International Journal of Electronic and Electrical Engineering.
ISSN 0974-2174 Volume 5, Number 1 (2012), pp. 69-82

http://www.irphouse.com

Microcontroller-Based Universal Stepper Motor Controllers

[1]Engr. Dr. Christiana C. Okezie, [2]Engr. Prof. Hyacinth C. Inyiama, [3]Engr. Nkolika O. Nwazor and [4]Engr. Chidiebele C. Udeze

[1,2]Department of Electronic and Computer Engineering
[3,4]Research & Development Department,
Nnamdi Azikiwe University, Awka Nigeria
Electronics Development Institute Awka, Nigeria

Abstract

Stepper motor's precision and accuracy in producing discrete stationary angular rotation has accounted for its use in various applications such as printers, sorters, CNC machines, volumetric pumps and so forth. This paper explored the essential features of a stepper motor-based control system that led to the development of a Microcontroller-based Universal Stepper Motor Controller. This versatile controller features several capabilities including the ability to control any type of Stepper Motor in various stepping modes, and to accept high-level command or command sequences from users, thus facilitating usage even by non-computer oriented personnel.

Keywords: Stepper motors, microcontroller, Controller, Universal, precision, opto-isolator

Introduction

For precise control of a motor, you can't just hook it up to a power source and let it run. You need exact position, speed and control [1]. With stepper motor (SM) incremental motion control can be achieved, which takes one angular or linear incremental step (depending on design) for every valid step command reaching it from a control system. It is therefore easy to keep track of the position of every load coupled to the SM shaft. A stepper motor is often referred to in terms of the number of stator windings (i.e. coils or phases) incorporated in its design. Hence, a 2-phase stepper motor has 2 stator windings; a 4-phase version has 4 and so on [2]. The essential features of a stepper motor control system are illustrated in Figure 1.

