

Using DataSockets to Control a Stepper Motor Over the Network

Samer El-Haj-Mahmoud

Electronics Engineering Technology Program

Texas A&M University

Instructor's Portion

Summary

This lab aims at providing the students with hands on experience in remotely controlling and monitoring a stepper motor over the network using DataSockets programming and the digital output of the 6024E data acquisition card. The lab assumes that the students have prior exposure to LabVIEW, and that they know the basics of writing and debugging LabVIEW VIs (Virtual Instrument). The experiment will teach them how to apply their LabVIEW knowledge to control a stepper motor through a client/server network program. The experiment also covers the basic hardware to buffer the digital output of the card along with the basic concepts of DataSockets and the DataSockets server. The students should study stepper motor operation and concepts (step mode, poles, wiring) before coming to the lab.

The objectives for this experiment include exposing students to the basic theory behind stepper motors, teaching the concepts of digital interface and data sockets network control, and having the students improve their LabVIEW skills by developing a VI to control a stepper motor. The instructions provided in the student's section describe a step-by-step approach to write the basic structure of the VI. The rest of the implementation is left to the students. The VI provided is a sample implementation, and is intended for the instructor's reference use and *not* for the students. The diagram can be password protected in case the students want to look at the front panel and see how the VI they are writing should behave. Before performing this lab, the students should have finished a preliminary experiment, "Digital Control of a Stepper Motor", which is included as an appendix at the end of this experiment.

Uses

This experiment applies to general instrumentation and electronics systems interfacing courses in electrical engineering or engineering technology programs. The experiment can also be useful in basic LabVIEW courses, teaching the concepts of DataSockets, Remote Control, and Digital Input/Output through data acquisition cards, in addition to other LabVIEW techniques such as the use of the Logical Select and Array components.

Equipment List

- 2 PCs running MS Windows
(visit http://www.ni.com/labview/lv_sysreq.htm for requirements specific to your operating system)
- LabVIEW Full Development System version 6i or later.
- 6024E DAQ from National Instruments (part number 322072C-01)
- CB-50LP I/O Connector Block from National Instruments (part number 777101-01)
- NBI Ribbon Cable from National Instruments (part number 180624-10)
- LM18293 Four Channel Push Pull Driver from National Semiconductor (<http://www.national.com/pf/LM/LM18293.html> for the datasheet of this chip). Note: In case it is not possible to obtain this IC, any equivalent current buffer can work. A simple solution would be to use 4 NPN BJT's (1N2222) in an open collector mode to drive the stepper motor lines.
- A 4-coil Stepper motor, with 7.5° step resolution (48 internal poles), such as part number 26M048B from Thomson Airpax Mechatronics: <http://www.allegromicro.com/techpub2/airpax/smh15.pdf>
- Prototype board and jumper wires.

Setup

Follow the steps listed below to prepare the workstations for this experiment. The instructions assume you are using the equipment list shown previously.

Note: Most of the manuals that are referred to ship with National Instruments hardware and software. If you can't find your hardcopy of the

manuals, you can get them online at <http://www.ni.com/manuals>. If you encounter problems during setup, contact technical support at <http://www.ni.com/support>.

Before the Day of the Lab

1. Install LabVIEW on both PCs (see the *LabVIEW Release Notes* for your version of LabVIEW).
2. Install your 6024E board (see the *6024E User Manual*, or online at: <http://www.ni.com/pdf/manuals/322072c.pdf>).
3. Connect the cable to the 6024E card and to the I/O connector block.

On the Day of the Lab

1. Power up both computers.
2. Start LabVIEW.

References

- NI's web site: <http://www.ni.com>
- LM18293 datasheet:
<http://www.national.com/ads-cgi/viewer.pl/ds/LM/LM18293.pdf>
- Thomson Airpax Stepper Motor Handbook:
<http://www.allegromicro.com/techpub2/airpax/airpax.htm>
- "How Struff Works" web site: <http://www.howstuffworks.com/>
- D. Trhomson, and P. Sweat. "Basic Use of DataSockets in LabVIEW", July 2001. Online at:
<http://www.ses-co.com/DataSockets.doc>

Student's Portion

Introduction

In this experiment, you will write LabVIEW VIs to remotely control a stepper motor using DataSockets Programming and the digital output of the 6024E Data Acquisition Card (DAQ). You will write two VIs, a client and a server. In addition, you will have to use the DataSockets Server. This lab requires background in stepper motor basic theory, including the concepts of poles and half/full steps control codes. These concepts are summarized in the Theory section of this experiment. In addition to writing the VI, you are required to build an interface circuit to drive the motor from the DAQ digital outputs.

Objective

- To learn how to use LabVIEW for Digital I/O through a DAQ card.
- To learn about stepper motor basics and how to control a stepper motor using the digital output of a DAQ in full/half step mode and clockwise/counterclockwise direction, while controlling the speed of rotation and displaying the current position.
- To learn the basics of network programming in LabVIEW and DataSockets.

