Device Driver Development of LSM Using General Purpose PCI I/O Board

Hyun-Joong Kim*, Sang-min Lee**, Woon-Chul Ham***

* Division of Electronics and Information Engineering, Chonbuk National University, Korea
(Tel : +82-63-270-2399; E-mail: horuseye@hotmail.com)
** Division of Electronics and Information Engineering, Chonbuk National University, Korea
(Tel : +82-63-270-2399; E-mail: rechade7@hotmail.com)
*** Division of Electronics and Information Engineering, Chonbuk National University, Korea
(Tel : +82-63-270-2399; E-mail: : wcham@moak.chonbuk.ac.kr)

Abstract: In this paper, position and speed control algorithm of LSM (Linear Stepping Motor) using general-purpose PCI I/O board is discussed. The main purpose of this paper is to show that LSM controller can be established on the non real time operating system such as Microsoft Win2000 under the assumption that thread priority strategy is well designed. We can guarantee sampling interval less than 5msec based on the Pentium III microprocessor. Therefore this kind of LSM controller development environment makes shorten the prior research period needed to verify the validness of the proposed control strategy. We also introduce the tool of the real-time windows target system of matlab, which also makes shorten the prior research period. The main focus of this paper is on developing general purpose NT device driver which can drive the general purpose PCI board and applying it for implementing the hardware interface for 2- axis linear stepping motor control. From the experimental results show that the developed LSM controller guarantee 2 micrometer resolution in position control with 10cm/sec moving speed

Keywords: Linear Stepping Motor (LSM), WDM, real-time windows target system, device driver, PCI

1. INTRODUCTION

Recently the importance and application examples of a linear stepping motor are increasing in industry field. A linear motor is a direct drive motor that has good performance in terms of accuracy, velocity and acceleration compared with the conventional linear motion system consisting of rotary motor, gear and ball screw [1][2]. The X-Y axis motion is based upon the Sawyer Principle, which says essentially that a rotary stepper motor may be laid out flat, and controlled by interacting flux fields (magnetism). When this motor is laid out flat it is called a "linear motor", because it will then travel in a straight line. The motor is a hybrid stepper motor, which means that it contains both permanent magnets and electromagnets. [3] The power comes from the permanent magnets, and the electromagnets are used to steer this power to create lateral torque. Called "linear motor", "X-Y motor", or most technically, the "forcer", it is made of permanent magnets and grooved electromagnets bonded to an aluminum shell. Each electromagnet assembly consists of two poles and a coil. The drawing below shows a simplified forcer. [4]

It rides on a sheet of air over a metal plate called a "platen", which has a waffle pattern etched into its surface. The grooves of both the forcer and the platen are filled in with epoxy, then ground and lapped smooth so that both surfaces are mechanically flat.

The purpose here is to explore a little further into control of linear stepping motor and device driver development of general purpose PCI I/O board. It is because I believe that need industrial that I have written this study. The purpose here is to explore a little further into PC based motor driver development and precise control of a linear stepping motor. Used in windows 2000 DDK development environment for general-purpose device driver development of fitness position and speed control. We also introduce the tool of the real-time windows target system of matlab, which also makes shorten the prior research period. We confirmed through the experiment. First of all, we have to inquire into movement principle of linear stepping motor. Secondly, we try to discuss and about a structure of WDM and Real-time windows target system. Only in the Final place, we will describe research about Development of general purpose PCI I/O board.

This work was supported in part by the Mechatronics Research Center of Chonbuk National University

2. PRINCIPLE OF THE LSM AND MICROSTEPPING

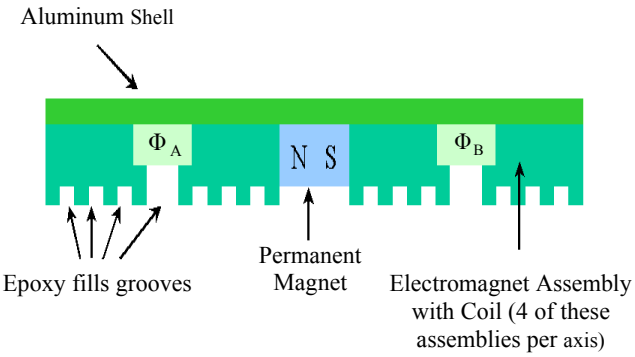

Fig. 1 Simplified Forcer Assembly

First of all, we have to inquire into principle of the LSM. It rides on a sheet of air over a metal plate called a "platen", which has a waffle pattern etched into its surface. The grooves of both the forcer and platen are filled in with epoxy, then ground and lapped smooth so that both surface are mechanically flat.