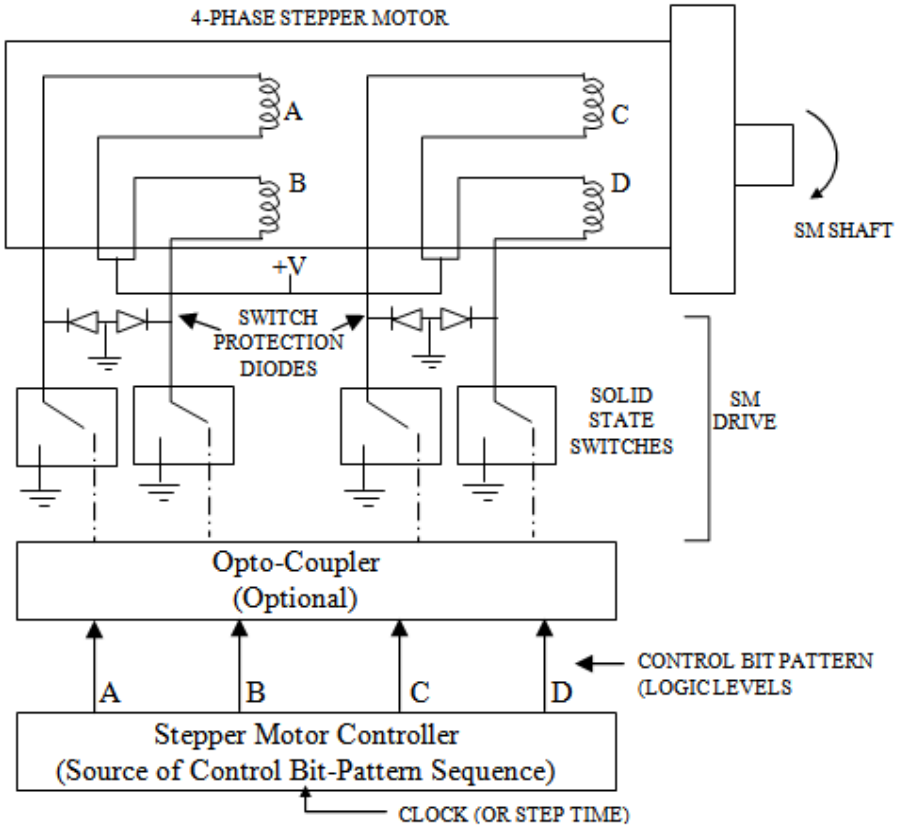

Figure 1: Essential Features of a Stepper Motor Control System

It is comprised of four main parts namely, a stepper motor, a stepper motor drive (or power circuit), a stepper motor controller, and an opto-isolator between the drive and the controller. The drive between the microcontroller and the stepper motor was because stepper motors consume high currents and voltages which the microcontroller cannot supply [3]. The solid-state switches comprising the drive are logic level operated. A Logic 1 control input applied via A, B, C, or D closes the corresponding switch, allowing current to flow through the coil controlled by the switch. Logic zero input to any of the switches causes the switch to open, thereby interrupting coil current flow. The pattern of logic 1's and logic zeros may be referred to as control bit-pattern. The mode of stepping of the SM is determined by the nature of the control bit-pattern sequence generated by its controller. Thus, a stepper motor in operation is always associated with a control bit-pattern sequence.

The maximum incremental distance for a stepper motor is the full step. The half step is so called because it corresponds to a distance equal to half that of a full-step. Similarly, a ¼ -step implies one quarter the distance covered in a full step, and so on. Typically, the smaller the step size the smoother the movement of the motor shaft and hence that of the load coupled to the motor shaft. Also the torque levels the motor is able to develop increase with decreasing step size.

The sophistication built into SM controllers differs according to need. A controller may cater for just clockwise (CW) motion or for counter clockwise (CCW) motion. The motion may also be possible in Full Steps only or in a particular fractional step only, depending on the bit-pattern sequence the controller is designed to generate. A versatile SM controller capable of bidirectional Full-Step or Fractional-Step control can also be realized. Many applications, for example, Research and Development (R & D) work, require the use of different stepper motors of varying number of phases. At a time of austerity, R & D funding is often slashed and it may not be possible to acquire separate SM controllers for the various stepper motors in use.

The use of microcontrollers and hence software control in SM controller implementations makes possible the realization of versatile, general purpose (i.e. universal), SM controllers which can be programmed to control any desired stepper motor depending on need. The use of software also enables the designer to incorporate many sophisticated features which would have been impractical if only rigidly interconnected (i.e. hardwired) circuits were to be used. The rest of this paper therefore high-lights the design steps leading to a Microcontroller-based Universal SM controller.

Methodology

A practical bipolar stepper motor interface is based on the MC3479 stepper motor driver IC. The MC3479 IC has a logic section that generates the proper control sequence to drive a bipolar stepper motor [4]. The implementation of a microcontroller-based SM controller involves two important steps. Firstly, the control bit-pattern sequence to be used is stored in the Read Only Memory (ROM) associated with the microcontroller. Secondly, a computer program (i.e. software) is written which accesses the bit-pattern sequence table one row at a time in a direction which corresponds to the desired type of motion (CW or CCW). To access any row of bit-patterns, the microcontroller must first determine its ROM address, where "address" as used here implies the unique code which identifies the location in memory where the bit-pattern is stored. It is possible to ease the programming effort required to generate the address of the next bit-pattern in a sequence by arranging for the bit-pattern output to the SM drive to serve as part of the address of the next bit-pattern in the sequence. This was achieved by the fully expanded control bit-pattern sequence derived from the basic control sequence.

The Generation of Control Bit-Pattern Sequence:

For any step size desired, the appropriate control bit-pattern sequence must be used. TABLE 1 through 3 shows the Full-Step, Half-Step, and 1/4 –step control bit-pattern sequences for a 5-phase stepper motor. Other fractional step control sequences (e.g. 1/8-step, 1/16-step, etc.) are possible for any stepper motor.

The bit-pattern in a sequence is usually generated one row at a time by the SM controller. The rate at which one row of bit-patterns is replaced by the next corresponds to the SM stepping rate since a step is taken by the motor only at the point when one row of bit-patterns is replaced by the next. The clock (or step time) is

an external signal applied to the SM controller and used to control the stepping rate (Figure.1). For every clock pulse a new row of bit-patterns is generated. Therefore, the clock frequency corresponds to the stepping rate.

