la medida debid al retraso
Ed "Edad" de la medida
M Magnitud física medida
t_1 instante de tiempo en el cual se realiza la medida
t_2 instante de tiempo en el cual se utiliza la medida realizada en t_1
t Tiempo

Figura 2: a) Esquema general de un sistema de instrumentación distribuida.
b) Representación simplificada del mismo sistema.

P Proceso físico
S Sensor
EP Equipo Primario
C_1 Concentrador 1
C_{n-1} Concentrador n-1
C_n Concentrador n, al que llega la medida realizada tras atravesar el sistema distribuido
C Concentrador al que llega la medida realizada tras atravesar el sistema distribuido, equivale en Cn en (a)

Figura 3: Arquitectura propuesta para el Sistema de Medida de Retrasos en instrumentación distribuída.

P Proceso físico
S Sensor, que aparece desconectado para inyectarle al sistema la señal analógica procedente del SMR
EP Equipo Primario
C_1 Concentrador 1
C_{n-1} Concentrador n, al que llega la medida realizada tras atravesar el sistema distribuido
SMR Sistema de Medida de Retrasos
SD Señal Digital
SA Señal Analógica

Figura 4: Esquema de una configuración interna básica del Sistema de Medida de Retrasos.

Pr Programa (software)
CPU Unidad Central de Procesos
Mm Memoria
SD Señal Digital, procedente de la línea de cominicación del sistema distribuido con el concentrador n (C_n)
SA Senal analógica, que el Sistema de Medida de Retrasos (SMR) envía al Equipo Primario (EP)
IC Interfaz de Comunicaciones
A Adaptadores de periféricos
D/A Convertidor Digital-Analógico

Figura 5: Señal analógica (SA) en forma de diente de sierra, generada por el Sistema de Medida de Retrasos (SMR).

SA Señal Analógica enviada al Equipo Primario (EP)
R Valor máximo de la señal analógica generada
T Período, en segundos, de la señal analógica generada
t Tiempo

Figura 6: Principio de operación. Determinación del retraso a partir de los valores V_1 y V_2.
a) En el caso de que $V_2 > V_1$
b) En el caso de que $V_2 < V_1$

SA Señal Analógica
R Valor máximo de la señal analógica generada
T Período, en segundos, de la señal analógica generada
V_1 Valor de la señal analógica inyectado en el Equipo Primario (EP)
V_2 Valor de la señal analógica en el momento en que el valor V1 llega, a través del sistema distribuido, al concentrador n (Cn)
$\underline{\delta}$ Diferencia entre los valores V_2 y V_1
t Tiempo

Ejemplo de realización de la invención

El sistema de medida, conectado a la instrumentación distribuida tal como se indica en la figura 3, realiza su función de acuerdo con un principio de operación que puede describirse en los siguientes pasos:

1) El sistema de medida de retrasos (SMR) genera una señal analógica (SA) que es inyectada en el equipo de instrumentación primario (EP). Esta señal tiene forma de diente de sierra (figura 5) con un valor máximo de $\underline{R}$ y un periodo de $\underline{T}$ segundos. Los valores de $\underline{R}$ y $\underline{T}$ son programables de forma que el valor máximo de la señal analógica (SA) no supene el máximo de la señal analógica (SA) no supere el máximo permitido por el equipo de instrumentación primario (EP), y que el periodo de la señal sea mayor que el máximo retraso que se desea medir.

2) El sistema de medida de retrasos (SMR) va leyendo contínuamente (por muestreo o por interrupciones) la línea de comunicaciones de entrada. Cuando se detecta que llega un valor de concentrador n, C_n, se almacena dicho valor que se denominará $\underline{V_1}$, y simultáneamente se comprueba cual es el valor $\underline{V_2}$ que en ese momento se está inyectando en el equipo de instrumentación primario (EP).

3) Dada la peculiar forma de la señal analógica (SA) inyectada, con los valores $\underline{V_1}$ y $\underline{V_2}$ puede determinarse el retraso. Pueden darse dos circunstancias, según que en el tiempo transcurrido por el retraso de transmisión, la señal analógica en diente de sierra haya evolucionado de forma continua o, por el contrario, haya alcanzado su valor máximo, haya caído a cero y de nuevo haya comenzado a subir en rampa.

a) En el primer caso (figura 6) cocurre que $\underline{V_2} > \underline{V_1}$ y el retraso, que se denominará $\underline{\delta}$, puede calcularse por la expresión

$$\delta = \frac{T}{R} (V_2\text{-}V_1)$$

b) En el segundo caso (figura 7) ocurre que $\underline{V_2} < \underline{V_1}$ y el retraso, que se denominará $\underline{\delta}$, puede calcularse por la expresión

$$\delta = \frac{T}{R} (R + V_2 - V_1)$$

4) El proceso anterior se repite varias veces permitiendo obtener para el retraso valores instantáneos, valores medios, desviaciones típicas, etc.

El sistema de medida de retrasos (SMR) utilizado ha sido implementado utilizando un ordenador con la siguiente arquitectura:

- Procesador 386-SX a 16 MHz.
- Memoria central de 1 Mbyte de capacidad.
- Disco magnético de 40 Mbytes de capacidad.
- Disquete de 3'5" y 1'4 Mbytes de capacidad.
- Monitor de 14" con resolución VGA (640x480 puntos y 256 colores).
- Teclado.
- Interfaz de comunicaciones asíncrona, según la norma RS-232-C, operando a 600 bits por segundo (bps).
- Tarjeta de salida analógica PC-LabCard modelo PCL-812 de Advantech Co., Ltd., dotada con 2 canales de salida analógicos de 12 bits, operando en un rango de 0 a 10 Voltios.

- Sistema operativo MS-DOS©versión 6.2 (MS-DOS es una marca restrada de Microsoft Corporation).

- Programa de medición codificado en lenguaje C y complicado utilizando el compilador Borland C/C++versión 3.1.

El equipo se ha utilizado en la medida de retrasos entre dos equipos de instrumentación distribuida, constituidos por un centro de control y una remota. La remota modelo Teletransa 6802, de la firma SAINCO, responde a la siguiente arquitectura:

- Unidad central de proceso basada en el microprocesador 6802, con canal de comunicaciones asíncrono, según la norma RS-232-C, operado a 600 bps.

- 2 tarjetas de adquisición de señales analógicas con resolución de 12 bits y rango de entrada de -10 a + 10 Voltios.

- Chasis y fuente de alimentación a 48 Voltios de corriente continua.

- Protocolo de comunicaciones Teletransa.

Por su parte el centro está constituido por un equipo PDP-11/84 de la firma Digital Equipment Corporation, con canal de comunicaciones asíncrono RS-232-C, operado con protocolo Teletransa.