Theory

1. Stepper Motor

For stepper motor theory, refer to the theory section of the experiment “Digital Control of a Stepper Motor” that is available in the appendix.

2. DataSockets

DataSockets is a network programming technology to simplify data exchange between computers and distributed I/O devices. This technique can be used to pass data over the Internet or Local Area Networks (LANs), without the complexity of low-level TCP programming. The data can be used for remote monitoring and control of processes that can be physically located far away from the operator. DataSockets also makes sharing of scientific and engineering data between software applications and between networked computers as easy as reading and writing data to files.

The main protocol that is used with DataSockets is called Data Sockets Transfer Protocol (DSTP). Connections made with standard URLs of the following format:

`dstp://machine_name/tag_name`

Figure 4 illustrates the client/server model in DataSockets network programming. The terms “Client” and “Server” refer to the two LabVIEW programs (or VIs) that you will write in this experiment. Each VI will run on a different machine on the network, or they can both run locally on the same machine. The operator usually controls the client side, while the actual motor is connected to the DAQ on the server host.

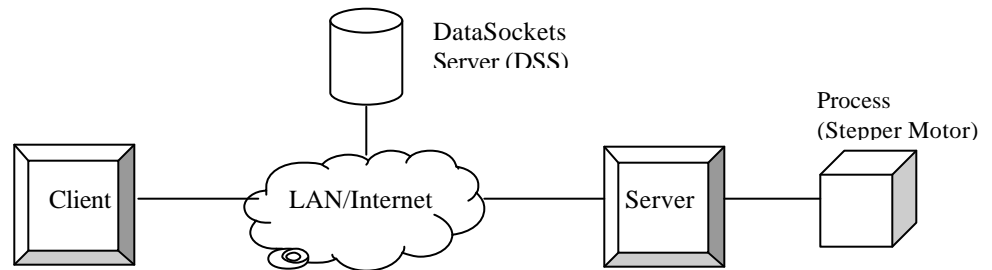


Figure 4 – Client/Server network diagram with DataSockets

The client VI has a GUI that contains all the necessary input to the process (or controls) and the feedback output from the process (the indicators). The operator uses the client to control the remote process and monitor the feedback measurements. All what the client does is to package the inputs from the controls, send (or write) them to the server, and read back from the server the feedback, unbundle them and display them on the indicators.

The server VI doesn't contain any controls. It reads from the network the values sent to it from the client, unbundle them and use them as the inputs to the control program that drives the process. It also collects the feedback measurements from the process and bundle them and write them back to the client to be displayed.

Each DSTP connection is unidirectional; it supports sending the data from one of the hosts to the other. This means we will need two DSTP connections for a closed loop communication between the client and the server programs. One connection will read from the client and write to the server, and the other will read from the server and write to the client. When creating the DSTP connections, it is necessary to assign a new tag to each connection.

For DataSockets connections to be established successfully, a DataSockets Server (DSS) must be running on the network. All communication through the DSTP between the client and the server programs are passed through the DataSockets Server. The DataSockets Server can be running on either the client or the server machines, but it can also be running on a third independent machine. Most programmers prefer to run the DataSockets

Server on the same machine that hosts the server VI. The important thing to remember is that all DSTP connections must use the IP address of the DataSockets Server (DSS) on both the client and server sides. This is explained in the following table:

	First connection	Second Connection
Client Side	dstp://DSS_IP/tag1 (Client Write)	dstp://DSS_IP/tag2 (Client Read)
Server Side	dstp://DSS_IP/tag1 (Server Read)	dstp://DSS_IP/tag2 (Server Write)

The DSS can be referred to by its IP address or DNS name. In addition, if the DSS is running locally on the client or the server sides, then the following IP address can be used on that side: 127.0.0.1, which is equivalent to the DNS name: *localhost*.

The DataSockets operation can be performed in LabVIEW using two main VIs: DataSockets Read and DataSockets Write. Both VIs need the address of the DataSockets to be specified as a URL (as mentioned above) of string type. In addition, the DataSockets Read requires a “type” bundled input to be able to cast the unbundled message components to their original types. For example, if the client sent a bundle consisting of an integer, a string and a double, then the server Read must be connected to a bundle of integer, string and double constants in that specified type. The constants can simply be 0 or empty strings. They should be bundled together and connected to the “type” input of the Server Read sub VI. This way, when you unbundled the output of the Read (or the “data”) the VI will cast each of them to its corresponding type.

Starting with LabVIEW version 6.1, a new and improved method for remote control of a VI is available. Using a simple enable feature, you can allow your VI to be completely controlled and viewed online (with any web browser) without adding any data sockets code to your program. This requires a server (provided with LabVIEW 6.1) and a LabVIEW client browser plugin, available from NI’s web site. This method is not going to be used in this experiment.

Finally, to be able to connect from one machine to the DataSockets Server on another, you need to set the DataSockets Server Manager access rights to “every host”. For more information on DataSockets, refer to the examples provided with your LabVIEW that are also available in the help system or you can refer to this useful document:

<http://www.ses-co.com/DataSockets.doc>

Pre-Lab Preparation

Read this experiment before coming to the lab. You should understand the operation of the stepper motor, and know the difference between full step and half step modes. You should also understand the difference between the client and the server programs, and the use of the DataSockets Server.