When the rows of a bit-pattern sequence are generated (and applied to the SM drive) in a top-to-down order, the SM steps in a clockwise or forward direction. In contrast, a bottom-up order of bit-pattern sequence results in a counter-clockwise or backward motion. The rows of a bit-pattern are generated as an endless chain. Thus, in the top-down direction, the last row is immediately followed by the first while in the bottom-up direction; the first row is followed by the last.

Table 1: 5-Phase Full-Step Sequence

Full Step	Control bit-pattern				
	S1	S2	S3	S4	S5
1	1	0	1	0	1
2	0	1	1	0	1
3	0	1	0	1	1
4	1	1	0	1	0
5	1	0	1	1	0
1	1	0	1	0	1

Table 2: 5-Phase Half-Step Sequence

Full Step	Control bit-pattern				
	S1	S2	S3	S4	S5
1	1	1	1	0	1
2	0	1	1	0	1
3	0	1	1	0	1
4	0	0	0	0	1
5	0	0	0	1	1
6	0	0	0	1	0
7	1	0	0	1	0
8	1	0	0	1	0
9	1	1	1	1	0
10	1	1	1	0	0
1	1	1	1	0	1

Table 3: 5-Phase ¼ -Step Sequence

Quarter Step	Control bit-pattern				
	S1	S2	S3	S4	S5
1	1	0	1	0	1
2	½	0	1	0	1
3	0	0	1	0	1
4	0	½	1	0	1
5	0	1	1	0	1
6	0	1	½	0	1
7	0	1	0	0	1
8	0	1	0	½	1
9	0	1	0	1	1
10	0	1	0	1	½
11	0	1	0	1	0
12	½	1	0	1	0
13	1	1	0	1	0
14	1	½	0	1	0
15	1	0	0	1	0
16	1	0	½	1	0
17	1	0	1	1	0
18	1	0	1	½	0
19	1	0	1	0	0
20	1	0	1	0	½
1	1	0	1	0	1

Fully Expanded SM Control Sequences

TABLE 4 shows the Half-step bit-pattern sequence for a 4-phase SM while Table 5 shows its fully expanded version. A close inspection of Table 5 will show that it covers four possible modes of SM operation and is comprised of four main sections as follows:

Section (1): when the bit pattern under the CW/CCW and F/H (i.e. Full Step or Half Step) columns are both zero(or CW/CCW, F/H =0,0) signifying the counter-clockwise, half-step mode;

Section (2): When CW/CCW, F/H =0,1, signifying the counter-clockwise, full-step mode;

Section (3): when CW/CCW, F/H =1,0 which is the clockwise ,half-step mode;

Section (4): when CW/CCW, F/H =1,1, the clockwise full-step mode.

The control bit-pattern sequence for each of the four modes of operation appears under the column headed by the label "PRESENT STATES" and within one of the four sections identified above. The columns of bit-patterns under present states are labeled A, B, C, and D. Each row of bit-patterns under NEXT STATES (labeled A′, B′, C′, and D′) is obtained by determining the next appropriate control bit-pattern following that under PRESENT STATES (on the same row) when the mode of stepping is as defined by the CW/CCW, F/H patterns. For example, when CW/CCW, F/H =0, 0 (counter-clockwise half-step mode) and the present states ABCD=1001, the next states A B C D=0001. This is because in the reverse order of the 4-phase half step sequence (TABLE 4) 1001 (top row) is followed by 0001(bottom row). Similarly, when CW/CCW, F/H =1,1 (clockwise, full-step mode), and ABCD=1001, the next states $A^1B^1C^1D^1$=1010, which is the next bit-pattern in the full-step sequence in a top-down direction (TABLE 4), and so on.

The preparation of the PRESENT STATES and NEXT STATES table as illustrated in TABLE 5 is a necessary first step in the realization of a versatile SM controller. Such a table is often referred to as a fully expanded State Transition Table (STT) for the particular motor whose control bit-pattern sequence is so expanded. When such tables are used in SM controller design, error-free transitions between the full-step and half-step modes are made possible.

