A simple but efficient stepper motor controller

Build the controller
load the hex into a pic16F84A
use the PC tool to test it
by a.ratti

20th November 2008

This project uses a pic micro 16F84a to drive a stepper motor in an efficient way. The goal was to project a cheap and compact unit capable of total control of the motor. This goal has been achieved thanks to some rather interesting software solutions

The Schematics:

The project uses three common components to built the controller: 1 x pic16F84a ; 2 x ULN2803 which have been paralleled in such a way to use 4 transistors per coil (see figure 1 & 2 In this way, there is the possibility to drive coils with up to 2 Amps of current per coil with a rather compact unit) and 1 x 7805.
An additional component (Max 232) is optional in case you need to connect the board to a RS232 serial port. (one for a serial network)

The micro IO pins have been used as follow:

1	pin for serial Rx	(T9600)
1	pin for serial Tx	(OT9600)
4	outputs to drive the motor	
1	output for busy signal	(High = Busy)
1	output for heart beating led	
3	inputs to detect limit switches	(Low = normal High = Active)
1	input to detect master hold signal	(High = normal Low = Active)

THE LIMIT SWITCHES

There are three limit switches all normaly closed, two for safety the third for detecting the home position. (zero position)

If not used (not reccommandable) the relative connections must be shorted to ground, otherwise the controller will not respond having both upper and lower limits activated.

Lower limit switch and home switch must be placed in such a way that they will be intercepted when the motor travel CCW. Naturally the home should be placed at the zero position and the limit just before the end of the CCW mechanical travel of your system.

The upper limit switch must be placed in such a way that it will be intercepted when the motor travel CW and his position is just before the end of the CW machanical travel of your system.

The controller is activated using the serial Rx line with a special code composed of 4 bytes. Where:

1st	Byte = Device Address (default =10)
2nd	Byte = Command type
3rd	Byte = Value (Low Byte)
4th	Byte = Value (High Byte)

Every Rx will be answered via the Tx line with: "addreess"; "command code" and "Done" if succesful or "Failed" if unsuccesful.

Controllers can be paralled in a serial network with a different address for each controller, since the controllere "ADDRESS" is changeable via Rx command.

HOMING POSITION

Homing could be automatic at start up if proper configuration is activated or it can be activate manually using the following command code:

Command code 10   200   will activate CCW homing travel which will end only when the zero switch will open (or you open the jump)

Command code 10   210   will activate CW homing travel which will end only when the zero swicth will open (or you open the jump)

It is obviuos that when you have choosen the direction of your system you will need only one of these two commands. Choosing the wrong one, the system will travel till it will activate the upper limit switch and then stop. (If you have them installed)

Once you have tested the homing manually, with the command codes 200 or 210 and found it working satisfactorely, you may want to activate the homing automatically at startup. To do so you need to use the following command code:

Command Code   10   40   0 = (No AutoHoming)

Command Code   10   40   10 = (CW AutoHoming)

Command Code   10   40   20 = (CCW AutoHoming)

CON FIGURATION

With the term configuration we intend the possibility to change mode of operation from full-step to half-step.
By default the system is setup to works in half-step. This mode double the steps per turn and make the motor run smootly, but take into account that in this mode the motor looses some torque.
If you need to change the mode of operation then you should use the following command code:

Command Code   10   30   3 = (Full Step operation)

Command Code   10   30   7 = (Half Step operation)

MOTOR SPEED

The speed is set in "UNITS OF DELAY". What ! Let me explain: the original intention was to use Steps/second as the unit of speed, but since I run out of memory space (the micro is full as egg) then I had to cut something that could be done outside and this convertion is one of the several cuts I had to do.
As you probaly know the stepper turns because we energize the various coils with a proper sequence . There are 4 sequences for the full-step mode and 8 sequences for half-step mode. Moving from one sequence to the next the motor move one step the time delay between the two sequence define the speed. (longer is the delay slower will rotate the motor; shorter is the delay quicker will rotate the motor). So to control the speed we must send to the controller a value representing "UNITS of DELAY". This value can be obtained from the following formula:

Units of Delay = 200000/(STEPS/second)

Example:

Let assume you need to load a speed of 400 steps /second

200000/400 = 500 (Units of Delay)

500 is the value you will need to load into the controller to obtain a speed of 400 steps/sec.