Listados de los programas utilizados

```
/******************************** TIEMPOS.C ******************************/
#if !defined(__SMALL__)
#error Hay que compilar este fichero en el modelo SMALL para poder usar la librería
#endif


#include <stdlib.h>
#include <conio.h>
#include <stdio.h>
#include <bios.h>
#include <alloc.h>
#include <errno.h>
#include <string.h>
#include <process.h>
#include <dos.h>
#include <time.h>
#include "interrup.h"
#include "reloj.c"
#include "tarjeta.c"
#include "fun_mat.c"


/************************ PROTOTIPO DE LAS FUNCIONES ************************/


#define BYTE unsigned char
#define PUERTO 1
#define CINCO  0.1
#define PRINC  0
#define FINAL  3000
#define canal  1
```

```
void gestionador (void);
void configura (void);
void salva_tabla ( float );
void calcula_med_var ( float *, float * );


/************* Variables globales **************/


int diferencias[6][4000];  /* Tabla para guardar temporalmente las medidas */
int num = 0;               /* Contador del número de medidas tomadas */
char linea[80];
int remota1,tarjeta1,contacto1;
int remota2,tarjeta2,contacto2;


/***************************** PROGRAMA PRINCIPAL *****************************/


void main()
{
 textbackground(BLUE);
 textcolor(YELLOW);
 clrscr();
 gestionador();
 textcolor(LIGHTGRAY);
 textbackground(BLACK);
 printf("\n\n\n Se ha interrumpido la lectura de la linea.\n");
}


/******************************** GESTIONADOR *********************************/


void gestionador(void)
{
 char c;
 int i;
 float vart;   /*variable que indica el tiempo de generación
                                    de la señal analógica*/
 float duracion;

 printf("\nIntroduzca la linea de comentario: ");
 gets( linea );

 vart=0.001;
 printf("\nIntroduce duración del ciclo en segundos:");
 scanf("%f",&duracion);
 incremento=FINAL*vart/duracion;
```

```
printf("\nPara la primera medida, nº de remota  :");
scanf("%d",&remota1);
printf("                    nº de tarjeta :");
scanf("%d",&tarjeta1);
printf("                    nº de contacto:");
scanf("%d",&contacto1);
printf("\nPara la segunda medida, nº de remota  :");
scanf("%d",&remota2);
printf("                    nº de tarjeta :");
scanf("%d",&tarjeta2);
printf("                    nº de contacto:");
scanf("%d",&contacto2);

clrscr();

configura();

instala_reloj(vart);
if( inicializar_driver()!=0 )
{
        printf("\nError en la inicialización del driver\n\n");
        desinstala_reloj();
        exit(1);
}

AbreRS232();
while( ( c = bioskey(1) ) != 27 )
 {
        if ( c )
                bioskey(0);

        if(contador>FINAL) contador=PRINC;
        if( salida_analogica((int)contador,canal)!=0 )
        {
                printf("\nError en la salida analógica\n\n");
                desinstala_reloj();
                exit(1);
        }
        if(reloj()>0)
        {
                printf("\nAumenta el tiempo de creación de la señal\n\n");
                desinstala_reloj();
                exit(1);
        }
}

desinstala_reloj();
CierraRS232();
salva_tabla( vart );
printf("Fin");
```

```
}

/******************************** CONFIGURA ***********************************/
/*                    Configuración de la UART-8250                          */

void configura(void)
{
 static int velocidad=600;
 static int bitstop=1;
 static int bitdatos=8;
 static int paridad=0;
 IniRS232(velocidad,paridad,bitstop,bitdatos);
}

/************************ SALVA_TABLA ***************************************/
/*Crea un fichero donde introduce los datos leidos y la diferencia de edades*/

void salva_tabla( float vart )
{
        char nomb_fich[12];
        float media, varianza;
        float maximo,minimo;
        int i,numerovalores;

        FILE *fp;

        time_t timer;
        struct tm *tblock;

        bioskey(0);

        if( num<=0 )
        {
          puts("\nNo se ha detectado ningun mensage de medidas");
          return;
        }

        printf("\nNombre del fichero donde se quiere almacenar los datos: ");
        scanf("%s",nomb_fich);
        if( ( fp=fopen(nomb_fich,"wt") )==NULL )
        {
                puts("\n\nError en la apertura del fichero");
                fclose(fp);
        }

        fprintf(fp,"%s\n",linea);
```

```
timer=time(NULL);
tblock=localtime(&timer);
fprintf(fp,"%s\n",asctime(tblock));
fprintf(fp,"tiempo de generación: %f segundos.\n",vart);
fprintf(fp,"incremento por unidad generada: %f\n\n",incremento);

for( i=0;i<num;i++ )
{
        if( diferencias[VALOR1][i]<0 )
                diferencias[EDAD1][i]=-1;
        else
                diferencias[EDAD1][i]=diferencias[CONTADOR][i]
                                                -diferencias[VALOR1][i];
        if( diferencias[EDAD1][i] < -1 )
         {
                diferencias[EDAD1][i]= (diferencias[CONTADOR][i]+FINAL)
                                                - diferencias[VALOR1][i];
         }
        if( diferencias[VALOR2][i]<0 )
                diferencias[EDAD2][i]=-1;
        else
                diferencias[EDAD2][i]=diferencias[CONTADOR][i]
                                                -diferencias[VALOR2][i];
        if( diferencias[EDAD2][i] < -1 )
        {
                diferencias[EDAD2][i]= (diferencias[CONTADOR][i]+FINAL)
                                                - diferencias[VALOR2][i];
         }
} /* fin del for de i */

diferencias[DESFASE][0]=-FINAL;
for (i=1;i<num;i++)
{
  if (diferencias[EDAD2][i]==-1) diferencias[DESFASE][i]=-FINAL;
  else
        if (diferencias[EDAD1][i]!=-1)
          diferencias[DESFASE][i]=diferencias[VALOR2][i]-
                                                diferencias[VALOR1][i];
        else
          if (diferencias[EDAD1][i-1]==-1)
                diferencias[DESFASE][i]=-FINAL;
          else
                diferencias[DESFASE][i]=diferencias[VALOR2][i]-

diferencias[VALOR1][i-1];
} /* Fin del for */

for (i=0;i<num;i++)
{
        fprintf(fp,"%5d %5d %5d %5d %5d %5d \n",
```

```
                                diferencias[CONTADOR][i],
                                diferencias[VALOR1][i],
                                diferencias[EDAD1][i],
                                diferencias[VALOR2][i],
                                diferencias[EDAD2][i],
                                diferencias[DESFASE][i]);
        } /* fin del for */

        fprintf(fp,"\n\nValores de la Edad 1\n\n");
        maximo=0;
        minimo=FINAL;
        media=0;
        numerovalores=0;
        for (i=0;i<num;i++)
        {
         if (diferencias[EDAD1][i]!=-1)

{
                numerovalores++;
                if (diferencias[EDAD1][i]>maximo) maximo=diferencias[EDAD1][i];
                if (diferencias[EDAD1][i]<minimo) minimo=diferencias[EDAD1][i];
                media=media+diferencias[EDAD1][i];
                fprintf(fp,"%5d\n",diferencias[EDAD1][i]);
         }
        }
        media=media/numerovalores;

        varianza=0.;
        for (i=0;i<num;i++)
        {
         if (diferencias[EDAD1][i]!=-1)
                varianza=varianza+potencia ( (diferencias[EDAD1][i]-media),2);
        }
        varianza=varianza/numerovalores;

        fprintf(fp,"\nMedia de la edad 1: %f\n",media);
        fprintf(fp,"Desviación típica de la edad 1: %f\n",
                                sqrt(varianza));
        fprintf(fp,"Máximo de la edad 1: %f\n",maximo);
        fprintf(fp,"Mínimo de la edad 1: %f\n",minimo);

        fprintf(fp,"\n\nValores de la Edad 2\n\n");
        maximo=0;
        minimo=FINAL;
        media=0;
        numerovalores=0;
        for (i=0;i<num;i++)
        {
         if (diferencias[EDAD2][i]!=-1)
         {
```

```
            numerovalores++;
            if (diferencias[EDAD2][i]>maximo) maximo=diferencias[EDAD2][i];
            if (diferencias[EDAD2][i]<minimo) minimo=diferencias[EDAD2][i];