Table 4: 4-Phase Half-Step Sequence

Half Step	Control bit-pattern				Full Step codes
	A	B	C	D	
1	1	0	0	1	1
2	1	0	0	0	
3	1	0	1	0	2
4	0	0	1	0	
5	0	1	1	0	3
6	0	1	0	0	
7	0	1	0	1	4
8	0	1	0	1	
1	1	0	0	1	1

Table 5: Fully Expanded State Transition Table for a 4-Phase SM

	Stepping mode		Present States				Next States				
HEX	CW/$\overline{C}\overline{C}\overline{W}$	F/$\overline{H}$	A	B	C	D	A′	B′	C′	D′	HEX′
09	0	0	1	0	0	1	0	0	0	1	01
08	0	0	1	0	0	0	1	0	0	1	09
0A	0	0	1	0	1	0	1	0	0	0	08
02	0	0	0	0	1	0	1	0	1	0	0A

06	0	0	0	1	1	0	0	0	1	0	02
04	0	0	0	1	0	0	0	1	1	0	06
05	0	0	0	1	0	1	0	1	0	0	04
01	0	0	0	0	0	1	0	1	0	1	05
19	0	1	1	0	0	1	0	1	0	1	05
18	0	1	1	0	0	0	0	0	0	1	01
1A	0	1	1	0	1	0	1	0	0	1	09
12	0	1	0	0	1	0	1	0	1	0	08
16	0	1	0	1	1	0	1	0	1	0	0A
14	0	1	0	1	0	0	0	0	1	0	02
15	0	1	0	1	0	1	0	1	1	0	06
11	0	1	0	0	0	1	0	1	0	0	04
29	1	0	1	0	0	1	1	0	0	0	08
28	1	0	1	0	0	0	1	0	1	0	0A
2A	1	0	1	0	1	0	0	0	1	0	02
22	1	0	0	0	1	0	0	1	1	0	06
26	1	0	0	1	1	0	0	1	0	0	04
24	1	0	0	1	0	0	0	1	0	1	05
25	1	0	0	1	0	1	0	0	0	1	01
21	1	0	0	0	0	1	1	0	0	1	09
39	1	1	1	0	0	1	1	0	1	0	0A
38	1	1	1	0	0	0	0	0	1	0	02
3A	1	1	1	0	1	0	0	1	1	0	06
32	1	1	0	0	1	0	0	1	0	0	04
36	1	1	0	1	1	0	0	1	0	1	05
34	1	1	0	1	0	0	0	0	0	1	01
35	1	1	0	1	0	1	1	0	0	1	09
31	1	1	0	0	0	1	1	0	0	0	09

KEY: CW/$\overline{\text{CCW}}$, F/$\overline{\text{H}}$, A,B,C,D = HEX A′, B′, C′,D′ , =HEX′
E.G 0 0 1 0 0 1 = 09 0 0 0 1 01
First row above

When a ROM is used to store a fully expanded STT such as TABLE 5, the bit-pattern under $\overline{\text{CW}}$/$\overline{\text{CCW}}$, F/H, A,B,C, and D is viewed as a ROM address in which is stored the bit-pattern under A′,B′,C′, and D′ on the same row as the corresponding ROM address. Looking back at TABLE 5, the hexadecimal (i.e. HEX) values of each ROM address is shown under the column labeled HEX while its content is shown on the same row under the column labeled HEX′. When a microcontroller-based SM controller is in use, the microcontroller retrieves and outputs the first bit-pattern in the

sequence to be generated, delays for a step-time ,and then uses the previously output bit-pattern to form the address of the next bit-pattern to output. This is then retrieved and output to the SM drive and this process continues until the SM takes the desired number of steps.

The high capacity of ROMs relative to the number of unique bit-patterns in a fully expanded STT suggests the use of a single ROM to store the fully expanded bit-pattern sequences of all the stepper motors in use. The fully expanded STT for any particular stepper motor would then be reached by supplying extra address inputs which access only the portion of memory where it is stored. The present state bit-pattern then serve as the address of the next bit-pattern in the portion of memory selected by the extra address input lines. When a microcontroller is programmed to select and to generate the control bit-pattern sequence of any one of the stepper motors whose fully expanded bit-pattern sequence are in the ROM, a microcontroller based universal SM controller results[5].