Bring the following to lab with you:

- This experiment.
- Your lab notebook and pencil.
- 2 virus-free formatted 3.5-inch floppy disks (always make a backup copy of your code on the second disk).

Answer the following questions in the datasheet provided at the end of this experiment *before* coming to the lab. Remember to include these answers in your lab report.

1. What is the main difference between the client and server programs in this experiment.
2. What are the VIs used for DataSockets communication.
3. What is the DataSockets Server? How many copies of it should be running in this experiment and where?
4. If the DataSocket server is running on the machine with IP address 192.196.0.1 , and the client machine has the address of 192.196.0.2 while the server machine has IP address 192.196.0.3. How would you write the dstp URLs for the following situations:
 - Client DataSockets Read
 - Client DataSockets Write
 - Server DataSockets Read
 - Server DataSockets Write
5. If the DataSockets Server is running on the same machine as the server program, how would your answers for question 4 differ? (Hint: use the *localhost* DNS name).
6. What is the logic 1 output voltage on the 6024E card? What is the logic 0 output voltage? What is the lowest input voltage to produce a logic 1 input? And the highest input voltage to produce a logic 0 input? Compare your answers to TTL and CMOS standards.
7. How do you produce a tri-state output? What is the voltage level?
8. How fast can a single digital output on the 6024 be changed?

9. Which VI is the most suitable for the Digital Control of the stepper motor (Functions Menu → Data Acquisition → Digital I/O)?
10. How will you store the last position of the motor in the LabVIEW VI ? How will you initialize this value?
11. What is the difference between a Digital Port and a Digital Line in the Digital I/O VIs?
12. If the consecutive positions of a Stepper Motor in the clockwise directions are numbered 0, 1, 2, 3...7, obtain the binary code given to the stepper motor corresponding to each position.

Workstation Details

Your workstation should have the following items:

- Two computers with National Instruments LabVIEW software and connected over a Local Area Network (LAN)
- National Instruments DAQ board (inside one of the computers)
- National Instruments DAQ board terminal block
- Stepper motor (48 internal poles with 4 coils and 5 terminals)
- LM 18293 push-pull quad line driver (or an alternative current driver hardware).
- Prototype board and jumper wires

Lab Procedure

1. Before starting this experiment, you need to complete a preliminary experiment, “Digital Control of a Stepper Motor”, that is available in the appendix. The VIs that you are going to write in that experiment will be used in the coming steps of this lab.
2. Start by writing the client side VI. Start a new VI in LabVIEW on the machine that will run the client (this doesn't need a DAQ to be installed inside it). You should then add the following controls and indicators to the front panel. You can copy and paste them from the VI of the preliminary experiment. For a description of the usage of each item, refer to the preliminary experiment.
 - a. Half-Stepping/Full Stepping-Mode Switch.
 - b. Clockwise/Counter-Clockwise-Rotation Switch.
 - c. Speed-of-Rotation Digital Control.
 - d. Number of Steps Digital Control.
 - e. Power Switch.
 - f. Current Position indicator.
 - g. Current Output Code.
 - h. Read URL Input String: This is where you will type the DataSockets URL for the client read.
 - i. Write URL Input String: This is where you will type the DataSockets URL for the client write.
2. After adding those controls, start wiring the VI diagram of the client program. Your code will need the following VIs:
 - a. An infinite while loop, which will wrap the entire client program.
 - b. A DataSockets Read and a DataSockets write VIs. You will need to connect the URL input of both of them to the corresponding input string (read or write) on the front panel.
 - c. Two bundle and one unbundle VIs. One of the bundle VIs will be used to bundle together the controls to form the inputs (or data) of the DataSockets Write, which will be the stepping mode, the rotation direction, the speed of rotation, the number of steps and the power switch. The unbundle will be used at the output (data) of the Read DataSockets VI to separate the values read from the

network and connect them to their corresponding indicators. These will be the current position indicator and current output code. Finally, the last bundle is going to be used to cast the types of the data output of DataSockets Read VI. It should be connected to the “type (Variant)” input of DataSockets Read. The types that you must bundle should correspond to the types of the data items unbundled at the output of DataSockets Read, i.e. TF (for stepping mode), TF (for rotation direction), U32 (for speed of rotation), etc..