            media=media+diferencias[EDAD2][i];
            fprintf(fp,"%5d\n",diferencias[EDAD2][i]);
          }
        }
        media=media/numerovalores;

        varianza=0.;
        for (i=0;i<num;i++)
        {
          if (diferencias[EDAD2][i]!=-1)
            varianza=varianza+potencia ( (diferencias[EDAD2][i]-media),2);
        }
        varianza=varianza/numerovalores;

        fprintf(fp,"\nMedia de la edad 2: %f\n",media);
        fprintf(fp,"Desviación típica de la edad 2: %f\n",
                                    sqrt(varianza));
        fprintf(fp,"Máximo de la edad 2: %f\n",maximo);
        fprintf(fp,"Mínimo de la edad 2: %f\n",minimo);

        fprintf(fp,"\n\nValores de los desfases\n\n");
        maximo=0;
        minimo=FINAL;
        media=0;
        numerovalores=0;
        for (i=0;i<num;i++)
        {
          if (diferencias[DESFASE][i]!=-FINAL)
          {
            numerovalores++;
            if (diferencias[DESFASE][i]>maximo) maximo=diferencias[DESFASE][i];
            if (diferencias[DESFASE][i]<minimo) minimo=diferencias[DESFASE][i];
            media=media+diferencias[DESFASE][i];
            fprintf(fp,"%5d\n",diferencias[DESFASE][i]);
          }
        }
        media=media/numerovalores;

        varianza=0.;
        for (i=0;i<num;i++)
        {
          if (diferencias[DESFASE][i]!=-FINAL)
            varianza=varianza+potencia ( (diferencias[DESFASE][i]-media),2);
        }
        varianza=varianza/numerovalores;

        fprintf(fp,"\nMedia del desfase: %f\n",media);
```

```
            fprintf(fp,"Desviación típica del desfase: %f\n",
                                    sqrt(varianza));
            fprintf(fp,"Máximo del desfase: %f\n",maximo);
            fprintf(fp,"Mínimo del desfase: %f\n",minimo);

            fprintf(fp,"\n\nValores de los desfases absolutos\n\n");
            maximo=0;
            minimo=FINAL;
            media=0;
            numerovalores=0;
            for (i=0;i<num;i++)
            {
              if (diferencias[DESFASE][i]!=-FINAL)
              {
                    numerovalores++;
                    if (abs(diferencias[DESFASE][i])>maximo)
                      maximo=abs(diferencias[DESFASE][i]);
                    if (abs(diferencias[DESFASE][i])<minimo)
                      minimo=abs(diferencias[DESFASE][i]);
                    media=media+abs(diferencias[DESFASE][i]);
                    fprintf(fp,"%5d\n",abs(diferencias[DESFASE][i]));
              }
            }
            media=media/numerovalores;

            varianza=0.;
            for (i=0;i<num;i++)
            {
              if (diferencias[DESFASE][i]!=-FINAL)
                    varianza=varianza+potencia ( (abs(diferencias[DESFASE][i])-media),2);
            }
            varianza=varianza/numerovalores;

            fprintf(fp,"\nMedia del desfase absoluto: %f\n",media);
            fprintf(fp,"Desviación típica del desfase absoluto: %f\n",
                                    sqrt(varianza));
            fprintf(fp,"Máximo del desfase absoluto: %f\n",maximo);
            fprintf(fp,"Mínimo del desfase absoluto: %f\n",minimo);

            fclose( fp );

    } /* Fin de salva tabla */


    /************************** INTERRUP.C ***********************************/

    #if !defined(__SMALL__)
    #error Hay que compilar este fichero en el modelo SMALL para poder usar la librería
    #endif
```

```
#include <bios.h>
#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include <mem.h>
#include <string.h>
#include <process.h>
#include <alloc.h>
#include <stdlib.h>
#include "interrup.h"
#include "defines.mac"
#include "fun_glob.h"

/************************ VARIABLES GLOBALES *******************************/

BYTE  AntImr,Reg3_RS232,Reg4_RS232; /* VARIABLES GLOBALES DEL SISTEMA */
void interrupt (*oldfunc)(void);

void static near ini_comunicaciones(int divisor,BYTE formato_dato );

char *men_error[] = {
                                " Velocidad no permitida...\n     Rango valido: [110, 150,
300, 600, 1200, 2400, 4800, 9600] baudios \n",
                                " Paridad no permitida...\n  Rango valido: [0:NONE, 1:PAR,
2:IMPAR]\n",
                                " Bit de stop no permitido...\n    Rango valido: [0:1, 1:2 (si
8,7,6 bit por caracter) ¦ 1,5 (si 5 bit por caracter)] bit de stop\n",
                                " Longitud caracter no permitido...\n      Rango valido:
[5,6,7,8] bit por caracter\n"
                              };

extern int diferencias[3][4000];  /* definida en tiempos,c */
extern int num;                   /* definido en tiempos.c */
extern float contador;            /* definido en reloj.c */
extern int remota1,tarjeta1,contacto1;
extern int remota2,tarjeta2,contacto2;
int estado_mensaje;
BYTE longitud,remota,tipo_mensaje,tarjeta,contacto;
BYTE ncaracteres,mensaje[100];

/************************** RECEPCION () ***********************************/
/*          Rutina de interrupcion del puerto serie                        */

void interrupt recepcion(void)
{
BYTE ch;
int valor1,valor2;
```

```
enable();

if(HA_LLEGADO_CARACTER)
 {
        ch=recibir_caracter();
        if( (trata_linea(ch,&valor1,&valor2)) )
        {
                if (contador<=MINIMOCONTADOR) valor1=valor2=-1;
                disable();
                diferencias[CONTADOR][num]=contador;
                enable();
                diferencias[VALOR1][num]=(float)(valor1-OFFSETCALIBRACION)/
                                                        FACTORCALIBRACION;
                diferencias[VALOR2][num]=(float)(valor2-OFFSETCALIBRACION)/
                                                        FACTORCALIBRACION;
                num++;
        }
 }
else
        if(HAY_ERROR_SERIALIZACION)
         {
          inportb(BASE_COM+5);  /*BORRAMOS EL ERROR DE SERIALIZACION*/
         }

outport(0x20,0x20); /* FIN DE SERVICIO DE INTERRUPCION*/
}


/***************************** INIRS232 () ***********************************/
/*          Rutina de inicializacion del puerto serie                    */


int IniRS232(int velocidad,int paridad,int bitstop,int bitdatos)
{
 unsigned long int divisor;

divisor = 115200L/velocidad;

switch(paridad)
 {
  case 0: /* NO EXISTE PARIDAD */

                paridad = 0x00;
                break;

  case 1: /* PARIDAD IMPAR */

                paridad = 0x08;
                break;
```

```
        case 2: /* PARIDAD PAR */
                        paridad = 0x18;
                        break;

        default: /* ERROR */
                       printf("/**********\n %s **********/\n",men_error[1]);
                       return 0;
       }

       switch(bitstop)
       {

        case 0: /* UN BIT DE STOP */

                        bitstop = 0x00;
                        break;

        case 1: /* DOS BIT DE STOP */

                        bitstop = 0x04;
                        break;


        default: /* ERROR */
                       printf("/**********\n %s **********/\n",men_error[2]);
                       return 0;
       }

      switch(bitdatos)
       {
        case 5: /* CINCO BITS DE DATOS */

                        bitdatos = 0x00;
                        break;

        case 6: /* SEIS BIT DE DATOS */

                        bitdatos = 0x01;
                        break;

        case 7: /* SIETE BITS DE DATOS */

                        bitdatos = 0x02;
                        break;

        case 8: /* 8 BITS DE DATOS */

                        bitdatos = 0x03;
                        break;
```

```
    default: /* ERROR */
                    printf("/**********\n %s **********/\n",men_error[3]);
                    return 0;
    }

 ini_comunicaciones(divisor,(paridad|bitstop|bitdatos));
 return 1;
}


/********************** INI_COMUNICACIONES () ******************************/