This is so that the forcer and platen may be separated by a film of air called an "air bearing", which provides friction-free movement (thus, no physical wear on the forcer or platen). This air film is 0.4-0.9 mils thick. Although flat mechanically, the grooves provide a surface that is not flat magnetically, causing the magnetic flux from the forcer to choose certain teeth on the platen as, the preferred pathway on the journey from the forcer's North pole to its South pole.

The forcer grooves are exactly the same pitch as the grooves of the platen, but are mechanically offset so that only one set of forcer grooves may be directly over the platen

grooves at one time. Fig. 2 shows these grooves from the side. The lines of force (flux) of the permanent magnet channel down through the poles of the phase A electromagnet, jump the air gap, run through the platen, and by again jumping the air gap, return through the poles of the phase B electromagnet.

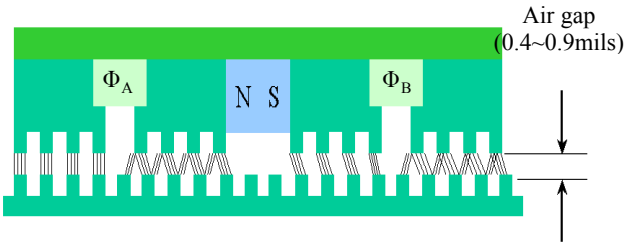

Fig. 2 Air Bearing Gap

Each pole would like to be directly over a platen tooth, but the spacing of the forcer poles makes it impossible for all forcer poles to line up with platen teeth at the same time.
Since the permanent magnet flux has divided equally among the poles, none are stronger than their neighbor, and a compromise state is reached.
This compromise position is called the "blank" state.

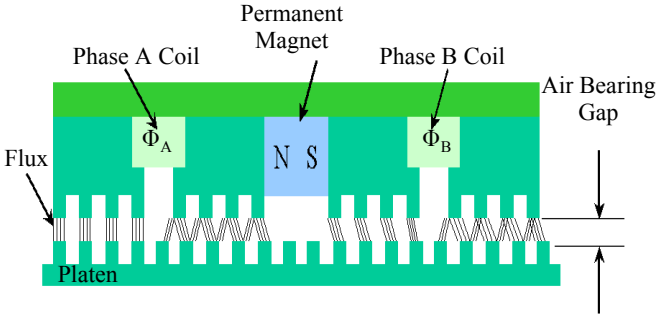

Fig. 3 Forcer in “Blank” State

The windings of the electromagnets create electromagnetic fields when energized, and the resulting polarities will either aid or oppose the polarity of the permanent magnet. This causes the flux, or lines of force, of the permanent magnet to be diverted from one pole to another pole.

Because of this diverting effect, the electromagnets are used as "steering magnets" for the permanent magnet's power.
Fig. 4 shows the flux diverted to Pole 1 when Phase A is energized and Phase B is not. By energizing the Phase A Coil, the flux is concentrated into Pole 1, which causes Pole 1 to line up with the nearest platen teeth.

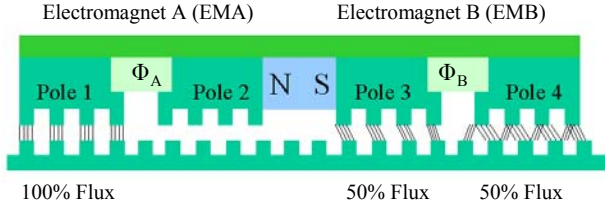

Fig. 4 Phase A Coil Energized; Pole 1 Aligns with Platen Teeth

It is the electromagnetic field of the polarized Phase A Coil that drives the permanent magnet flux away from Pole 2 and into Pole 1. This increase in flux at Pole 1 results in a greater attractive force.

