LABORATORI NAZIONALI DI FRASCATI
SIS – Pubblicazioni

LNF–04/11 (NT)
14 Giugno 2004

PICCA - A PIC DEMO BOARD FOR LABVIEW COURSES

Marco Cordelli[1], Agnese Martini[1], Luciano Trasatti[1]

[1)]INFN-Laboratori Nazionali di Frascati Via E. Fermi 40, I-00044 Frascati, Italy

Abstract

We have designed and built, with the help of Servizio di Progettazione Elettronica of LNF, a demo board using a PIC 16F876 microcontroller, several peripherals and an RS/232 link intended as a control link with a separate computer running a high level control language, i. e. LabVIEW, with the main purpose of setting up in our LabVIEW courses a working system for data acquisition /controls.

PACS.: 01.50.Ht; 01.50.Lc

1 Architecture

In the past few years we have held , both in the LNF and other INFN sections, several LabVIEW courses with the purpose of introducing new programmers to this language that is being increasingly used in our community. One of the most diffused criticisms from the students at the end of our courses was the lack of hardware to implement a real data acquisition system. Since the cost of commercial boards for this purpose was outside our possibilities, we decided to build our own.

We chose a multiprocessor architecture, which we have widely used in the past. A small peripheral board takes care of the actual hardware interfaces, while the main computer, through an RS/232 link, can access already decoded and formatted data. Using this system the main high level program has the possibility to concentrate on display, histograms and generally on the human interface problem.

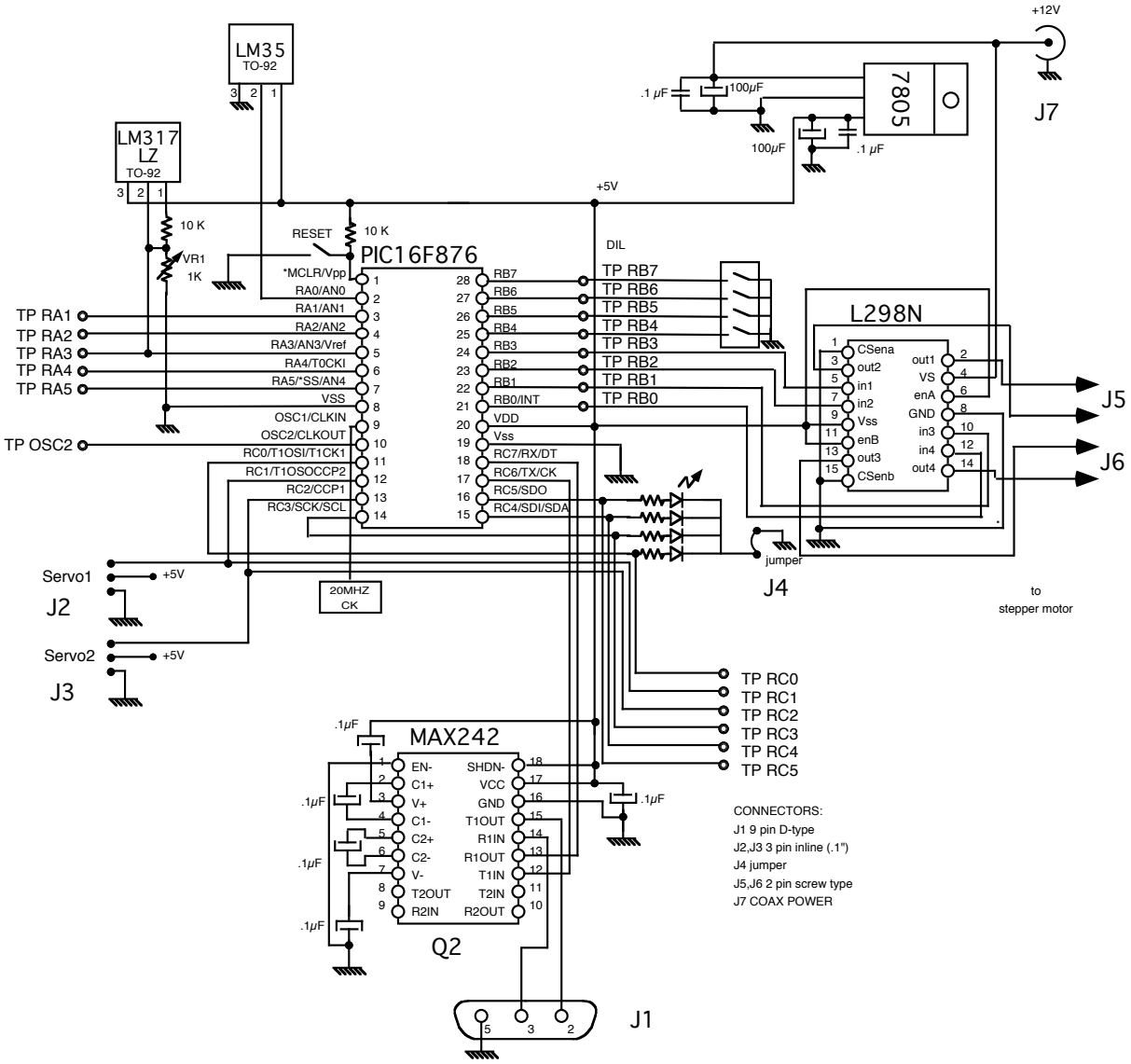

FIG. 1: PICCA circuit diagram.

