Revista de Comunicación Vivat Academia
ISSN: 1575-2844 · DOI: http://dx.doi.org/10.15178/va.2000.17.1-12
Julio/Agosto 2000 · Año III · nº 17 · pp. 1-12

MOTORES PASO A PASO, INTRODUCCIÓN A SU FUNCIONAMIENTO Y CONTROL EN LAZO ABIERTO

STEPPER MOTORS, INTRODUCTION TO HIS OPERATION AND OPEN LOOP CONTROL

AUTORES
Arturo Pérez París: Alumno de la Escuela Politécnica. Universidad de Alcalá
arturo.perez@arrakis.es

CURRÍCULUM VITAE

Alumno de la Escuela Politécnica de la Universidad de Alcalá de Henares (España).
Ingeniero de Soporte Técnico en Kone Elevadores

RESUMEN

Con esta tercera parte se pretende que nos familiaricemos un poco más con las secuencias de activación de las fases de los motores paso a paso. Para ello, prescindiremos del circuito controlador L297 y pasaremos a realizar el control generando nosotros mismos las señales ABCD desde un PC. El control será en lazo abierto y no tendremos potestad sobre la corriente que consuma el motor, a no ser que se implemente circuitería adicional.

PALABRAS CLAVE

Motores – Principio de funcionamiento – Control en lazo abierto

ABSTRACT

With this third part is to occupy ourselves a little more with the activation sequences of the phases of the stepper motors. To do this, omit the controller circuit L297 and pass control to make ourselves generating ABCD signals from a PC. The control loop is open and we have no authority over the current consumed by the engine, unless you implement additional circuitry.

KEY WORDS

Engines - Principle of operation - open-loop control

ÍNDICE

1. INTRODUCCIÓN

Con esta tercera parte se pretende que nos familiaricemos un poco más con las secuencias de activación de las fases de los motores paso a paso. Para ello, prescindiremos del circuito controlador L297 y pasaremos a realizar el control generando nosotros mismos las señales ABCD desde un PC. El control será en lazo

abierto y no tendremos potestad sobre la corriente que consuma el motor, a no ser que se implemente circuitería adicional. En primer lugar, se comprobará el funcionamiento del puerto paralelo del ordenador, utilizando para ello las funciones de entrada y salida que nos permite el programa "debug" de MSDOS. Para realizar una operación de salida, será suficiente con poner en el propio "prompt" de "debug" la siguiente secuencia:

-o 378 OF

Donde ´o´ indica que se trata de una operación de salida,´378´es la dirección del puerto paralelo y ´0F´es el dato que pretendemos sacar. Para realizar una entrada de datos, será suficiente con poner la letra ´i´ seguida de la dirección que queremos leer. Previamente habrá que sacar por el puerto paralelo el dato ´FF´, ya que, al tratarse de un puerto bidireccional, con salida en colector abierto, es necesario desactivar todas las salidas antes de realizar una lectura:

-o 378 0FF
-i378
1A
-

Al realizar la operación de entrada, aparecerá en pantalla el contenido de esa dirección ´1A´ en el caso del ejemplo.

Se realizará el montaje, que a continuación se muestra, y se hará un programa para el PC de forma que genere por los 4 bits de menor peso del puerto paralelo una secuencia que permita atacar al motor paso a paso en los mismos modos que lo hace el circuito L297.