3. Next, you will start writing the server program that will run on the second machine, which has the DAQ card installed. This program is essentially the same one that was created in the preliminary experiment but with added data sockets functionality. The front panel of the server program should contain the following controls and indicators. For the first two items, refer to the preliminary experiment for an explanation of each item.
 - a. Current Position indicator.
 - b. Current Output Code
 - c. Read URL Input String: This is where you will type the DataSockets URL for the server read.
 - d. Write URL Input String: This is where you will type the DataSockets URL for the server write.
4. Start wiring the server program VI diagrams with the controls and indicators to implement the functionality of the stepper motor control. The best way to do this is to start with the code written in the preliminary experiment, and do the following changes to it
 - a. Delete all controls from the front panel, only leaving the current position indicator and current output cycle.
 - b. Change the main For Loop into a while loop. The loop stops when the current iteration is greater than or equal the number of iterations that will be received from data sockets read (from the client).
 - c. Add one DataSockets Read and one DataSockets write VIs. You will need to connect the URL input of both of them to the corresponding input string (read or write) on the front panel.
 - d. Add two bundle and one unbundle VIs. These are going to be used in a way similar to that of the client program. You will need them to collect together the data to be sent to the client (server write) and to separate the data read from the client. Make sure that the data you write in the server is in the same order you read in the client, and vice versa.

- e. Connect the bundle to the indicators: current output code and current position indicator (in the same order specified in the client). Connect the output on the bundle to the DataSocket write.
 - f. Connect the output of the DataSocket read to the unbundled VI. The outputs of the unbundled go to stepping mode, the rotation direction, the speed of rotation, the number of steps and the power switch (in the same order specified in the client). The other bundle is going to be used to cast the types of the data output of DataSockets Read VI. It should be connected to the “type (Variant)” input of DataSockets Read. The types that you must bundle should correspond to the types of the data items unbundled at the output of DataSockets Read
5. These are the basic VIs that you will need to have working client and server programs. You will still need more VIs in the server program to control the speed of rotation (milliseconds wait) and to show the current motor position. Remember that one whole rotation is 360 degrees, and that the motor goes 3.75 degrees for each half step and twice as much for full steps. If you have any questions about writing the remaining code, ask your lab instructor.
 6. After finishing your code save it to the floppy disks. Then, connect the DAQ digital output pins with the motor and the current driver IC. Follow the connections in the diagram in Figure 5.
 7. Next, you need to configure and run the DataSockets server. First, run the DataSockets Server Manager from the Start menu (Start → Programs → National Instruments → DataSockets → DataSockets Server Manager). Set permissions for all groups to “everyhost”. Then, run DataSockets Server.
 8. Run your client and server programs on the corresponding machines. You need to set the correct URLs for client and server read/write depending on the address of the machine running the DataSockets Server. Make sure the tag for client read and server write are identical, and the same for client write and server read. Refer to the theory section for more information on this. If you have any questions, ask your lab instructor.
 9. Test your code and connections by moving the motor as follows:
 - Full step clockwise
 - Full step counter clockwise
 - Half step clockwise
 - Half step counterclockwise
 - All the above combinations with varying speed

- Loss of torque observation when using half step vs. full step.

Note down the angle of rotation (the current position) while you are experimenting with the possible combinations above. If you are having problems controlling the motor over the network, or if a network is unavailable, you may try to run both the client and the server along with the DataSockets Server on the same machine. Ask your instructor for help.

Lab Report

Your report must be typed and of professional quality. You should include the following in your report:

1. All schematics, which must be computer-generated. Components must be labeled with circuit references and values. The schematics must be fully compliant with standard engineering practices for circuit depiction.
2. Snapshots of the front panel and block diagrams of all VI's and Sub VI's that you wrote or used.
3. A Theory section describing the theory behind the concepts used in this lab.
4. An implementation section describing the details of your VI code and the logic flow.
5. The test results and observations.
6. The answers to the Pre-lab questions in the provided datasheet.
7. Selected sections of the datasheet of the PCI 1200, the stepper motor and/or the LM 18293.

Data Sheet

1. Answer :
.....

2. Answer :

3. Answer :
.....

4. Answer :
.....

5. Answer :
.....

6. Answer :
.....
.....

7. Answer :

8. Answer :
.....

9. Answer :
.....

10. Answer :

Position	A1	A2	B1	B2
0				
1				
2				
3				
4				
5				
6				
7				

APPENDIX

Digital Control of a Stepper Motor

Samer El-Haj-Mahmoud

Electronics Engineering Technology Program

Texas A&M University

Instructor's Portion

Summary

This lab provides the students with hand on experience in controlling a stepper motor using the digital output of the 6024E data acquisition card. The Lab assumes that the students have prior exposure to LabVIEW, and that they know the basics of writing and debugging LabVIEW VI (Virtual Instrument) code. The experiment will teach them how to apply their LabVIEW knowledge to control a stepper motor. The experiment also covers the basic hardware to buffer the digital output of the card. The students should study stepper motor operation and concepts (step mode, poles, wiring) before coming to the lab.

The objectives for this experiment include exposing students to the basic theory behind stepper motors, teaching the concepts of digital interface, and having the students improve their LabVIEW skills by developing a VI to control a stepper motor. The instructions provided in the student's section describe a step-by-step approach to write the basic structure of the VI. The rest of the implementation is left to the students. The VI provided is a sample implementation, and is intended for the instructor's reference use and *not* for the students. The diagram can be password protected in case the students want to look at the front panel and see how the VI they are writing should behave.