Results and Discussion

A software-based universal stepper motor controller was achieved with the program design method presented. The difference in terms of programming effort, between a software-based controller intended for just one type of stepper motor and that designed for several types of stepper motors (each with a different number of phases) is minimal. This, as will be shown presently, is because all of the formal control parameters are the same for the various SM types and only the actual values supplied during initialization are different. Figure.2 illustrates a typical microcontroller-based stepper motor control system.

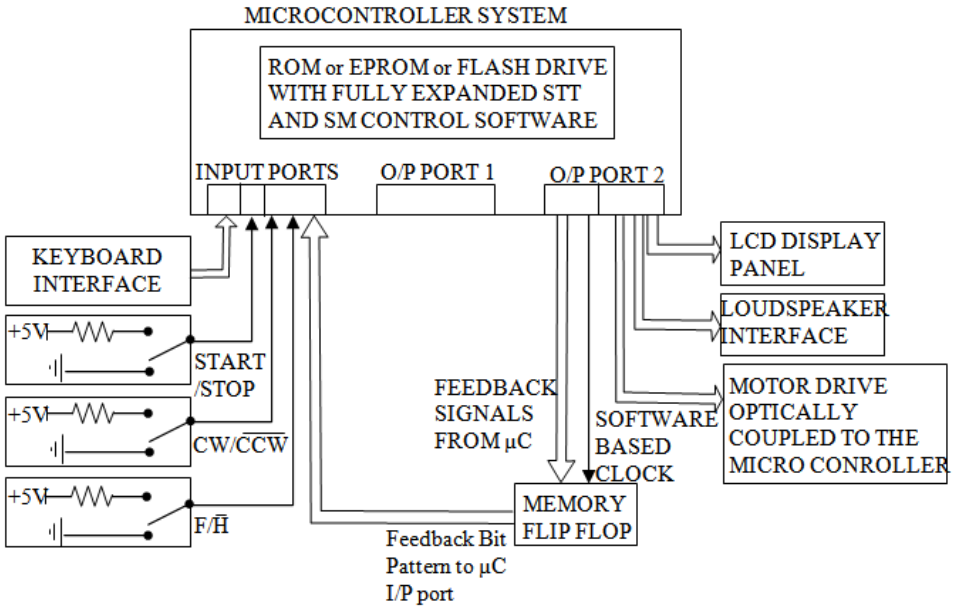

Figure 2: Microcontroller Based Stepper Motor Control System

The software which enables high level SM control commands to be input by the user and which also generates appropriate stepper motor control bit-pattern sequences, is usually stored in a ROM after all software errors had been corrected. A ROM has the characteristic that once the information is stored in it, it cannot be altered. That is, once written it may only be read, hence the phrase "Read Only" associated with its name. It is non-volatile and retains its contents even if power is switched off. An Erasable and Reprogrammable ROM (EPROM) is a convenient means of program storage during program development. An EPROM is similar to a ROM except that it may be reprogrammed in order to correct any imperfections in a previous programming.

The fully expanded State Transition Tables for all Stepper Motors catered for by the Universal Controller are also stored in the Application Program ROM (or EPROM), in the implementation developed by the authors. Tables 6, 7, and 8 contain the basic control-bit-pattern sequences from which the fully expanded versions were derived following the steps outlined in section 2 above.

The keyboard (Figure2) facilitates the input of variables or control parameters which make software-based controllers very flexible, while the LCD display enables the microprocessor to transmit responses to user commands in addition to providing the current status of the controlled device where necessary.

Figure3 is the programming flow chart for the software-Based Universal SM controller (SUSMC).TABLE 9 indicates the number of phases of each SM catered for by the SUSMC, as well as the modes of stepping (bidirectional or unidirectional Full-Step/Half-Step) which may be obtained from each. The programming style employed allows the range of motors catered for to be increased from 6 to 9 simply by appending the fully expanded state Transition Tables of additional motors to those in the Application program ROM/EPROM. The number of phases of each additional motor is also added to the list shown in TABLE 6.