/*          Rutina de inicializacion de las comunicaciones                */

void static near ini_comunicaciones(int divisor,BYTE formato_dato)
{
 outportb(BASE_COM+3,0x80); /* PARA ACCEDER AL DIVISOR DEL LATCH DE VELOCIDAD
*/
 outportb(BASE_COM,divisor); /* BYTE MENOS SIGNIFICATIVO DE LA VELOCIDAD */
 outportb(BASE_COM+1,divisor >> 8); /* BYTE MAS SIGNIFICATIVO DE LA VELOCIDAD
*/
 outportb(BASE_COM+3,formato_dato);/* INICIALIZA EL PUERTO SERIE A FORMATO DATO
*/
}


/********************** ENVIAR_CARACTER () *********************************/
/*          Rutina para enviar un caracter por el puerto serie            */
/*          Imprime en pantalla el caracter que va a transmitir           */

void enviar_caracter(BYTE ch/*,int *i,int *j*/)   /* TRANSMITE UN CARACTER TRAMA */
{
 /*char buffer[2];*/
 while(NO_SE_PUEDA_TRANSMITIR)
        ;
 TRANSMITIR(ch);  /* SE TRANSMITE EL CARACTER DE COMIENZO*/

 cprintf("%x ",ch);

}


/********************** RECIBIR_CARACTER () ********************************/
/*          Rutina para recibir un caracter por el puerto serie           */
/*           Imprime en pantalla el caracter que ha recibido              */
```

```
char recibir_caracter(void)
{
 BYTE ch;

 ch = RECIBIR;

 if( ch==0xb0 )
 {
        textcolor(RED);
        cprintf("*REMOTA*");
        textcolor(YELLOW);
 }

 cprintf("%x ",ch);
 return(ch);

}


/************************** INI_INTERRUPCIONES ******************************/


void AbreRS232(void)
{

 /* GUARDAR REGISTROS DE 8259 Y 8250 Y VECTORES INTERRUPCION */

 Antlmr  = inportb(BASE_8259+1);
 oldfunc = getvect(VECTOR_COM);

 Reg3_RS232 = inportb(BASE_COM+3);
 Reg4_RS232 = inportb(BASE_COM+4);

 outportb(BASE_8259+1,Antlmr | (0xFF ^ MASCARA_COM)); /* DESACTIVAR IRQ DEL
PUETO SERIE*/

                                                          /* DE DATO
*/
 inportb(BASE_COM);          /* ELIMINAR INT. DE DATO RECIBIDO */
 inportb(BASE_COM+2);         /* ELIMINAR INT. DE DATO TRANSMITIDO */
 inportb(BASE_COM+5);         /* ELIMINAR INT. DE ERROR DE SERIALIZACION */
 inportb(BASE_COM+6);         /* ELIMINAR INT. DE ENTRADA DE RS232  */

 outportb(BASE_COM+1,0x05);    /* ACTIVAR INT. DE RECEPCION, SERIALIZACION */
 outportb(BASE_COM+4,0x08);    /* ACTIVAR INT.GENERAL DE LA PLACA RS232  */

 inportb(BASE_COM);          /* ELIMINAR INT. DE DATO RECIBIDO */
 inportb(BASE_COM+2);         /* ELIMINAR INT. DE DATO TRANSMITIDO */
 inportb(BASE_COM+5);         /* ELIMINAR INT. DE ERROR DE SERIALIZACION */
 inportb(BASE_COM+6);         /* ELIMINAR INT. DE ENTRADA DE RS232  */
```

```
 setvect(VECTOR_COM,recepcion);

 outportb(BASE_8259+1,Antlmr & MASCARA_COM); /* ACTIVAR IRQ DEL PUERTO SERIE*/
}


/***************************** CIERRARS232() ********************************/
/*              Rutina de cierre del puerto serie                       */


void CierraRS232(void)
{
 outportb(BASE_8259+1,Antlmr);     /* DESACTIVAR IRQ DEL PUETO SERIE*/
 outportb(BASE_COM+3,Reg3_RS232);  /* RESTAURAR REGISTRO 3 */
 outportb(BASE_COM+4,Reg4_RS232);  /* RESTAURAR REGISTRO 4 */
 outportb(BASE_COM+1,0x00); /* DESABILITAMOS LAS INTERRUPCIONES */
 outportb(BASE_COM+4,0x00); /* DESABILITAMOS LA INT GENERAL DE LA PLACA */
 setvect(VECTOR_COM,oldfunc);      /* RESTAURAR VECTOR DE INTERRUPCIONES */
}


/***************************** TRATA_LINEA **********************************/
/* Comprueba si el mensaje recibido corresponde a una señal de medidas o nó*/


int trata_linea(BYTE ch,int *valor1,int *valor2)
{
        switch(estado_mensaje)
        {
        case 0: /* No ha llegado cabecera */
          if (ch==0xB0)
          {
                estado_mensaje=1;
                ncaracteres=0;
                mensaje[ncaracteres++]=ch;
          }
    return(0);

        case 1: /* Ha llegado cabecera; debe llegar longitud */
          longitud=ch;
          estado_mensaje=2;
          mensaje[ncaracteres++]=ch;
          return(0);

        case 2: /* Ha llegado longitud; debe llegar remota */
          remota=ch;
          estado_mensaje=3;
          mensaje[ncaracteres++]=ch;
          return(0);
```

```
            case 3: /* Ha llegado remota; debe llegar tipo mensaje */
              tipo_mensaje=ch&0x0F;
              tarjeta=(ch>>4)&0x0F;
              estado_mensaje=4;
              mensaje[ncaracteres++]=ch;
              return(0);

            case 4: /* Están llegando los datos del mensaje */
              if (ncaracteres>=longitud-3) estado_mensaje=5;
              mensaje[ncaracteres++]=ch;
              return(0);

            case 5: /* Está llegando primer byte de CRC */
              estado_mensaje=6;
              mensaje[ncaracteres++]=ch;
              return(0);

            case 6: /* Está llegando segundo byte de CRC */
              estado_mensaje=0;
              mensaje[ncaracteres++]=ch;
              break;

        } /* Fin del switch de estado_mensaje */

      /*  if(!CRC(mensaje,CRC_16,COMPROBAR)) return(0); */

        *valor1=*valor2=-1;
        if ((remota==remota1)&&(tarjeta==tarjeta1))
              *valor1=((mensaje[2*contacto1+3]<<8)&0xFF00)+mensaje[2*contacto1+2];
        if ((remota==remota2)&&(tarjeta==tarjeta2))
              *valor2=((mensaje[2*contacto2+3]<<8)&0xFF00)+mensaje[2*contacto2+2];

        return(1);
      }


      /*************************************************************/
      /**                    RELOJ.C                              **/
      /*************************************************************/


      /*************************************************************/
      /**               FICHEROS INCLUDE                          **/
      /*************************************************************/

      #include <dos.h>
      #include <conio.h>
      #include <process.h>
      #include "reloj.h"
```

```
/*************************************************************/
/**                VARIABLES GLOBALES                       **/
/*************************************************************/

/* puntero a una función de interrupción para guardar el vector de
   interrupción de la rutina de reloj original del DOS */
void interrupt (* reloj_del_DOS) ();

/* Guarda el número de tics que debe esperar la función reloj() desde
   la última vez que se llamó */
static int fin_cont_reloj ;

/* Contador del reloj. En cada tic se incrementa en uno */
static int cont_reloj = 0;

/* Contador del número de pulsos que voy enviando a la remota; al llegar a
   4095 se vuelve a inicializar a 0 */
float contador = 0;

/* Variable que nos indica el incremento que debe efectuar contador cada vez
   que se envie una señal a la remota*/
float incremento = 1.;

/*************************************************************/
/**                    FUNCIONES                            **/
/*************************************************************/


/*****************    instala_reloj()    ********************/

/* Actualiza el vector de interrupción de la rutina de reloj del DOS para
   que apunte a mi_reloj(). Previamente el vector de interrupción original
   es salvado. Además cambia la frecuencia de la interrupción para ponerla
   en 1000 Hz e inicializa la variable fin_cont_reloj */

void instala_reloj ( float tmuest )
{
  /* El mínimo período de muestreo permitido es 0.001 segundos,
        equivalente a esperar 1 tic en cada muestra */
  if ( tmuest < 0.001 )
       {
         cputs("\n\nError en instalación del reloj.\r\n");
         cputs("Período de muestreo muy bajo [mínimo 0.001 seg].\r\n");
         exit(1);
       }

  /* deshabilita interrupciones */

  /* salva el vector de interrupción de la rutina de reloj del DOS */
  reloj_del_DOS = getvect ( INT_RELOJ );
```

```
    /* avisa al TIMER de que se va modificar el valor del contador 0 */
    outportb ( PUERTO_CONTROL_TIMER , 62 );

    /* el nuevo divisor de frecuencia se calcula:
          frecuencia de trabajo del TIMER = 1193180 Hz
          frecuencia de tics de reloj     = 1000 Hz (especificada en reloj.h)
          divisor = 1193180/1000 = 1193 = 0x04A9 */

    /* byte bajo del nuevo divisor de frecuencia */
    outportb ( PUERTO_CONT0_TIMER , 0xA9 );

    /* byte alto del nuevo divisor de frecuencia */
    outportb ( PUERTO_CONT0_TIMER , 0x04 );

    /* actualiza el vector de interrupción de la rutina de reloj para que
          apunte a la rutina  mi_reloj() */
    setvect ( INT_RELOJ , mi_reloj );

    /* habilita interrupciones */

    /* establece el número de tics que debe esperar la función reloj() */
    fin_cont_reloj = (tmuest*FREC_RELOJ);

   }

  /******************    mi_reloj()    ***********************/

  /* Actualiza el contador de reloj en cada tic */

  void interrupt mi_reloj()
   {
     /* incrementa el contador en 1 */
     cont_reloj++;
     contador=contador+incremento;

     /* interrupción servida */
     outportb ( PUERTO_CONTROL_INT , 0x20 );

   }


  /*****************    desinstala_reloj()    *****************/

  /* Restaura la rutina de reloj original del DOS y la frecuencia de
     los tics a 18.2 Hz */

  void desinstala_reloj ()
   {
     /* deshabilita interrupciones */
```

```
    disable();

    /* avisa al TIMER de que se va modificar el valor del contador 0 */
    outportb ( PUERTO_CONTROL_TIMER , 62 );

    /* el nuevo divisor de frecuencia se calcula:
            frecuencia de trabajo del TIMER = 1193180 Hz
            frecuencia de tics de reloj     = 18.2 Hz
            divisor = 1193180/18.2 = 65535 = 0xFFFF */

    /* byte bajo del nuevo divisor de frecuencia */
    outportb ( PUERTO_CONT0_TIMER , 0xFF );

    /* byte alto del nuevo divisor de frecuencia */
    outportb ( PUERTO_CONT0_TIMER , 0xFF );

    /* actualiza el vector de interrupción de la rutina de reloj para que
            apunte a la rutina  original del DOS */
    setvect ( INT_RELOJ , reloj_del_DOS );

    /* habilita interrupciones */
    enable();

  }


/*********************     reloj()    ***********************/

/* Espera fin_cont_reloj tics desde la última vez que se llamó o
   desde que se ejecutó instala_reloj(). Devuelve un entero que si
   es <= 0 significa que no se ha pasado del tiempo, el contador
   del reloj no se ha pasado de su límite fin_cont_reloj. El valor
   de éste entero en valor absoluto son los tics que le han sobrado.
   Si es positivo indica el número de tics que han pasado de
   fin_cont_reloj */

int reloj()
 {
  /* valor del contador al empezar la rutina y valor temporal para
          controlar su llegada a fin_cont_reloj */
  int ini_cont , cont_aux;

  /* deshabilita interrupciones */
  disable();

  /* valor de cont_reloj al comienzo de la rutina */
  ini_cont = cont_reloj;

  /* habilita interrupciones */
  enable();
```

```
        /* leer cont_reloj hasta que no llegue a fin_cont_reloj */
        do{
                /* deshabilita interrupciones */
                disable();

                /* guarda cont_reloj en variable auxiliar */
                cont_aux = cont_reloj;

                /* habilita interrupciones */
                enable();

              } while ( cont_aux < fin_cont_reloj );

        /* deshabilita interrupciones */
        disable();

        /* resetea el contador de reloj */
        cont_reloj = 0;

        /* habilita interrupciones */
        enable();

        /* devuelve el número de tics que han sobrepasado a fin_cont_reloj,
                si es negativo es que le sobró tiempo */
        return (ini_cont - fin_cont_reloj);

 }

/*****************************************************************/
/**                   TARJETA.C                                 **/
/*****************************************************************/


/*****************************************************************/
/**                   FICHEROS INCLUDE                          **/
/*****************************************************************/

#include <tarjeta.h>

/*****************************************************************/
/**                   FUNCIONES                                 **/
/*****************************************************************/


/*****************    inicializar_driver()    ****************/

/* Inicializa el controlador de la tarjeta. Si no hubo errores
   devuelve un 0 y otro valor en caso contrario */

int inicializar_driver()
```

```
 {
  /* datos de entrada al driver (ary1 y ary2 no se usan) */
  int dat[5] , ary1[1] , ary2[1];

  /* dirección base donde está instalada la tarjeta */
  dat[0] = DIR_BASE;
  /* nivel de IRQ (no usado) */
  dat[1] = 2;
  /* nivel de DMA (no usado) */
  dat[2] = 1;

  /* llama al driver para inicializarlo devolviendo un posible
        código de error */
  return(pcl812(INIC_DRIVER,dat,ary1,ary2));

 }


/*****************   entrada_analogica()   ***************/

/* Recoge una entrada analógica por el canal especificado
   con la función selecciona_canal(). El rango de conversión
   es de -2048 a 2047. Si hubo errores devuelve un 1 y en
   caso contrario un 0 y el valor convertido */

int entrada_analogica ( int *ent )
{
  /* datos de entrada al driver (ary1 y ary2 no se usan) */
  int dat[5],ary1[1],ary2[1];

  /* realiza una conversión A/D (-2048 a 2047) devolviendo
        un 1 en caso de error */
  if (pcl812(UNA_CONVERSION_AD,dat,ary1,ary2))
        return 1;
  else
       {
        *ent = dat[0];
        return 0;
       }

}


/******************   salida_analogica()   ***************/

/* Envía un dato digital a la tarjeta sacando ésta uno analógico
   por el canal D/A indicado en el parámetro canal. En caso de error
   devuelve un valor distinto de 0 y un 0 en caso contrario */

int salida_analogica ( int valor , int canal )
```

```
{
 /* datos para la tarjeta (ary1 y ary2 no se usan) */
 int dat[5],ary1[1],ary2[1];

 /* salida analógica por canal 1 ó 2 */
 dat[0] = canal;
 /* valor digital que saca (0-4095) */
 dat[1] = valor;

 /* da salida al dato devolviendo un código de error */
 return(pcl812(UNA_CONVERSION_DA,dat,ary1,ary2));

}


/******************   selecciona_canal()   ***************/

/* Selecciona uno de los canales A/D de la tarjeta. Hay disponibles
   16 canales (0-15). Si hubo errores devuelve un valor distinto de
   0 y un 0 si la operación tuvo éxito */

int selecciona_canal ( int c )
 {
  /* datos para la tarjeta (ary1 y ary2 no se usan) */
  int dat[5] , ary1[1] , ary2[1];

  /* especifica números de canales inicial y final para las
        conversiones A/D (en este caso el inicial y el final
        son el mismo) */
  dat[0] = dat[1] = c;

  /* selecciona el canal elegido devolviendo un código de
        error */
  return(pcl812(SELEC_CANAL,dat,ary1,ary2));

 }


/******************   convertir_salida()   ****************/

/* Convierte el rango de entrada [0,4095] en otro
   [0,MARGEN_INF,2048,MARGEN_SUP,4095] según la siguiente regla:

   0,2047    --> 0,MARGEN_INF
   2049,4095 --> MARGEN_SUP,4095
   2048      --> 2048

   Esto es debido a que  el motor no se mueve en el rango
   MARGEN_INF,2048,MARGEN_SUP. En caso de error devuelve un 1 y
   un 0 en caso contrario */
```

```
int convertir_salida ( int *sal , float val )
 {
  /* en principio no hay errores */
  int err=0;

  /* implementa la regla anterior */
  if ( val == 2048.0 )

        *sal = 2048;

  else if ( (val>2048.0) && (val<4096.0) )

        *sal = MARGEN_SUP + ((val-2048)*(4095-MARGEN_SUP)/2047);

  else if ( (val<2048.0) && (val>-1.0) )

        *sal = val * MARGEN_INF / 2047;

  else

        err = 1;

  /* devuelve un código de error */
  return err;

 }

/******************************* FUN_MAT.C ******************************/

#if !defined(__SMALL__)
#error Hay que compilar este fichero en el modelo SMALL para poder usar la librería