Uses

This experiment applies to general instrumentation and electronics systems interfacing courses in electrical engineering or engineering technology programs. The experiment can also be useful in basic LabVIEW courses, teaching the concepts of Digital Input/Output through data acquisition cards, in addition to other LabVIEW techniques such as the use of the Logical Select and Array components.

Equipment List

- PC running Windows, Macintosh, Linux, Sun, or HP-UX
(visit http://www.ni.com/labview/lv_sysreq.htm for requirements specific to your operating system)
- LabVIEW Full Development System
- 6024E DAQ from National Instruments (part number 322072C-01)
- CB-50LP I/O Connector Block from National Instruments (part number 777101-01)
- NBI Ribbon Cable from National Instruments (part number 180624-10)
- LM18293 Four Channel Push Pull Driver from National Semiconductor
(<http://www.national.com/pf/LM/LM18293.html> for the datasheet of this chip). Note: In case it is not possible to obtain this IC, any equivalent current buffer can work. A simple solution would be to use 4 NPN BJTs (1N2222) in an open collector mode to drive the stepper motor lines.
- A 4-coil Stepper motor, with 7.5° step resolution (48 internal poles), such as part number 26M048B from Thomson Airpax Mechatronics:
<http://www.allegromicro.com/techpub2/airpax/smh15.pdf>
- Prototype board and jumper wires.

Setup

Follow the steps listed below to prepare the workstations for this experiment. The instructions assume you are using the equipment list shown previously.

Note: Most of the manuals that are referred to ship with National Instruments hardware and software. If you can't find your hardcopy of the manuals, you can get them online at <http://www.ni.com/manuals>. If you encounter problems during setup, contact technical support at <http://www.ni.com/support>.

Before the Day of the Lab

3. Install LabVIEW (see the *LabVIEW Release Notes* for your version of LabVIEW).
4. Install your 6024E board (see the *6024E User Manual*, or online at: <http://www.ni.com/pdf/manuals/322072c.pdf>).
5. Connect the cable to the 6024E card and to the I/O connector block.

On the Day of the Lab

1. Power up the computer.
2. Start LabVIEW.

References

- NI's web site: <http://www.ni.com>
- LM18293 datasheet:
<http://www.national.com/ads-cgi/viewer.pl/ds/LM/LM18293.pdf>
- Thomson Airpax Stepper Motor Handbook:
<http://www.allegromicro.com/techpub2/airpax/airpax.htm>
- "How Stuff Works" web site: <http://www.howstuffworks.com/>

Student's Portion

Introduction

In this experiment, you will write a LabVIEW VI to control a stepper motor using the digital output of the 6024E Data Acquisition Card (DAQ). This lab requires background in stepper motor basic theory, including the concepts of poles and half/full steps control codes. These concepts are summarized in the Theory section of this experiment. In addition to writing the VI, you are required to build an interface circuit to drive the motor from the DAQ digital outputs.

Objective

- To learn how to use LabVIEW for Digital I/O through a DAQ card.
- To learn about stepper motor basics and how to control a stepper motor using the digital output of a DAQ in full/half step mode and clockwise/counterclockwise direction, while controlling the speed of rotation and displaying the current position.

Theory

A motor is an electromagnetic rotating machine where current flow in a coil around the poles induces an electromotive force that attracts or repels a rotating part in the center of the motor. DC motors consist of a **rotor**, which is the center part and a **stator**, which is the outer block as in Figure 1. To allow the rotor to turn without twisting the wires, the ends of the wire loop are connected to a set of contacts called the **commutator**, which rubs against a set of conductors called the **brushes**. Stepper motors are very different from DC motors in the way they are physically composed and used. Unlike DC motors, which may be composed of two large stator magnets, one on each side of the rotor, stepper motors are composed of many ferrite poles surrounding a permanently magnetized stator core. Figure 1 shows the two basic models of DC and stepper motors.

As illustrated in Figure 1, each pole within the stepper motor is wound by a thin wire. When any one of these wires is energized by current passing through it, a magnetic field is created across the ferrite material. This thereby induces a magnetic North and a magnetic South on the pole.

A magnetized stator pole will attract a magnetized portion of the rotor with the opposite polarity. By using a specific combination of energized poles, the rotor can be “held” in place by the magnetism, or it can be made to rotate by energizing adjacent poles and de-energizing the current ones so that the rotor is then attracted to the adjacent source of magnetism.