Because we have not energized the Phase B Coil, it creates no field, and thus causes no diversion of flux n the B assembly. The flux remains divided equally in the ϕB assembly, and creates no lateral torque to oppose the power of Pole 1.

Figure 5 shows ϕB now energized, and ϕA current stopped. The induced polarities of poles 3 and 4 are now different, and the flux is diverted to pole 3. Pole 3 will pull the forcer to the nearest platen teeth, which are to the right.
Meanwhile, the flux passing through ϕA has divided equally between poles 1 and 2, so neither pole may oppose the torque developed by pole 3.

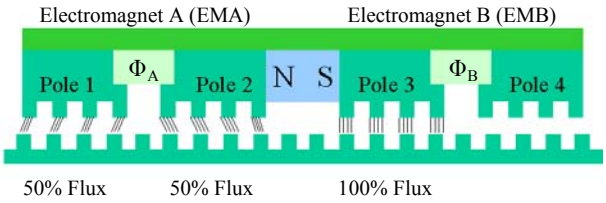

Fig. 5 Phase B Coil Energized; Pole 3 Aligns with Platen Teeth

Figure 6 continues the rightward movement by now diverting the flux to pole 2, while canceling at poles 3 and 4.

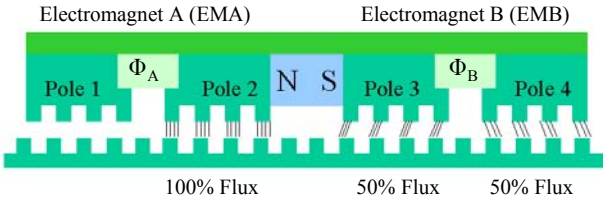

Fig. 6 Phase A Coil Energized Reverse Polarity; Pole 2 Aligns with Platen Teeth

Fig. 7 completes one full cycle by assigning the major flux density to pole 4, which finds the closest platen teeth lying just to the right.

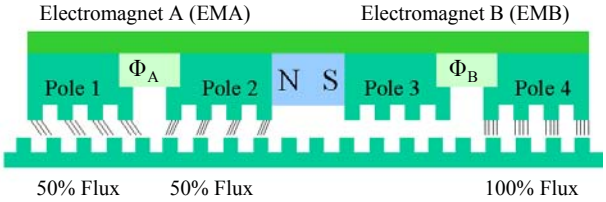

Fig. 7 Phase B Coil Energized Reverse Polarity; Pole 4 Aligns with Platen Teeth

These four individual figures show one full motor cycle according to different currents at different times.
Each figure shows the motor position when one of the drive waves is at peak - Phase A peaked high, then Phase B peaked high, then Phase A peaked low, and finally Phase B peaked low. Because the A and B phases are 90 out of phase, whenever one of them is at peak, the other is crossing the zero volt line, and thus is de-energized at that time.

The following page illustrates the full cycle - note that the teeth of Pole 1 will move exactly one tooth on the platen (20 mils). [5]

Microstepping is a way of moving the stator flux of a

stepper more smoothly than in full or half step drive modes.

This results in less vibration, and makes noiseless stepping possible down to 0 Hz. It also makes smaller step angles and better positioning possible.

Theoretically it is possible to use non-integer fractions of a full-step, but this is often impractical. A stepper motor is a synchronous electrical motor. This means that the rotor's stable stop position is in synchronization with the stator flux. The rotor is made to rotate by rotating the stator flux, thus making the rotor move towards the new stable stop position. The torque (T) developed by the motor is a function of the holding torque (T_H) and the distance between the stator flux (f_s) and the rotor position (f_r).

$$T = T_H \cdot \sin(f_s - f_r) \quad (1)$$

Where f_s and f_r are given in electrical degrees.

The relationship between electrical and mechanical angles is given by the formula:

$$f_{el} = (n/4) \cdot f_{mech} \quad (2)$$

where n is the number of full-steps per revolution.