ISSN: 1575-2844 · DOI: http://dx.doi.org/10.15178/va.2000.17.1-12

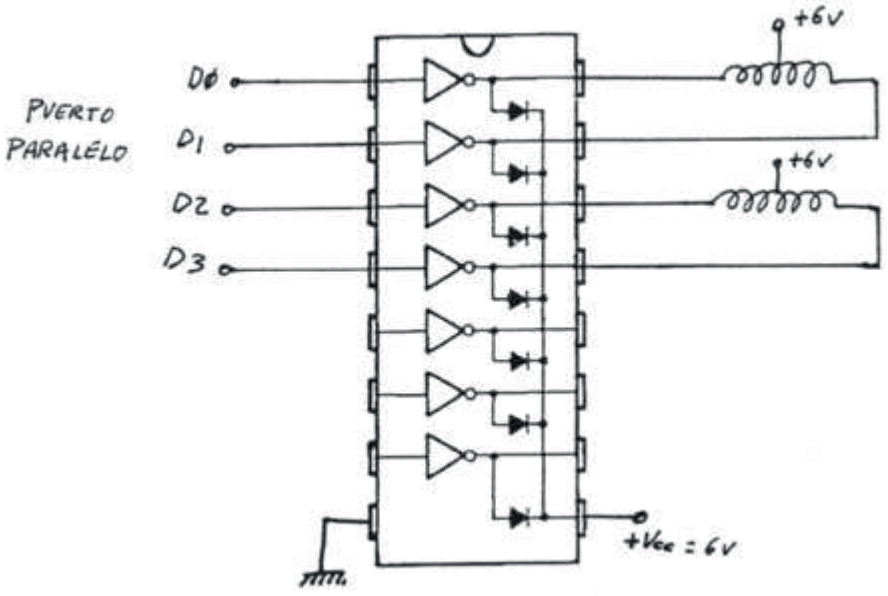

Fig. 22

El puerto paralelo de PC (CENTRONICS) es accesible a través de un conector DB-25. Estaba inicialmente ideado únicamente para conectar la impresora, por lo que no disponía de todas las posibilidades que iremos describiendo. Es por esto por lo que posiblemente, si se intentan realizar algunas operaciones sobre PC´s antiguos, no sea posible llevarlas a cabo. El puerto paralelo tiene una estructura común como la mostrada en la figura:

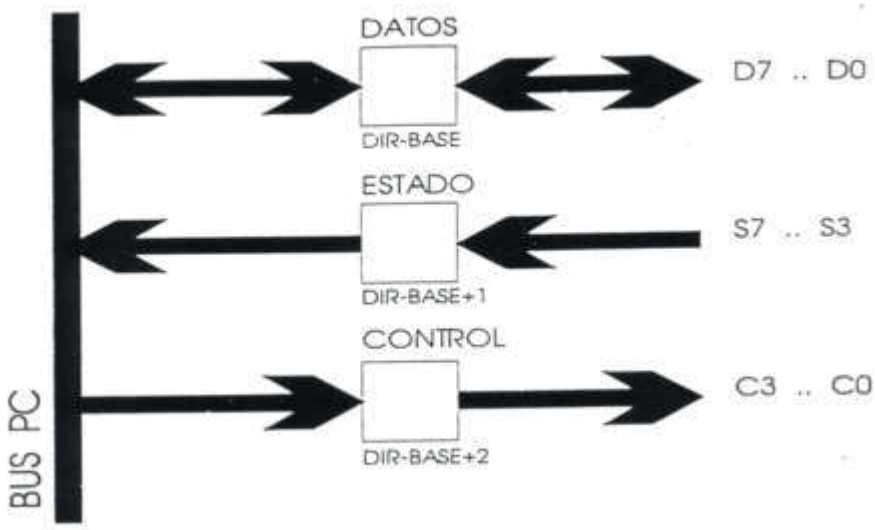

Fig. 23 (Ver Ref.2)

El puerto paralelo es un conjunto, a su vez, de 3 puertos básicos:

Uno bidireccional de 8 bits, el puerto de DATOS, que puede actuar como entrada o salida (inicialmente era sólo de salida) y que se encuentra situado en una posición que denominamos DIR-BASE. Esta dirección se puede obtener del propio PC. Una de las formas de obtenerla es leyendo la información que proporciona el ordenador en el proceso de arranque. Las direcciones base más usuales del puerto paralelo son 0x3BC y 0x378. Está formado por los bits D0 a D7 que están conectados a los pines de conector del 2 al 9 respectivamente. Debido a que las operaciones de entrada y salida se realizan a través de las mismas lineas, en el buffer de salida, que tiene sus salidas en colector abierto, debe sacarse el dato 0xFF antes de realizar una operación de entrada.