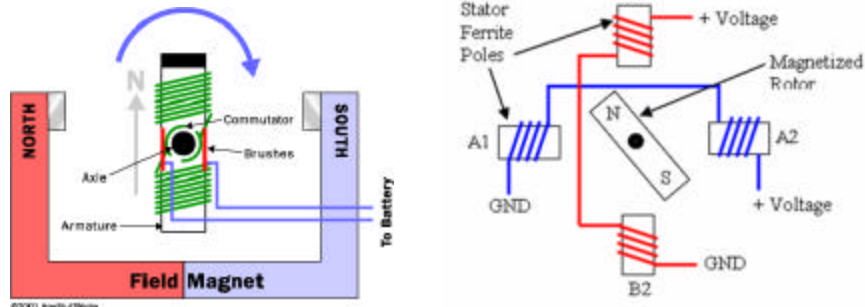


Figure 1 - Comparing a DC motor (left) to a Stepper Motor (right)

(Source: <http://www.howstuffworks.com/motor1.htm>)

The stepper motor shown in Figure 1 has 4 internal ferrite poles, but pairs of poles are connected together to form one coil. The motor we will use in this lab has 48 internal ferrite poles, but they are connected to form four coils. Each of the four coils forms one terminal of the motor. In addition, there is a fifth terminal that corresponds to a common line with all coils. This common terminal can be used as the common ground (or a common source) for the motor. For a four-coil motor, a four-bit code sequence is needed to energize the appropriate poles at any one time. The sequence needed to make the rotor turn can be done in either “full step” or “half step” increments. The difference between these two modes is that the application of some of the codes is omitted. This can be seen in the tables in Figure 2. Both the speed and direction may be changed at any instant by applying the appropriate codes at the stepper motor’s signal lines.

Normal 4-Step Sequence

Step	Q ₁	Q ₂	Q ₃	Q ₄
1	ON	OFF	ON	OFF
2	ON	OFF	OFF	ON
3	OFF	ON	OFF	ON
4	OFF	ON	ON	OFF
1	ON	OFF	ON	OFF

1/2 Step 8-Step Sequence

1	ON	OFF	ON	OFF
2	ON	OFF	OFF	OFF
3	ON	OFF	OFF	ON
4	OFF	OFF	OFF	ON
5	OFF	ON	OFF	ON
6	OFF	ON	OFF	OFF
7	OFF	ON	ON	OFF
8	OFF	OFF	ON	OFF
1	ON	OFF	ON	OFF

Figure 2 - Stepper Motor switching sequence

(Source: <http://www.allegromicro.com/techpub2/airpax/smh29.pdf>)

The angular rotation of the rotor is half the distance when “half-step” codes are used than when “full-step” codes are used. This means that it takes half as many full-step codes to make the rotor make one complete turn compared to using half-steps. This also means that a “finer” angular increment is possible when using half-step codes. Depending on the application, this may be an advantage. In the example given, the rotor shifts 45 degrees between the applications of two half-step codes, while it shifts 90 degrees between the applications of two full-step codes. The more poles a motor has, the smaller the rotational increments on both half and full steps. On the other hand, this also means that more code sequences are needed to make one complete turn. There are advantages and disadvantages to having a stepper motor with an N number of poles. The tradeoff is rotational speed versus angular resolution. The following formula defines angular resolution of a stepper motor.

$$\text{Angular Resolution} = \frac{360 \text{ Degrees}}{2 \times \text{number of internal poles}}$$

For the motor used in this lab, the number of internal poles is 48, so the minimum angular resolution would be: $\frac{360}{2 \times 48} = 3.75$.

This resolution is for half-step configuration. The full-step mode exhibits twice the angular resolution of the half step mode.

Another characteristic difference between using half-step increments and full-step increments to make the rotor move is the torque force capability. The motor will exhibit a greater torque force when full-step codes are used because there is twice as many energized stator poles applying magnetic attraction on the rotor than when half-step codes are used.

Finally, to drive a stepper motor, a signal conditioning stage is needed. Since the output current of the DAQ used (the PCI 6024E) is too low to drive the motor, a current amplifier (line driver) will be connected between the DAQ digital output and the motor wires. An example is given later using the LM18293 line driver chip, but other current driving hardware can be used such as using four BJT transistors in open-collector mode as shown in Figure 3.

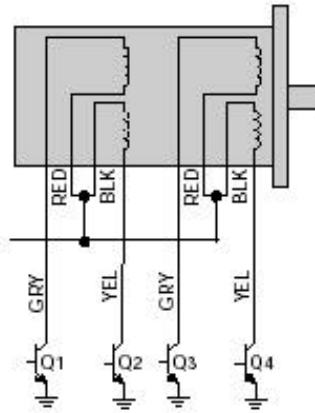


Figure 3 - Stepper Motor with BJT current driver

(Source: <http://www.allegromicro.com/techpub2/airpax/smh29.pdf>)

Pre-Lab Preparation

Read this experiment before coming to the lab. You should understand the operation of the stepper motor, and know the difference between full step and half step modes.

Bring the following to lab with you:

- This experiment.
- Your lab notebook and pencil.
- 2 virus-free formatted 3.5-inch floppy disks (always make a backup copy of your code on the second disk).

Answer the following questions in the datasheet provided at the end of this experiment *before* coming to the lab. Remember to include these answers in your lab report too. Some of these questions need experimentation to find the answer.

13. What is the logic 1 output voltage on the 6024E card? What is the logic 0 output voltage? What is the lowest input voltage to produce a logic 1 input? And the highest input voltage to produce a logic 0 input? Compare your answers to TTL and CMOS standards.
14. How do you produce a tri-state output? What is the voltage level?
15. How fast can a single digital output on the 6024 be changed?
16. Which VI is the most suitable for the Digital Control of the stepper motor (Functions Menu → Data Acquisition → Digital I/O)?
17. How will you store the last position of the motor in the LabVIEW VI? How will you initialize this value?