When a stepper is driven in full-step and halfstep modes the stator flux is rotated 90 and 45 electrical degrees, respectively every step of the motor. From the formula above we see that the motor develops a pulsing torque. The reason for this is that $f_s - f_r$ is not constant in time due to the discontinuous motion of f_s .

In many applications Microstepping can increase system performance, and lower system complexity and cost, compared to full- and half-step driving techniques. Microstepping can be used to solve noise and resonance problems, and to increase step accuracy and resolution. [6]

3. SYSTEM OF LINEAR STEPPING MOTOR DRIVER

In this section, the point is structure of the controller and real-time windows target system.

3.1 Structure of the Controller

The purpose here is to explore a little further into PC BASED Motor driver development and Precise Control of a Linear Stepping Motor (LSM) The motor control Use General Purpose PCI I /O Board. Motor Driver is used LMD18245.

The main focus of this paper is on developing general purpose NT device driver which can drive the general purpose PCI board and applying it for implementing the hardware interface for 2-axis linear stepping motor control. From the experimental results show that the developed LSM controller guarantee 2 micrometer resolution in position control with 10cm/sec moving speed. Fig. 8 shows Schematic diagram of experimental apparatus

3.2 Real-time Windows Target System

The main purpose of this paper is to show that LSM controller can be established on the non real time operating system such as Microsoft Win2000 under the assumption that thread priority strategy is well designed.

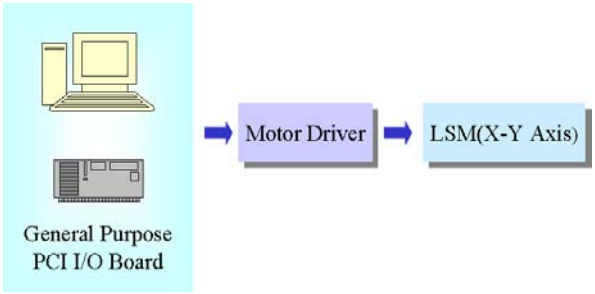

Fig. 8 Schematic diagram of experimental apparatus

We can guarantee sampling interval less than 5msec based on the Pentium III microprocessor. Therefore this kind of LSM controller development environment makes shorten the prior research period needed to verify the validness of the proposed control strategy. We also introduce the tool of the real-time windows target system of matlab, which also makes shorten the prior research period. The utility of the Matlab, we can control easily I/O Board. Real-Time Windows Target is a PC solution for prototyping and testing real-time systems. It is an environment where you use a single computer as a host and target. Use Real-time windows System in the Matlab, First must register a Real-time kernel at the pc. Real-Time Windows Target uses a small real-time kernel to ensure the real-time application runs in real time. The real-time kernel runs at CPU ring zero (privileged or kernel mode) and uses the built-in PC clock as its primary source of time. In the Second place, Created from the generated C code using either a Microsoft Visual C/C++ compiler or a Watcom C/C++ compiler. In Real-time windows target system, the Analog Input, Analog Output, Digital Input, and Digital Output blocks provide an interface to your physical I/O boards and your real-time application. They ensure that the C code generated with Real-Time Workshop correctly maps block diagram signals to the appropriate I/O channels. [7]

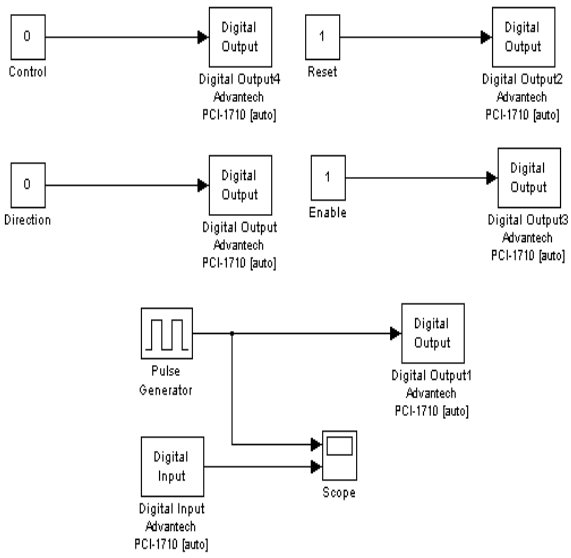

Fig. 9 Control Structure of Real-time Windows target system

Fig. 9 shows used real-time windows target system in Matlab. Used Digital input and output of the I/O board and input clock signal approved the square wave. The experiment demo used I/O Board PCI-1170. Currently, use PLX-9054 Chip are doing the General Purpose PCI I/O Board development.