Table 6: 3 –Phase Full-Step Sequence

FULL STEP	CONTROL BIT PATTERN					
	A^1	B^1	C^1	A^2	B^2	C^2
1	1	0	0	0	1	0
2	0	0	1	0	1	0
3	0	0	1	1	0	0
4	0	1	0	1	0	0
5	0	1	0	0	0	1
6	1	0	0	0	0	1

Table7: 6-Phase Full Step Sequence

FULL STEP	CONTROL BIT PATTERN		
	A	B	C
1	1	0	0
2	0	0	1
3	0	1	0

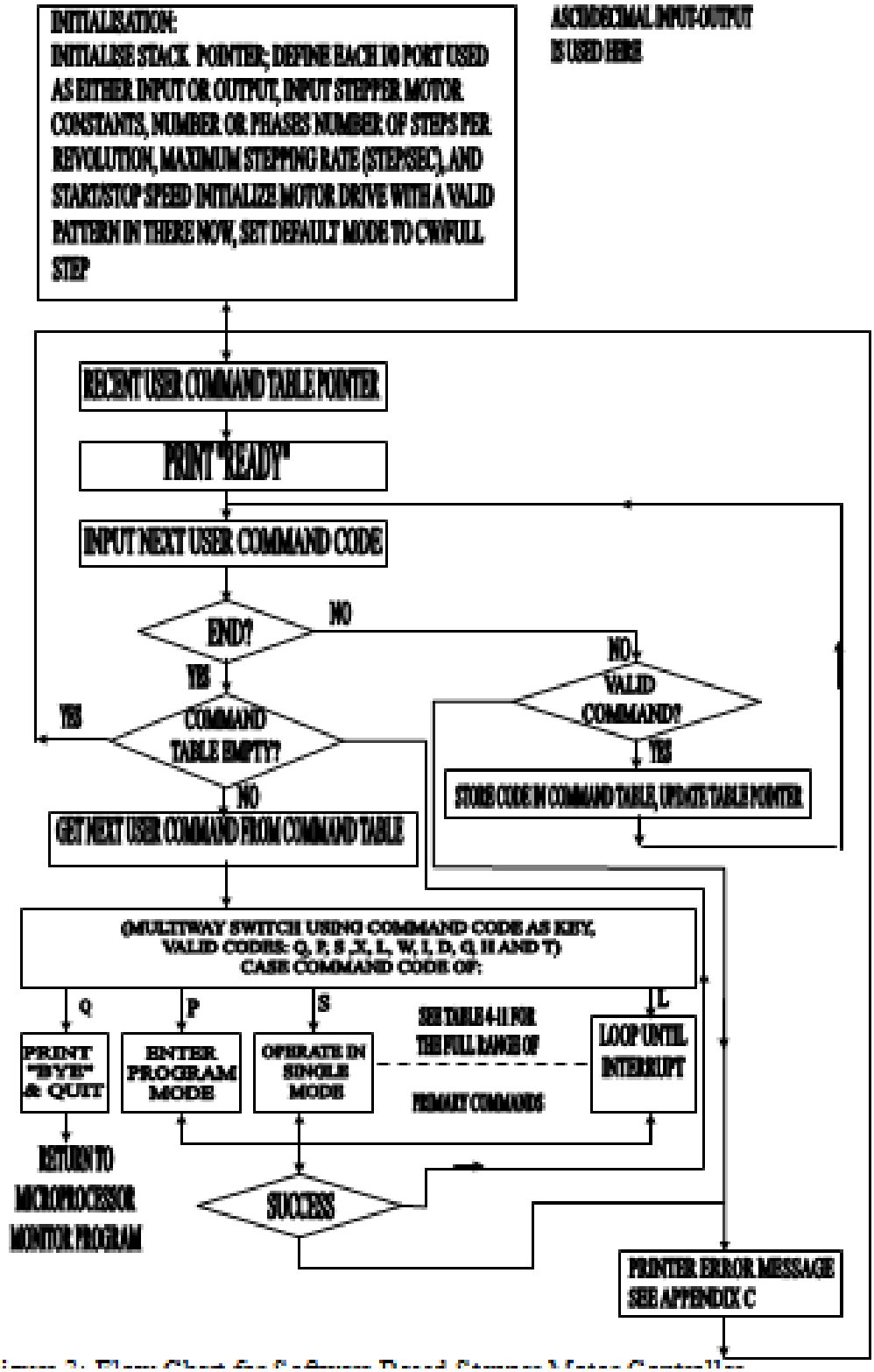

Figure 3: Flow Chart for Software Based Stepper Motor Controller

Table 8: 8-Phase Full Step Sequence

FULL STEP	CONTROL BIT-PATTERN							
	A^1	B^1	C^1	D^1	A^2	B^2	C^2	D^2
	1	1	1	1	0	0	0	0
	0	1	1	1	1	0	0	0
	0	0	1	1	1	1	0	0
	0	0	0	1	1	1	1	0
	0	0	0	0	1	1	1	1
	1	0	0	0	0	1	1	1
	1	1	0	0	0	0	1	1
	1	1	1	0	0	0	0	1