18. What is the difference between a Digital Port and a Digital Line in the Digital I/O VIs?
19. If the consecutive positions of a Stepper Motor in the clockwise directions are numbered 0, 1, 2, 3...7, obtain the binary code given to the stepper motor corresponding to each position.

Workstation Details

Your workstation should have the following items:

- Computer with National Instruments LabVIEW software
- National Instruments DAQ board (inside the computer)
- National Instruments DAQ board terminal block
- Stepper motor (48 internal poles with 4 coils and 5 terminals)
- LM 18293 push-pull quad line driver (or an alternative current driver hardware).
- Prototype board and jumper wires

Lab Procedure

1. Start a new VI in LabVIEW. You should then add the following controls and indicators to the front panel:
 - j. Half-Stepping/Full Stepping-Mode Switch: This switch controls whether the motor rotates in increments of full steps or half steps.
 - k. Clockwise/Counter-Clockwise-Rotation Switch: This control is used to alternate between the two directions of rotation.
 - l. Speed-of-Rotation Digital Control: A knob to control the speed of the motor in steps/second. This function varies the motor speed by controlling how fast the actual binary array is fed to the motor. In other words, it controls the time, in “milliseconds of wait,” that lapses between the outputs of each binary array that drives the stepper motor.
 - m. Number of Steps Digital Control: By using this function the user can specify the maximum number of steps the motor will make — whether they are full-steps, or half-steps. This will be the number of iterations to the main loop in the VI code.
 - n. Power Switch: This is the main On/Off switch for the VI.

- o. Current Position indicator: This is a gauge-like indicator that shows the current position of the motor in degrees, assuming it started at 0 degrees. This indicator should reflect the direction of rotation of the motor and the amount of degrees each step moves.
 - p. Current Output Code: This is an indicator with 4-leds array showing the current output code to the motor in binary.
2. After adding those controls, start wiring the VI diagrams with the controls to implement the functionality of the stepper motor control. You might find the following VI components useful in your code:
- g. A main loop, best implemented as a For Loop structure. The number of iterations is the input from the number of steps control on the front panel.
 - h. A 1-dimensional array (lookup table) of control codes. Since the motor used has 4 coils, there will be 8 different codes for a half-step configuration. We will use one array only (the half-step codes arranged in clockwise order) and obtain the values from this array using an index. The index itself depends on the step mode and the direction of rotation. The codes in the array should be written in binary (make sure to right click on the array and set the Format Precision to Binary).
 - i. Two Select VIs (from Functions → Comparison). These will be used with the two switches (step mode and rotation direction). The Select VI wired to the Step Mode switch will select a 1 for half-step and 2 for full step. This is equivalent to skipping every other code in the array when going in full steps. Similarly, the Select wired to the direction of rotation switch will select a 1 for a clockwise rotation and a -1 for a counterclockwise rotation. This is equivalent to going forward or backward in the array elements.
 - j. A Multiplication VI to multiply the outputs of the two Selects described in “c” above. There will be 4 different combinations for the output of the multiplier VI, specifying the increment/decrement to the index of the array:
 - 1 = half-step clockwise
 - 2 = full-step clockwise
 - -1 = half-step counterclockwise
 - -2 = full-step counterclockwise
 - k. An Addition VI to add the increment/decrement obtained in “d” with the previous value stored in a shift register. The output of

the Addition VI feeds the input the shift register. This way, you will keep track of the current value of the index to traverse the lookup table. Make sure the shift register output (to the left) is initialized to zero (attach it to a zero constant outside the loop)

- l. A Quotient and Remainder VI. This will be used to wrap the index over the 8 elements of the array. You will use it to divide the output of the adder in “e” with a constant integer (8), and take the Remainder as the actual index to the array.
 - m. A Write To Digital Port VI to send the value obtained from the array to the digital port of the 6024E. Configure the output so that you send to Port 0 (PA) of device 1 (6024E). The default port length is 8 but we are using only 4 bits (the least significant bits of the word).
 - n. A Stop VI (Application Control → Stop) wired to the power switch and placed inside the main loop.
3. These are the basic elements that you will need to have a working program. You will need more VI components to control the speed of rotation (milliseconds wait) and to show the current motor position. Remember that one whole rotation is 360 degrees, and that the motor goes 3.75 degrees for each half step and 7.5 degrees for each full step. If you have any questions about writing the remaining code, ask your lab instructor.
 4. After finishing your code save it to the floppy disks. Next, you need to connect the DAQ digital output pins with the motor and the current driver IC. Follow the connections in the diagram in Figure 4.
 5. Test your code and connections by moving the motor as follows:
 - Full step clockwise
 - Full step counter clockwise
 - Half step clockwise
 - Half step counterclockwise
 - All the above combinations with varying speed
 - Make the observation necessary to characterize the loss of torque observation when using half step vs. full step.