#endif

#include <stdio.h>
#include <math.h>
#include "fun_mat.h"


float potencia ( float valor, int exp )
{
        float val_aux;
        int a;

        val_aux=valor;

        if( exp==NULO )
                        val_aux=1;
        else
        {
                for( a=1;a<exp;a++ )
```

```
                {
                        val_aux = val_aux*valor;
                }
        }

        return( val_aux );
}

/****************************** INTERRUP.H ******************************/
#define COM1 0
#define COM2 1
#define BASE_COM1    0x3F8
#define BASE_COM2    0x2F8
#define BASE_8259    0x20
#define VECTOR_COM1  0x0C
#define VECTOR_COM2  0x0B
#define MASCARA_COM1 0xEF
#define MASCARA_COM2 0xF7
#define PUERTO_ES_UNO   /* SE DEBE DEFINIR SI EL PUERTO A USAR ES UNO,SINO */
#define FALSE 0         /* SE DEBE DEFINIR EL PUERTO_ES_DOS */
#define TRUE 1
#define CONTADOR 0
#define VALOR1 1
#define EDAD1 2
#define VALOR2 3
#define EDAD2 4
#define DESFASE 5
#define OFFSETCALIBRACION   18
#define FACTORCALIBRACION   0.561
#define MINIMOCONTADOR      1000

#if defined( PUERTO_ES_UNO )

#define BASE_COM    BASE_COM1
#define VECTOR_COM  VECTOR_COM1
#define MASCARA_COM MASCARA_COM1
#define COM         COM1

#elif defined( PUERTO_ES_DOS )

#define BASE_COM    BASE_COM2
#define VECTOR_COM  VECTOR_COM2
#define MASCARA_COM MASCARA_COM2
#define COM         COM2

#else
#error Hay que definir que puerto se utiliza
#endif
```

```
#define TX_BUFFER_VACIO          0x20
#define TX_ESTADO                (inportb(BASE_COM + 5) & TX_BUFFER_VACIO)
#define NO_SE_PUEDA_TRANSMITIR   TX_ESTADO == 0
#define TRANSMITIR(X)            outportb(BASE_COM,X)
#define RECIBIR                  inportb(BASE_COM)
#define RX_BUFFER_LLENO          0x01
#define RX_ESTADO                (inportb(BASE_COM + 5) & RX_BUFFER_LLENO)
#define NO_SE_PUEDA_RECIBIR      RX_ESTADO == 0
#define BYTE   unsigned char
#define IIR_ESTADO               (inportb(BASE_COM+2) & 0x07)
#define HA_LLEGADO_CARACTER      IIR_ESTADO == 0x04
#define HAY_ERROR_SERIALIZACION  IIR_ESTADO == 0x06


int  IniRS232(int velocidad,int paridad,int bitstop,int bitdatos);
void enviar_caracter(BYTE ch/*,int *i,int *j*/);
char recibir_caracter(void);
void CierraRS232(void);
void AbreRS232(void);
int  trata_linea(BYTE ch,int *valor1,int *valor2);

/*************************************************************/
/**                    RELOJ.H                              **/
/*************************************************************/

#ifndef RELOJ_H
#define RELOJ_H

/*************************************************************/
/**                    DEFINES                              **/
/*************************************************************/

/* Puerto de control del TIMER */
#define  PUERTO_CONTROL_TIMER  0x43

/* Puerto del contador 0 del TIMER */
#define  PUERTO_CONT0_TIMER    0x40

/* Puerto del controlador de interrupciones */
#define  PUERTO_CONTROL_INT    0x20

/* Interrupción del reloj */
#define  INT_RELOJ             0x08

/* Nueva frecuencia del reloj. Este valor no debe ser modificado */
#define  FREC_RELOJ            1000
```

```
/*************************************************************/
/**                    PROTOTIPOS                           **/
/*************************************************************/


/******************   instala_reloj()   *******************/

/* Actualiza el vector de interrupción de la rutina de reloj del DOS para
   que apunte a mi_reloj(). Previamente el vector de interrupción original
   es salvado. Además cambia la frecuencia de la interrupción para ponerla
   en 1000 Hz e inicializa la variable fin_cont_reloj */

void instala_reloj ( float );


/******************   mi_reloj()   ************************/

/* Actualiza el contador de reloj en cada tic */

void interrupt mi_reloj ( void );


/*****************   desinstala_reloj()   *****************/

/* Restaura la rutina de reloj original del DOS y la frecuencia de
   los tics a 18.2 Hz */

void desinstala_reloj ( void );


/*********************   reloj()   ************************/

/* Espera fin_cont_reloj tics desde la última vez que se llamó o
   desde que se ejecutó instala_reloj(). Devuelve un entero que si
   es <= 0 significa que no se ha pasado del tiempo, el contador
   del reloj no se ha pasado de su límite fin_cont_reloj. El valor
   de éste entero en valor absoluto son los tics que le han sobrado.
   Si es positivo indica el número de tics que han pasado de
   fin_cont_reloj */

int reloj ( void );


#endif

/*************************************************************/
/**                    TARJETA.H                            **/
/*************************************************************/

#ifndef TARJETA_H
```

```
#define TARJETA_H

/*************************************************************/
/**                    DEFINES                              **/
/*************************************************************/

/* Dirección base de la tarjeta de conversión PCL-812 */
#define  DIR_BASE              0x220

/* Número de función del controlador de la tarjeta para
   inicializar éste */
#define  INIC_DRIVER           0

/* Número de función para seleccionar canal de entrada analógico */
#define  SELEC_CANAL           1

/* Número de función para realizar una conversión A/D por el
   canal indicado en la función selecciona_canal() */
#define  UNA_CONVERSION_AD     3

/* Número de función para realizar una conversión D/A */
#define  UNA_CONVERSION_DA     15

/* Valor entero por encima de 2048 en el que todavía no se
   mueve el motor */
#define  MARGEN_SUP            2900

/* Valor entero por debajo de 2048 en el que todavía no se
   mueve el motor */
#define  MARGEN_INF            1200


/*************************************************************/
/**                    PROTOTIPOS                           **/
/*************************************************************/

/*****************    pcl812()    ***************************/

/* Driver de la tarjeta, contiene las funciones de conversión */

extern pcl812 ( int , int* , int* , int* );


/*****************    inicializar_driver()    ***************/

/* Inicializa el controlador de la tarjeta. Si no hubo errores
   devuelve un 0 y otro valor en caso contrario */
```

```
int inicializar_driver ( void );

/*****************    entrada_analogica()    ***************/

/* Recoge una entrada analógica por el canal especificado
   con la función selecciona_canal(). El rango de conversión
   es de -2048 a 2047. Si hubo errores devuelve un 1 y en
   caso contrario un 0 y el valor convertido */

int entrada_analogica ( int * );


/******************    salida_analogica()    ***************/

/* Envía un dato digital a la tarjeta sacando ésta uno analógico
   por el canal D/A indicado en el parámetro canal. En caso de error
   devuelve un valor distinto de 0 y un 0 en caso contrario */

int salida_analogica ( int , int );


/******************    selecciona_canal()    ***************/

/* Selecciona uno de los canales A/D de la tarjeta. Hay disponibles
   16 canales (0-15). Si hubo errores devuelve un valor distinto de
   0 y un 0 si la operación tuvo éxito */

int selecciona_canal ( int );


/*****************    convertir_salida()    ***************/

/* Convierte el rango de entrada [0,4095] en otro
   [0,MARGEN_INF,2048,MARGEN_SUP,4095] según la siguiente regla:

   0,2047    --> 0,MARGEN_INF
   2049,4095 --> MARGEN_SUP,4095
   2048      --> 2048

   Esto es debido a que  el motor no se mueve en el rango
   MARGEN_INF,2048,MARGEN_SUP. En caso de error devuelve un 1 y
   un 0 en caso contrario */

int convertir_salida ( int * , float );


#endif

/******************************** FUN_MAT.H ********************************/
```

```
#define NULO 0

float potencia ( float, int );
```

