JOURNAL OF ENGINEERING annals OF FACULTY OF ENGINEERING HUNEDOARA
Tome V (year 2007), Fascicole 3, (ISSN 1584 – 2673)

LABVIEW USING TO DRIVING A BI-DIRECTIONAL MOTOR

HAMMES Ana Daniela

UNIVERSITY POLITEHNICA TIMISOARA
FACULTY OF ENGINEERING HUNEDOARA

ABSTRACT:
This article proposes to realize:

- explain how to use the graphical programming language LabVIEW to write to a specific register address and to build a VI;
- how to connect a bi-directional motor to the PC, using the parallel port;
- how to build the interface between PC and the motor;

KEYWORDS:
Parallel Port, LabVIEW, programming, bi-directional motor, register address

1. INTRODUCTION

The Parallel Port is the most commonly used I/O port for connecting the own circuits to PC. This port, in standard mod, will allow the input of 5 bits or the output of 12 bits. Using the semiconductor integrated circuit M54544L, we may directly driven a small size bi-directional motor rotating in both forward and we may read the number of revolution (rotation). The LabVIEW, integrated fully for communication with hardware, use us to write the program.

2. BI-DIRECTINAL MOTOR APLICATION

PC parallel port was never designed to provide any output power to devices connected to it (1). Was only designed for connecting printer to the PC, but in time everything change, parallel port have become a port where we can connect a external application. The application presented, figure 1, need 2 commands line (378h) and 1 read line (379h).

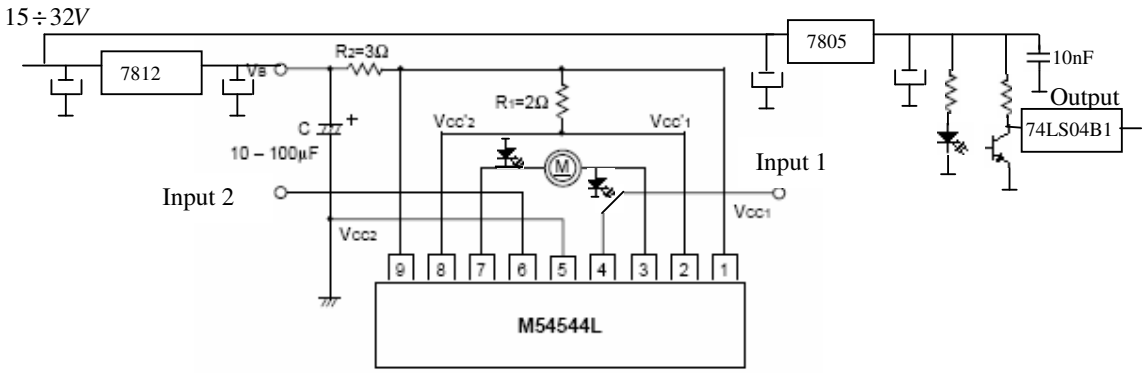

Figure 1. Application scheme

The pin configuration of semiconductor integrated circuit M54544L is presented in figure 2.

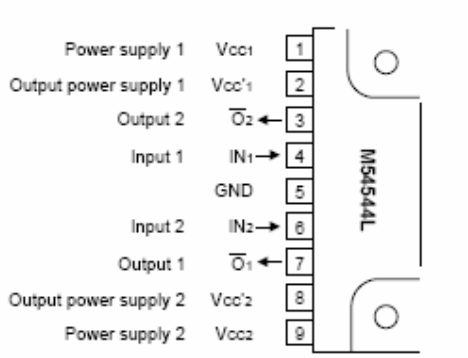

Figure 2. Pin configuration

When both inputs 1 and 2 are set to low-level, output 1 and 2 are set to "OFF". When input 1 is set to high-level and input 2 is set to low-level, output 1 is set to high-level and output 2 is set to low-level (forward rotation status). When input 1 is set to low-level and input 2 is set to high-level, output 1 is set to low-level and output 2 is set to high-level (reverse rotation). When both inputs 1 and 2 are set to high-level, bouth outputs 1 and 2 are set to low-level (brake status). (3) The logic truth table it is presented in table 1.

Table 1. Logic truth table

Input		Output		Remarks
IN_1	IN_2	$\overline{O_1}$	$\overline{O_2}$	
L	L	"OFF" state	"OFF" state	No operation of IC
H	L	H	L	ex Forward rotation
L	H	L	H	Reverse rotation
H	H	L	L	Brake

In the figure 3 it present a picture with the practical application, of the bi-directional motor.

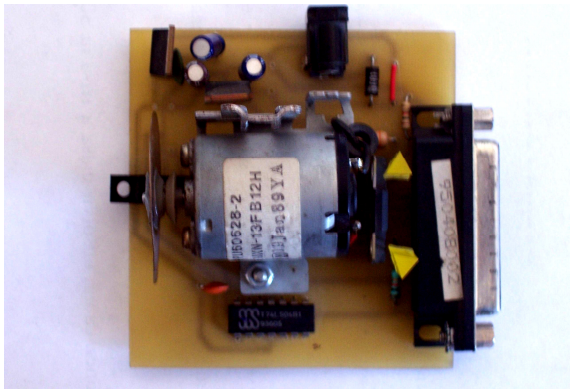

Figure 3. Practical aplication

3. LabVIEW PROGRAMING

LabVIEW is an entirely graphical language which looks somewhat like an electronic schematic diagram on the one hand and a 1950'a vintage style electronic instrument on the other-these are the concepts of the block diagram and the front panel. LabVIEW is hierarchical in that any virtual instrument that we design (any complete functional unit it called a virtual instrument and is almost always

referred to as a "VI") can be quickly convert in to a module which can be a sub-unit or another VI. This is entirely analogous to the concept of a procedure in a traditional procedure in traditional programming.