4. DESIGN OF WDM DEVICE DRIVER

Drivers are source code compatible across all platforms, and in some cases binary compatible. Drivers support Plug and Play, Power Management, and Windows Management Instrumentation. WDM is a layered architecture consisting of a "stack" of drivers supporting single hardware devices. MS supplies many drivers for standard Windows system devices Sometimes resulting in a Class Driver / Miniport architecture based on WDM.

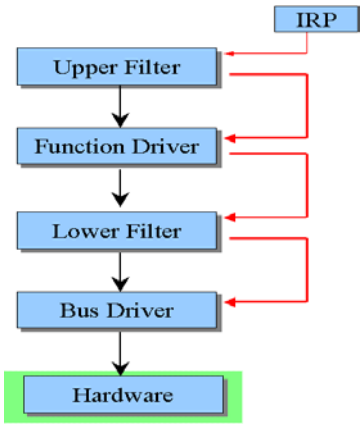

Fig. 10. Layered Architecture

Bus drivers, which drive an I/O bus and provide per--slot functionality that is device-independent. Function drivers, which does know how to control a device's main features is layered above the bus driver. Filter drivers, which filter I/O requests for a device, a class of devices, or a bus. The bus driver object at the bottom of the device stack is called the Physical Device Object (PDO). Each WDM driver must make its Functional Device Object (FDO) for each device. Finally, you must remember the next device down the stack. This is so you know where to pass a request when you need to send it for processing down the stack of drivers. Our AddDevice routine calls IoRegisterDeviceInterface to register the link between WDM_GUID and our FDO. It then has to enable it using IoSetDeviceInterfaceState. When our device is removed, the WdmPnp routine disables the device interface.

IoRegisterDeviceInterface actually makes a symbolic link to our device. The actual link name is a long string that includes our GUID. Win32 programs like our TestWdm use various SetupDiGetClassDevs functions (such as SetupDiGet ClassDevs) to find all devices, which support a particular GUID. Eventually it can get the symbolic link name, which it can pass to CreateFile to open a handle to our device.

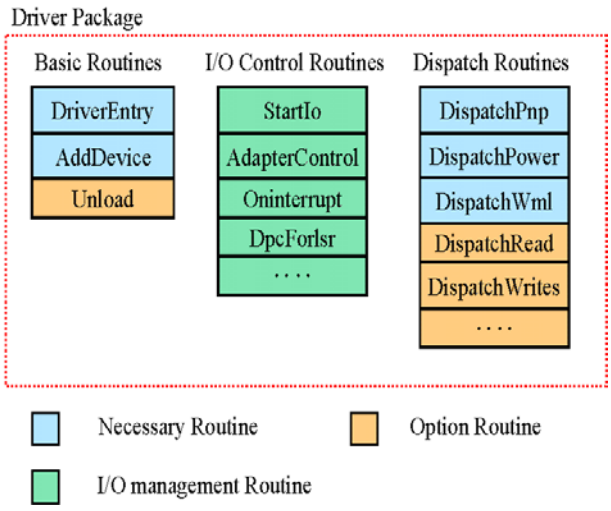

Fig.11 WDM Driver Routines

Five IOCTLs can be used with the WDMIO and PHDIO driver, as listed in Table 1. An IOCTL is an operation that is run using the DeviceIoControl Win32 function. An IOCTL is identified by an IOCTL code, and can have both input and output parameters

Table 1. WDMIO and PHDIO IOCTL Codes

IOCTL	Input	Output	Description
IOCTL_PHDIO_RUN_CMOS	Yes	No	Run the passed commands
IOCTL_PHDIO_CMOS_FOR_READ	Yes	No	Store the commands to read a byte and store inthe read buffer
IOCTL_PHDIO_CMOS_FOR_READ_START	Yes	No	Store the commands that start the read process
IOCTL_PHDIO_FOR_WRITE	Yes	No	store the commands to output a byte from the write buffer
IOCTL_PHDIO_GET_RW_RESURTS	No	Yes	Get the command results of the last read or write operation

The first IOCTL, IOCTL_PHDIO_RUN_CMDS, is used to run PHDIO commands straightaway. The three following IOCTLs store the commands that are used later in the processing of read and write requests. Finally, IOCTL _PHDIO_GET_RESULTS retrieves the command results and output from the last read or write.