Note the angle of rotation (the current position) while you are experimenting with the possible combinations above.

Lab Report

Your report must be typed and of professional quality. You should include the following in your report:

8. All schematics, which must be computer-generated. Components must be labeled with circuit references and values. The schematics must be fully compliant with standard engineering practices for circuit depiction.
9. Front panel and block diagram of all VI's and Sub VI's that you wrote or used.
10. A Theory section describing the theory behind the concepts used in this lab.
11. An implementation section describing the details of your VI code and the logic flow.
12. The test results and observations.
13. The answers to the Pre-lab questions.
14. Selected sections of the datasheet of the 6024E, the stepper motor and/or the LM 18293.

Data Sheet

1. Answer :

.....

.....

.....

.....

.....

2. Answer :

3. Answer :

.....

4. Answer :

.....

5. Answer :
.....

6. Answer :
.....

7. Answer :

Position	A1	A2	B1	B2
0				
1				
2				
3				
4				
5				
6				
7				

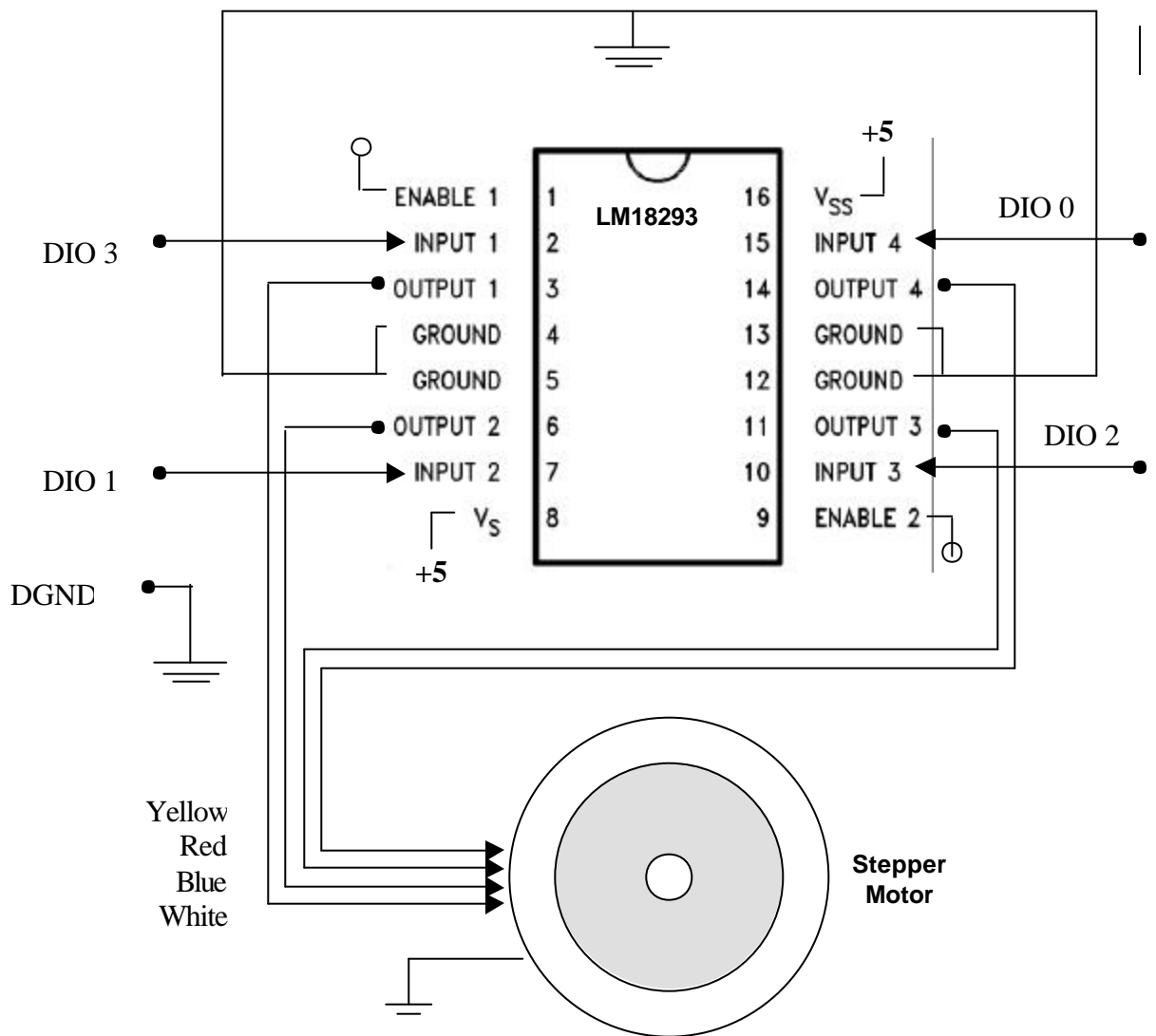


Figure 4 – Connection Schematic