Table 9: Programmed Modes Of Operation For Stepper Motors

Number of Phases	CW	CW	Full-Step	Half-Step
2	✓	✓	✓	✓
3	✓	✓	✓	✓
4	✓	✓	✓	✓
5	✓	✓	✓	✓
6	✓	✓	✓	✓
8	✓	✓	✓	✓

No change in programming is necessary beyond the addition of such data as described above. Similarly, many more stepper motor types may be accommodated by one universal controller with minimal modifications to software but a larger ROM or EPROM device would be required.

Speed-time graphs discussion

The time interval between one motor step and the next determines the rate of stepping (measured in steps per second). The stepper motor takes one step each time the next control bit-pattern in the sequence is sent to the motor drive. Thus, the time interval between motor steps corresponds to the time between the sending of two successive bit-patterns to the SM drive. A software timing loop is used in the SUSMC to time out each motor step before the next step command is issued.

To illustrate this, suppose a software loop is such that it takes T microseconds to count down one by one, the value stored in a pair of registers or two predefined memory locations, then the maximum count that can be counted down to zero in one second by the software loop is given by:

$$N=1{,}000{,}000/T \ldots. \quad (1)$$

where T is in microseconds

The count 'n' required for motor step rate 'S' steps per second is therefore given by:

$$n = N/S \quad 2)$$

Any desired motor step time may thus be programmed as an integer count obtained after rounding up the result in equation (2). This is then decremented to zero in the software loop to represent the interval between major steps commands

When SM acceleration and deceleration is required, it is necessary to modify the countdown number ni between each major step and the next.

One good approach is to set

$$n_i = N/(S_i + a_i) \quad \text{for acceleration } \ldots(3)$$

And $$n_i = N/(S_i - a_i) \quad \text{for deceleration } \ldots(4)$$

Where

S_i = present speed in steps per second

a_i = incremental acceleration or deceleration value per step.

In ramping (i.e. acceleration/deceleration), it is normal practice to keep the slope of acceleration equal in magnitude to the slope of deceleration [6]. Where an even number of SM steps is to be taken during acceleration/deceleration (Figure. 4A), the controller starts from the pull-in speed of the SM and increases the rate of stepping per second (by the value of the acceleration rate) after each step, until half the total number of steps have been taken. The pull-in speed is one for which the SM can easily start or stop when carrying the load coupled to its shaft. It is often referred to as the start/stop speed of the motor. After half the desired number of steps, the controller begins to decelerate the motor which then stops after the same number of steps and time interval as for acceleration control. The graph for the odd number of steps has a flat top of 1 step and the remaining even numbers are covered as two equal slopes of acceleration and deceleration respectively (Figure.4B).