Un puerto de 5 bits de entrada, el puerto de ESTADO, está situado en la dirección siguiente a DIR-BASE, osea, en DIR-BASE +1. Cuando se realiza una operación de

ISSN: 1575-2844 · DOI: http://dx.doi.org/10.15178/va.2000.17.1-12

entrada, se leen simultáneamente 8 bits. Al hacer una operación de entrada sobre el puerto de estado, al disponer únicamente de 5 bits, éstos se cargan en los 5 bits de mayor peso. En S7 se reflejará el nivel del terminal correspondiente complementado, es decir, si las líneas tuvieran los niveles HHLLH, el valor leído sería 01001XXX. Esta situación se refleja en la siguiente figura:

Fig. 24

donde 'Si' representa el nivel de la línea de estado correspondiente. A continuación se relacionan los bits con el pin del puerto correspondiente:

BIT	PIN
S7	11
S6	10
S5	12
S4	13
S3	15

Fig. 25

Un puerto de 4 bits de salida, el puerto de CONTROL, está situado en la dirección siguiente a ESTADO, esto es, DIR-BASE+2. En el caso de hacer una salida de control, la información debe colocarse en los 4 bits de menor peso de la variable utilizada. En este caso los bits C3, C1 y C0 se traducirán en el puerto en los niveles

complementarios de los indicados en la información 0110; esto se traducirá en la salida en unos niveles HHLH.

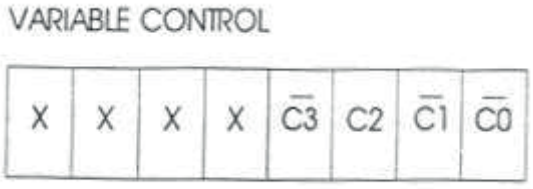

Fig. 26

La relación de pines de este puerto es la siguiente:

BIT	PIN
C3	17
C2	16
C1	14
C0	1

Fig. 27

Los pines del 18 al 25 están todos conectados a masa.

Con el fin de pasar a la praxis, lo primero que se implementó fue el siguiente programa que pide el ángulo de giro y la velocidad a la que queremos que gire el motor. El programa determinará, en función de la velocidad y del ángulo, si ha de hacer funcionar el motor en medio paso o en paso completo; además deberá realizar un pequeño control, sabiendo en todo momento en qué posición se encuentra el motor, obviamente, una vez inicializado:

ISSN: 1575-2844 · DOI: http://dx.doi.org/10.15178/va.2000.17.1-12

2. PROGRAMA

```
#include <dos.h>
#include <stdio.h>
char secuencia[]={5, 1, 9, 8, 10, 2, 6, 4};
char i, prim; //estas variables determinan la posición del motor y si está en fase de inicialización.
main()

{
void acciona (int medpas, char sentido,float velocidad,float angulo);
char sentido;
float velocidad,angulo,nmp;
signed int medpas;
prim=1;
while (1)

{
system("cls");
printf("µngulo (en grados sexagesimales): ");
scanf("%f",&angulo);
fflush(stdin);
printf("Velocidad de giro (en pasos/s): ");
scanf("%f",&velocidad);
fflush(stdin);
if (angulo<0)            //Aquí determino el sentido de giro en función
      sentido='d';        //del signo del  ángulo introducido.
else
      sentido='i';
```

```
nmp=(angulo/3.75)+0.5; //Determinación del número de medios pasos para
medpas=nmp;              //completar el  ángulo pedido con aproximación
                         //por redondeo.
acciona(medpas,sentido,velocidad,angulo);        //Acciono el motor de la forma
pedida.
prim=0
delay (1500);          //Le doy un tiempo de reposo (es prescindible).
}

}

 // Función en la que determinamos el modo de funcionamiento, qué velocidad se
ha de aplicar y las
//actuaciones "in situ" a realizar sobre el motor.

void acciona (int medpas, char sentido,float velocidad,float angulo)

{
int a,t,sup;
sup=0;       // En principio modo medio paso.
if ((velocidad<300)||(angulo>7,5))     //Determinación si se emplea paso completo o
medio paso

{          //en función de la velocidad de giro.
sup=1;
medpas=medpas/2;
}

t=1000/velocidad;     //Obtengo el tiempo que ha de pasar entre pasos (en ms.).
```