We have used the board in the LabVIEW course held at LNF on June 7-11 2004, and the reesults have been highly encouraging, allowing the students to come to grips with the general

problems that occur in computer communications, like data translation, timing and sinchronization.

2 Hardware

PICCA has been designed as a multilayer pc board. The circuit diagram is shown in Fig. 1.

We have chosen a PIC microcontroller. These CPUs are very popular at LNF for their versatility, speed and low cost.

The implemented peripherals are:

- 10 bit A/D converter reading ambient temperature: the LM35 sensor gives an output of 0mV+10.0mV/°C starting from 2 °C, and the ADC converts the input in a range from 0 - 5 V.
- 4 bit digital input (switches)
- 4 bit digital output (4 LEDs, red, green, yellow, white)
- stepper motor driver
- servomotor driver
- a reset button is provided.

A pin grid array section has been provided for custom expansion, and all the CPU pins have been extended to the edge of this area. An external reference voltage driver for the A/D converters has been provided.

3 SOFTWARE

An assembler program has been written for the PIC microcontroller, to make it possible to control the features of the board through the RS/232 link.

The communication protocol has been kept as simple as possible.

All messages from PICCA end with a "?", to facilitate sinchronization.

All numbers, both input and output, are in hexadecimal format.

The PIC UART is set at 9600 baud, no parity, 1 stop bit, no flow control.

A list of the messages follows: messages from PICCA are shown as *italic*, commands from user as **bold**. Commands are not case sensitive.

On power up and after the execution of every command, PICCA sends this message:

PICCA is alive and well

A=DAQ, M=step, T=servo, D=dig I/O ?

3.1 Analog input: temperature

The user sends **A**

PICCA asks:

N meas(01-3F)HEX?

The user sends the number of measurements to be executed and averaged, as a two digit hexadecimal number. Range **01** to **3F.**

Picca answers:

nnnn r mmmm

PICCA is alive and well

A=DAQ, M=step, T=servo, D=dig I/O ?

where *nnnn* is the average of the measurements in hexadecimal format, r indicates the remainder *mmmm* of the averaging division, again in hexadecimal.

3.2 Stepper motor control:

The stepper motor must be connected to the four wire screw type connector J5 - J6. One of the two windings must be connected to the two left poles, the other to the two right poles. The motor is operated in half step mode, so one actual step of the motor counts as two steps in the program.

The user sends **M**

PICCA asks:

Steps/s(004D-0FA0,0000=reset)?

The user sends the required speed in steps per second as a 4 digit hexadecimal number. Sending **0000** will take you back to reset.

Picca asks:

N steps(004D-0FA0)Hex?

The user sends the required number of steps as a 4 digit hexadecimal number: **0001** to **FFFF**.

Picca asks:

Dir (L or R)?

The user sends **L** for left direction or **R** for right direction. The real direction is somewhat arbitrary because it depends on how the motor wires were connected. Inverting the wires of one of the windings changes the direction of rotation.

PICCA starts rotating the motor. When the required number of steps has been executed, PICCA sends the message:

Steps left 0000 HEX

PICCA is alive and well

A=DAQ, M=step, T=servo, D=dig I/O ?

If at any time during the motor run the user wants to stop it, he can send **S**; the rotation will stop immediately, and PICCA will send the message

Steps left nnnn HEX

PICCA is alive and well

A=DAQ, M=step, T=servo, D=dig I/O ?

where *nnnn* is the number of steps not executed.

3.3 Servomotor control

A servomotor is a motor coupled to a potentiometer, that can rotate a little more than 180 degrees, and assumes a position that depende on the duty cycle of the control signal sent continuously by the CPU. This control is handled in PICCA by an interrupt mechanism. This allows the user to position the servo and force it to remain in the chosen position while other actions are possible, for example, analog measurements.

The user sends **T**

PICCA asks:

dc (01E0-0620, 0000=off)?

The user sends the required position as a four digit hexadecimal number. Range **01E0** to **0620.**

Picca positions the servo and returns to reset, sending:

PICCA is alive and well

A=DAQ, M=step, T=servo, D=dig I/O ?

Note that the interrupt mechanism is now operating and the motor kept in the position required. If the user wants to stop the motor control, he must execute the sequence:

The user sends **T**

PICCA asks:

dc (01E0-0620,0000=off)?

The user sends **0000.**

Picca stops controlling the servo and returns to reset, sending:

PICCA is alive and well

A=DAQ, M=step, T=servo, D=dig I/O ?

3.4 Digital I/O

Picca can read the state of the four switches on the board and can write four bits to a register that lights the four LEDs (jumper J4 must be installed for the LEDs to light).

The user sends **D**

PICCA asks:

R=read, W=write ?

The user sends **R** .

Picca reads the position of the four switches, returns the value as a one digit hexadecimal number and returns to reset, sending:

0n

PICCA is alive and well

A=DAQ, M=step, T=servo, D=dig I/O ?

where *n* gives the four bits (*0-F*)

or

The user sends **W** .

PICCA asks:

Number to write(00-0F)HEX?

The user sends a two digit hexadecimal number, range **00-0F.**

Picca turns on and off the appropriate LEDs and returns to reset, sending:

PICCA is alive and well

A=DAQ, M=step, T=servo, D=dig I/O ?

4 ACKNOWLEDGEMENTS

We would like to thank V. Chiarella for continuing support and encouragement.

We would like to thank the LNF Servizio di progettazione elettronica, and in particular G. Paoluzzi and G. Papalino, for their constructive help.