REIVINDICACIONES

1. Sistema para la medida de retrasos en instrumentación distribuida, **caracterizado** porque utiliza una unidad de procesos de datos, gobernada por un software específico, que permite medir el tiempo que transcurre desde que un equipo de instrumentación primario lee el valor de un sensor, hasta que dicho valor llega, mediante comunicaciones intermedias, a un equipo concentrador final. Para ello el sistema de medida está conectado formando un bucle, cuya entrada es una señal digital de comunicación de datos, conectada internamente a un interfaz de comunicaciones, y cuya salida es una señal analógica de instrumentación en diente de sierra, obtenida internamente a partir un convertidor ditital/analógico, que se inyecta en el equipo de instrumentación primario.

2. Sistema para la medida de retrasos en instrumentación distribuida, según reivindicación 1, **caracterizado** porque la unidad de proceso es de propósito general y consta de una memoria, uno o varios adaptadores de periféricos en función de las prestaciones requeridas, una interfaz de comunicaciones adaptada a las características de la línea de comunicación de datos, un convertidor digital/analógico cuyas características de resolución, señal de salida y tolerancia se adaptarán al sistema de instrumentación al que se acopla, y un conjunto de programas de control y operación.

3. Sistema para la medida de retrasos en instrumentación distribuida, según reivindicaciones 1 y 2, basado en: a) la inyección de una señal analógica en forma de diente de sierra en el equipo de instrumentación primario; y b) en la lectura de la información retrasada de la línea digital de comunicaciones.

4. Sistema para la medida de retrasos en instrumentación distribuida, según reivindicaciones 1 a 3, **caracterizado** porque la señal analógica que se inyecta en el equipo de instrumentación primario es de parámetros definibles, ajustándose de tal forma que el valor máximo de la señal no supere el máximo permitido por el equipo de instrumentación primario, y cuyo periodo sea mayor que el máximo retraso que se desea medir.

5. Sistema para la medida de retrasos en instrumentación distribuida, según reivindicaciones 1 a 4, **caracterizado** porque el retraso se obtiene como diferencia ponderada de los valores que, en el mismo instante de tiempo, poseen la señal analógica de entrada al equipo de instrumentación primario y la señal digital que llega al concentrador final por la línea de comunicaciones.

6. Sistema para la medida de retrasos de instrumentación distribuida, según reivindicaciones 1 a 5, constituido por una arquitectura modular y flexible que permite adaptarse a los distintos sistemas de instrumentación y/o control distribuidos.

7. Sistema para la medida de retasos en instrumentación distribuida, según reivindicaciones 1 a 6, de aplicación en el telecontrol de redes de energía eléctrica, y en la gestión de las redes de telecomunicación.