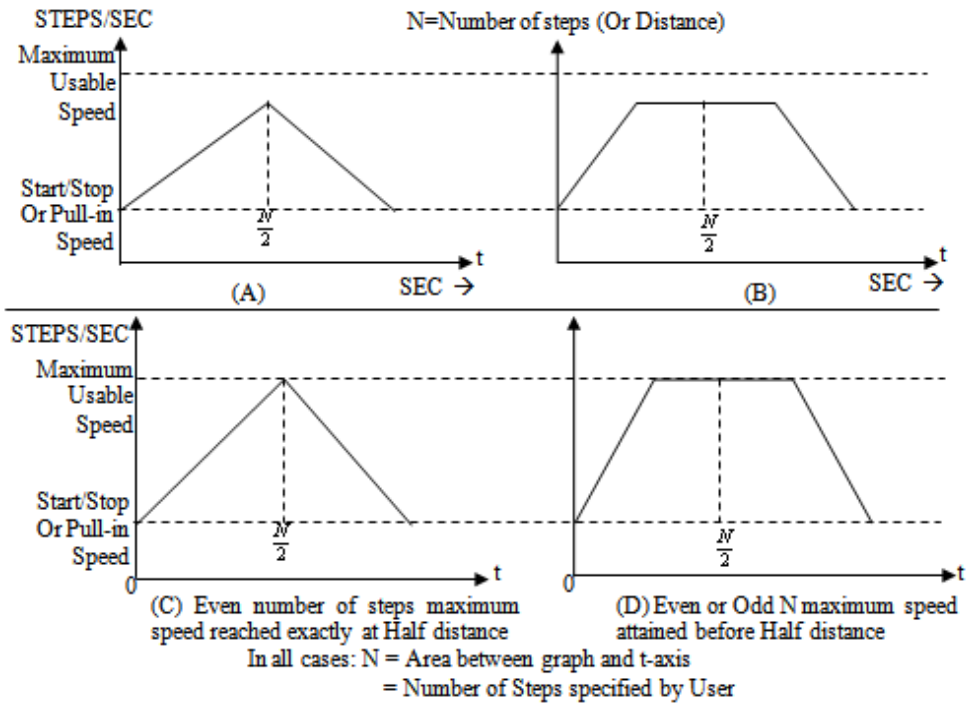

Figure 4: Speed-time graphs for stepper motors controlled by the universal software based controller.

If during acceleration, an estimated next-step rate ($Si+ a_i$) exceeds the maximum step rate specified for the motor, the microcomputer continues to use the maximum step rate (i.e. drives the motor at slew speed) until it is time to decelerate(Figure.4D).Similarly ,deceleration is never taken below the user specified start/stop speed of the SM. The step rate supplied by the user is assumed to have whatever mode (full step or half step)the user inputs during the PROGRAM MODE of the SUSMC which comes before command execution(See TABLES 10 and 11).

Conclusions

The use of a single ROM to store the fully expanded State Transition Tables of several stepper motors makes possible the realization of a low-cost universal SM controller. When a microcontroller is used to access the bit-pattern sequences in the ROM, programming effort is eased as the present output bit-pattern serves as the address of the next. Furthermore, the use of microcomputers in the implementation of universal SM controllers permit the incorporation of extra capabilities which facilitates usage, thus extending the usefulness of such controllers even to non computer oriented users. A very important feature of the SUSMC in this respect is the ability of the controller to accept high-level commands and the provision for high-level command sequences which minimizes the number of human interventions required once a control operation commences.

References

[1] http://www.ehow.com/how_4702247_build-stepper-motor-controller.retieved September 28, 2010
[2] Anita Azmi, 2009, State program vs ladder program for stepper motor control, Proceedings of MUCEET 2009 Malaysian Technical Universities Conference on Engineering and Technology, June 20-22, 2009, MS Garden, Malaysia.
[3] Kuo, B.C.; "Incremental Motion Control Systems and Devices" Department of Electrical Engineering, University of Illinois, U.S.A., in Co-operation with Warner Electronic Broke and Dutch company. Beloit,U.S.A.,and Wes tool Limited; Durham, England,(1974),pp.A-1 to C-32, J-1 to J-26.
[4] Mazidi M A, Mazidi J.C, The microcontroller and embedded systems, 2000, Prentice-Hall, Inc. Pearson Education, pp 28.
[5] W,K, Chen, Linear Networks and Systems (Book style) Belmont, CA: Wadsworth, 1993, pp123-135.
[6] Curtis D Johnson, Process Control Instrumentation Technology, 8th Edition, 2006, Prentice-Hall Inc. pp. 1-10
[7] Gordon McComb, 2001, working with stepper motors, The Robot Builder's Bonanza, second Edition, pp 279-293, McGraw-Hill, New York.
[8] Reston Condit, 2004, Stepper Motor Control Using the PIC16F684, Microchip Technology Inc.
[9] BiPOM Electronics, Inc, 2002, www.bipom.com.
[10] Roger L. Tokheim, Digital Electronics Principles and Applications, Fifth Edition, 1999, Glencoe McGraw-Hill, pp 112-115.