A win32 program passes a block of commands in an IOCTL input buffer. These Commands are either run straightaway or run one or more times during the processing of read or write request. A Command is a singles byte, followed by one or more parameter bytes. Table 2 lists all commands, their parameters, and their output.

Table 2 shows all the available WDMIO and PHDIO commands. The last two commands (PHDIO_WRITE_NEXT and PHDIO _READ _NEXT) can only be used in commands that handle interrupt -driven writes and reads. All commands and parameters are bytes. In all cases, the reg parameter is the offset into the I/O port address range.

Also fast and can do the control easily through oneself WDM Device Driver Development. [8]

Table 2. WDMIO and PHDIO Commands

Command	Input parameters	Output	Description
PHDIO_WRITE	reg,Value		Write value to a register
PHDIO_READ	reg	Value	Read value from a register
PHDIO_DELAY	delay		Delay for given microseconds. Delay must be 60μs or less
PHDIO_WRITES	reg,count,Values,delay		Write values to same register with delay (<=60μs)
PHDIO_READS	reg,count,delay	Values	Read values from same register with delay (<=60μs)
PHDIO_OR	reg,Value		Read register, OR with value and write back. Use to set bit(s)
PHDIO_AND	reg,Value		Read register, AND with value and write back. Use to clear bit(s)
PHDIO_XOR	reg,Value		Read register, XOR with value and write back. Use to toggle bit(s)
PHDIO_IRQ_CONNECT	reg,mask,Value		Connect to interrupt
PHDIO_TIMEOUT	seconds		Specify time-out for reads and writes
PHDIO_WRITE_NEXT	reg		Write next value from write buffer
PHDIO_READ_NEXT	reg		Store next value in read buffer

5. Conclusion

In this paper, the point I want to make is that Device Driver development and linear stepping motor control. The purpose here is to explore a little further into PC based motor driver development and precise control of a linear stepping motor. This research is the PC-based immediately control. The control of motor in the PC based used general purpose PCI I/O board. We can get the charge with the frugality of much time, the convenience of an application program development. Also fast and can do the control easily through oneself device driver development the purpose of this research is one among a semiconductor test equipment began for efficient utility of probe system. It will become the aid at an automation industry development if this research is completed.

REFERENCES

[1] Sed A, Nasar, I. Boldea, “Linear Electric Motors Theory, Design, and Practical Applications”, Prentice-Hall. Inc. USA.

[2] Gerco. Otton, Theo J.A. de Vries, Job van Amerongen, drian M. Rankers and Erik W. Gaal, “ Linear Motor, Motion Control Using a Learning Feedforward Controller,” *IEEE/ASME Trans. on Mechatronics,* Vol. 2, NO. 3, pp.179-187, 1997.

[3] C. S. Moon, J. I. Park, K. D. Lee, S. G. Lee, and J. H. Lee, “Design of position controller of linear pulse motor using neural network,” *Proceeding of American Control Conference*, pp. 990-991, 1998

[4] Wong, F.Y.; Schulze-Lauen, H.; Youcef-Toumi, K., “Modelling and digital servo control of a two-axis linear motor” *American Control Conference,* Vol. 5, pp. 3659 –3663, 1995

[5] Linear motor theory of Electrogal

[6] Microstepping theory of New Japen Radio Co., Ltd

[7] Matlab “Real-Time Windows Target For Use with Real-Time Workshop”, User’s Guide Version 2

[8] Chris Cant, “Writing Windows WDM Device Drivers” R&D Books Miller Freeman, Inc.