ISSN: 1575-2844 · DOI: http://dx.doi.org/10.15178/va.2000.17.1-12

```
if (sentido=='d')
      // Sentido de giro a derechas.

{
if (prim==1) i=0;
medpas=medpas*(-1); //Ajuste, para no manejar números negativos.
for (a=0;a<=medpas;a++)

{ //Recorrido a derechas de la tabla para generar el código convenido
printf("%i ",secuencia[i]);
outp (0x378,secuencia[i]);
i=i+sup+1;
if (i>=8) i=0;
delay(t);
}

}

if (sentido=='i')
// Sentido de giro a izquierdas.

{ //Recorrido a izquierdas de la tabla para generar el código convenido
if (prim==1) i=7;
for (a=medpas;a>=0;a--)

{
printf("%i ",secuencia[i]);
outp (0x378,secuencia[i]);
i=i-1-sup;
```

```
if (i<=-1) i=7;
delay(t);
}

}

return;
}
```

Hemos de realizar la observación sobre el error de no controlar mínimamente la posición del motor desde el principio; esto es, al conectar el sistema desconocemos la posición del motor, ya que carecemos de la circuitería necesaria para verificarlo. Por lo tanto, el primer ataque al motor deberá automatizarse como inicialización del sistema, con el fin de ubicar el motor en una posición conocida, por ejemplo tal y como lo hacen las impresoras con el cabezal de escritura al ser inicializadas o al ser encendidas por primera vez. De ahí la necesidad de las variables "i" y "prim" en el programa a nivel global. Quizás analizado por un programador experto (véanse los que vienen de las ingenierías de informática), nos dirían que esta solución es un desaprovechamiento enorme de memoria y en sí de los recursos del sistema. Pero "per se" es la solución más sencilla de ver y, a efectos docentes, la mejor de mostrar. Claro está que el programa es mejorable a todas luces. Empezando por una mejora estructural de los datos, como ya he dicho, y terminando por el añadido de funciones a las que, para altas velocidades de giro del motor, no se pasará de forma brusca, como lo hacemos aquí, sino de forma gradual, ganándose así la mejora en la respuesta del motor ante éstas y obteniéndose mejores prestaciones. Aun, con todo esto, como piedra angular de desarrollo de futuras aplicaciones, nos puede valer perfectamente tanto el programa expuesto como estas consideraciones.

ISSN: 1575-2844 · DOI: http://dx.doi.org/10.15178/va.2000.17.1-12

Como reseña final apuntaremos la idea de la implementación de un sistema que sea un híbrido entre los expuestos en la primera y segunda parte de esta memoria. Con esto indico la independización del motor respecto del PC, al usar un controlador (el L297) que únicamente recibirá, por ejemplo, la señal de reloj a una u otra frecuencia, en función de la velocidad angular que se requiera, además del control de sentido de giro, una buena inicialización, etc. Como resultado, se simplifica el software a implementar, mejorando las cualidades del sistema final.

3. AGRADECIMIENTOS

Bien, hasta aquí la tercera (y última) parte de este artículo. Quiero agradecer desde estas páginas la colaboración de D. José Antonio Bernal Martín, Dª. Ana Mª. Pérez París y Dª. Matilde París del Pozo, cuya ayuda ha hecho posible la redacción del mismo. Además quisiera agradecer la supervisión de D. Enrique Santiso Gómez, profesor de electrónica en el Politécnico de la UAH. Por último, aunque no por ello menos importante, agradecer a D. Julio Gutiérrez , su espléndido montaje de gráficos y dibujos, para las tres entregas. Con todo me despido de ustedes hasta el próximo artículo. Hasta entonces, y siguiendo la tradición que me he marcado en esta revista:

Espero que el presente escrito haya resultado del gusto del lector; si no hubiera sido así, desde aquí hago propósito de enmienda para que el próximo salga mejor. Si por el contrario le gustó (al más puro estilo "shakespeariano"), quedemos como amigos y volvamos a encontrarnos donde a la diosa fortuna más la complazca.

4. BIBLIOGRAFÍA

Ref.2: "Análisis, diseño, y realización de sistemas electrónicos de control discreto". Fco. Javier Rodríguez Sánchez, Felipe Espinosa Zapata, Enrique Santiso Gómez, Juan Jesús García Domínguez. Universidad de Alcalá de Henares.