This language has two "faces":

- **the block diagram** - is almost the "back side" of the VI, practical is the program. It shows how the control and the indicators fit together as well licks the hidden modules where all the work gets done;
- **the front panel** – the face that the users sees practical is the user interface. It contains controls and indicators. By intelligent design of the front panel of a VI it is fairy simple to produce a simple clean design for the user.

LabVIEW uses terminology, icons, and ideas familiar to technicians, scientists, and engineers, and relies on graphical symbols rather than textual language to describe programming actions. LabVIEW is integrated fully for communication with hardware such as GPIB, VXI, RS-232, RS-485, and plug-in data acquisition boards. LabVIEW also has built-in libraries for using software standards such as TCP/IP Networking and ActiveX. In the figure 4 it is present a picture with the user interface.

Figure 4. User interface

For left rotation we have to send successive to the data port, address 378h, the value $00100000_{(2)}$. For right rotation we have to send successive to the data port the value $00001000_{(2)}$. These values result von the fact that LabVIEW consider the pin 9 as least significant bit. For the effective sending of this value at the parallel port we use the function OutPort.

A control is a front panel object that the user manipulates to interact with the VI. Simple examples of controls are buttons, slides, dials, and text boxes. You build the block diagram using the terminals from the front panel controls and indicators and the VIs, functions, and structures from the Functions palette. Each palette icon represents a sub palette, which contains VIs and functions you place on the block diagram (2).

As we add nodes to the block diagram, we wire them to each other and to the terminals from the front panel objects using the Wiring tool, found on the Tools palette. A complete block diagram appears similar to a flowchart.

This application use as principal structure:

- Sequence structure, onsists of one or more sub diagrams, or frames that execute sequentially. As an option, we can add sequence locals that allow we to pass information from one frame to subsequent frames by popping up on the edge of the structure.
- While loop structure, repeats the sub diagram inside it until the conditional terminal (an input terminal) receives a FALSE or TRUE Boolean value. The default behavior and appearance of the conditional terminal is Continue if True. When a conditional terminal is Continue if True, the While Loop repeats its sub diagram until a FALSE value is passed to the conditional terminal.

The block diagram it is present in figure 5.

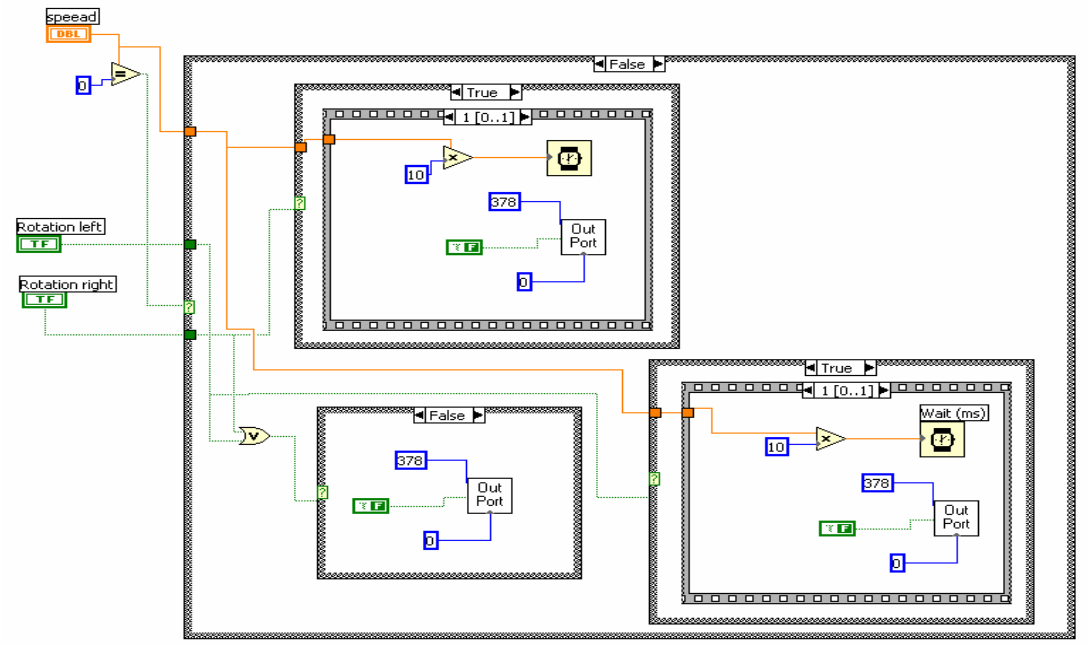

Figure 5. The block diagram

4. CONCLUSIONS

With this easy application we can understand the basic knowledge of the parallel port programming and to control an external application use the LabVIEW programming. We use the bi-directional motor example because in this way can see how to send and how to receive data to/from PC and how to build a LabVIEW application.

BIBLIOGRAPHY

(1.) H.P. Messmer, „The Indispensable PC Hardware Book". Fourt Edition Addison Wesley, 2000;
(2.) http://www.ni.com;
(3.) http://www. datasheetcatalog.com/catalog/p556080.shtml.