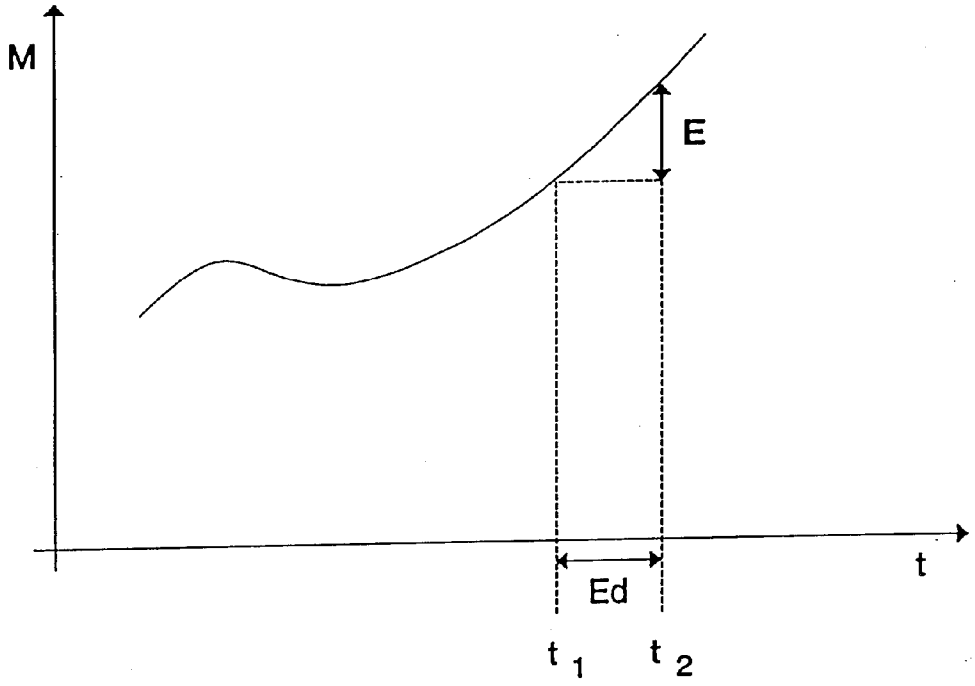


Figura 1

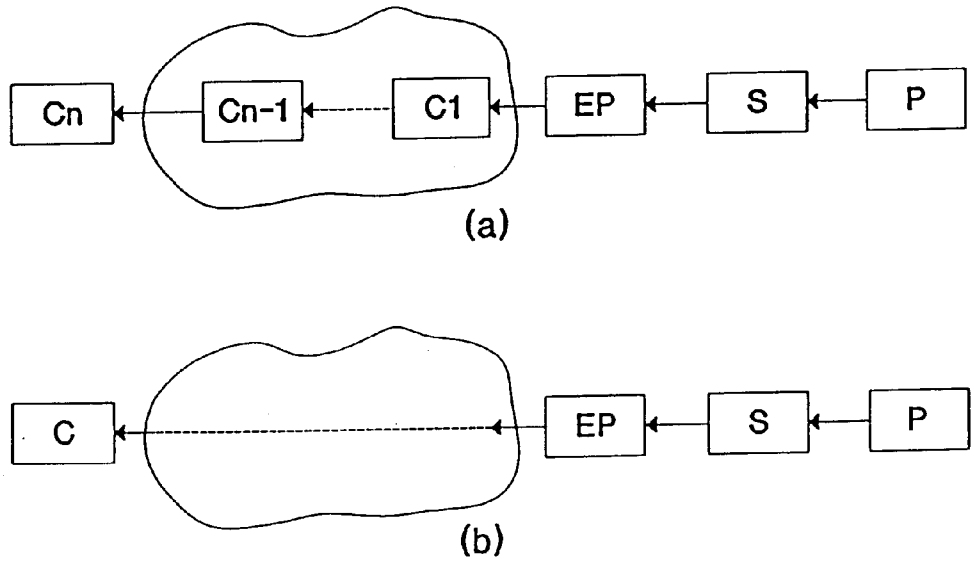


Figura 2

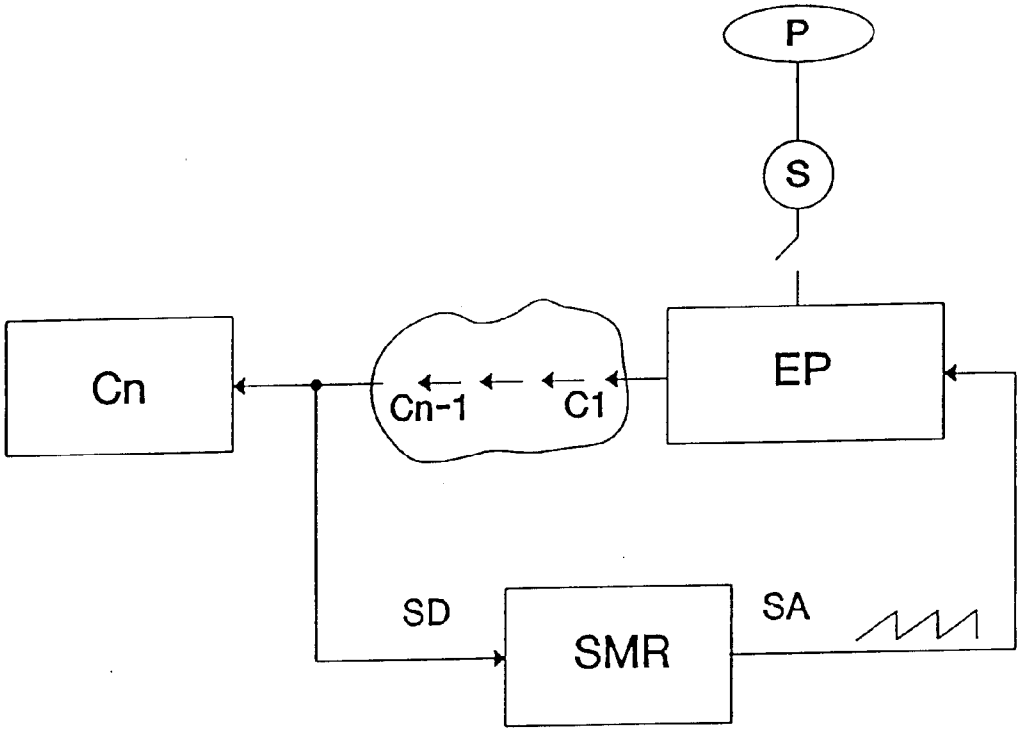


Figura 3

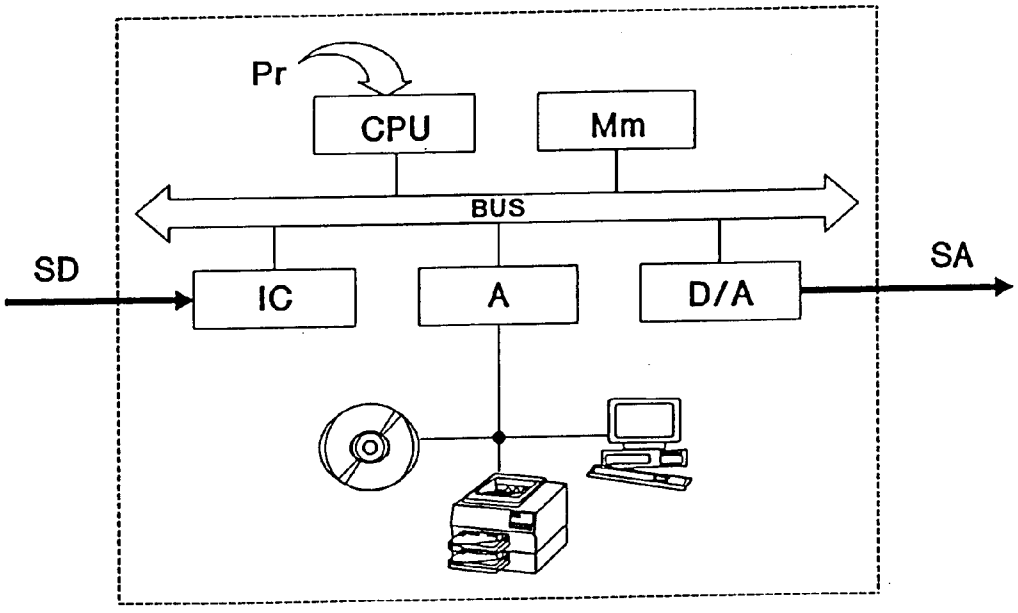


Figura 4

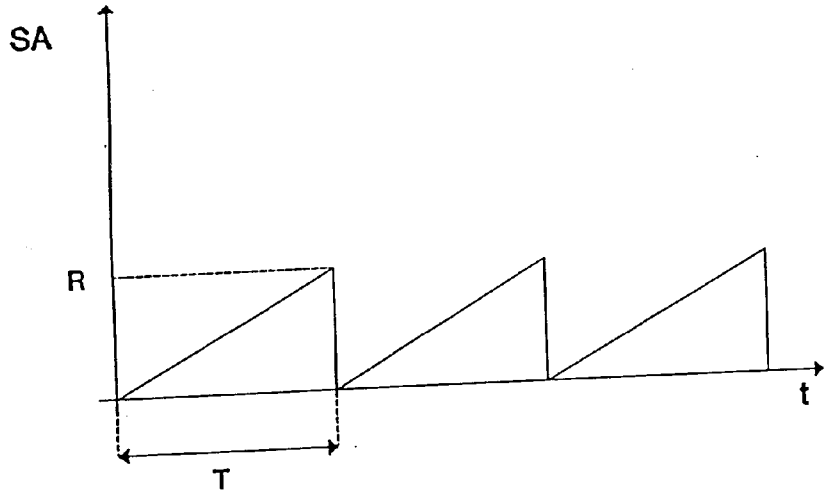


Figura 5

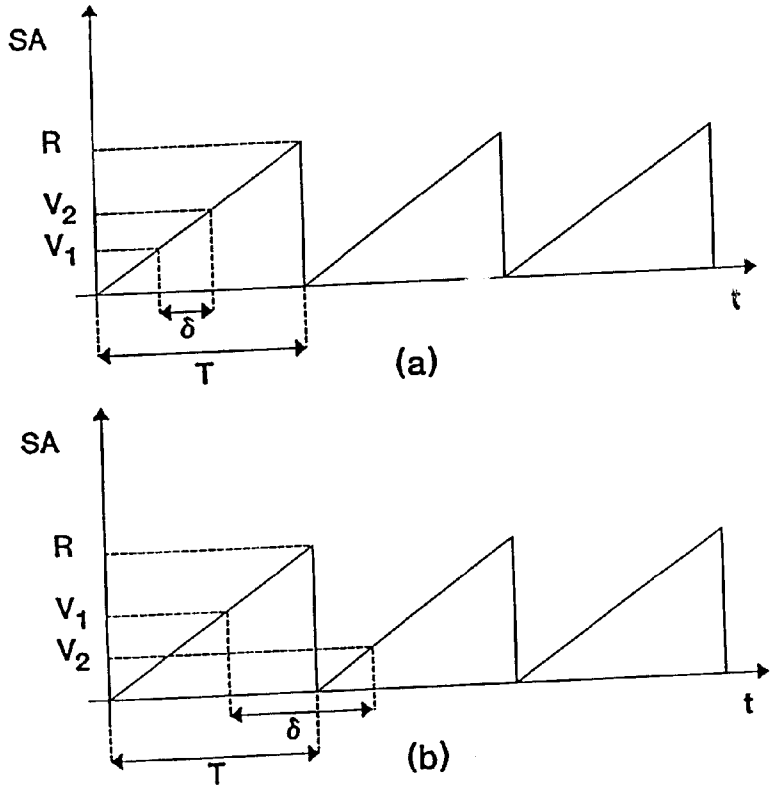


Figura 6

OFICINA ESPAÑOLA
DE PATENTES Y MARCAS

ESPAÑA

(11) ES 2 120 826

(21) N.° solicitud: 9402453

(22) Fecha de presentación de la solicitud: **24.11.94**

(32) Fecha de prioridad:

INFORME SOBRE EL ESTADO DE LA TECNICA

(51) **Int. Cl.**[6]: G08C 25/00

DOCUMENTOS RELEVANTES

Categoría	Documentos citados	Reivindicaciones afectadas
A	BASE DE DATOS PAJ en EPOQUE, 25.09.1991, TOKIO: JAPAN PATENT OFFICE & JP 31-050489 A (OKI ELECTRIC IND. CO., Ltd) 26.06.1991, resumen.	1
A	US 4337463 A (VANGEN) 29.06.1982, todo el documento.	1,7

Categoría de los documentos citados

X: de particular relevancia

Y: de particular relevancia combinado con otro/s de la misma categoría

A: refleja el estado de la técnica

O: referido a divulgación no escrita

P: publicado entre la fecha de prioridad y la de presentación de la solicitud

E: documento anterior, pero publicado después de la fecha de presentación de la solicitud

El presente informe ha sido realizado

[×] para todas las reivindicaciones

[ ] para las reivindicaciones n°:

Fecha de realización del informe	Examinador	Página
02.10.98	A. Figuera González	1/1