Now if the motor have 400 steps per turn and we use half-step mode than the motor will rotate 360 degree with 800 steps, and it will make one turn in 2 seconds or 30 turns /min

Never works near the critical speed, the motor looses steps performing unprecise travel.
Higher the speed less is the motor torque. (critical when motor is fed with fixed voltage)

Anytime you need to change the speed of the motor you must use the following command code:

Command Code   10   90   Value (Units of Delay)

The Units of Delay value to Tx must be split into two bytes (low byte & high byte)

ACCELERATION – DECCELERATION RAMP
What we do is to start the motor at a lower speed and once is is moving accelerate it till reaching the setted speed, travel at that speed till destination less a little bit and then deccelarate to stop.
This is known as a trapezioidal travel and is the best way to deal with the system inertia.
By default the ramp is set at 800 steps (one motor turn). If you needed to can change this value you can do it by using the following commands code:

Command Code   10   80   Value (steps)

The Ramp value to Tx must be split into two bytes (low byte & high byte)

ROTATING THE MOTOR UNDER CONTROL

Once the motor is assembled in a working mechanical piece of equipment, motion could be different from rotation, so not knowing what the application will be, we will say that we want to move to a target point. To do that we must know in which direction move and how meny steps we must apply to the motor to reach the target point. To accomplish this task we have two commands:

Command Code   10   100   Value (Steps) – Move the motor CW for the number of steps
                                         given in value.

Command Code   10   110   Value (Steps) – Move the motor CCW for the number of steps
                                         given in value.

The motion value to Tx must be split into two bytes (low byte & high byte)

Once the above command is given the motor will turn for the number of steps given at the speed and ramp previously setted .(Trapezioiadal Travel)

In order to make life simpler, I have put together a small program in VB with all the command protocol already fixed in it. (see fig 1)
It will help you in testing the controller once you have built it.
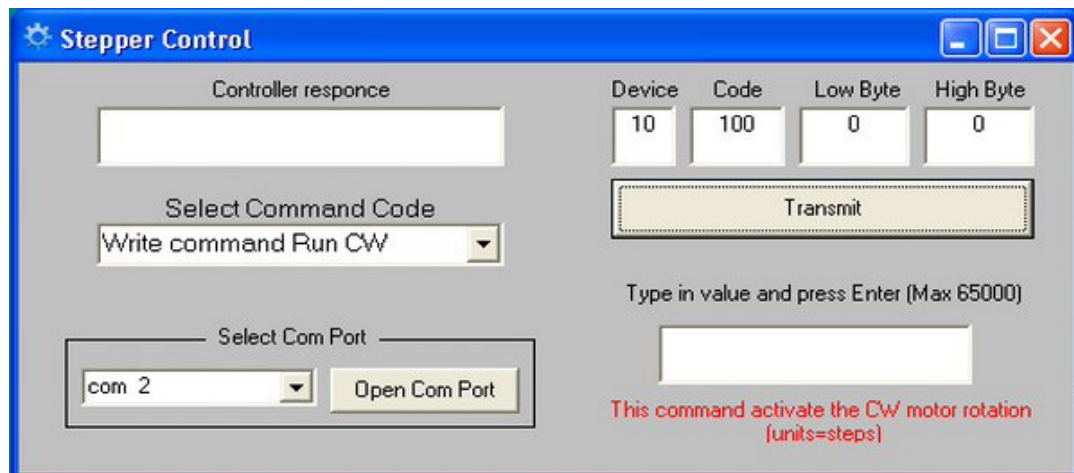

Figure 1

Copy the program into your PC select the serial port and open it. Enter the value (right side) and press enter to split value into two bytes.
Open the combo to select the command and press TRANSMIT to send command to the controller.

If you need to change address then you have to modify the default value manualy.

SCHEMATIC

The circuit is quite simple and it should not represent a problem in assemling it in a compact way.
Use a 16 MHz quartz or resonator.

Attention should be paid to the motor connections, in case your motor has different color wiring than you should adjust the wiring ,while you test the controller, to obtain the correct rotation in function of the comand you have sent (invert motor wiring if necessary)

It has already been mention that limit switches input must be grounded to have the motor working.
If homing is activated, it can be stopped only opening the home switch or removing the jump.

Great attention should be paid if you change the system address, because once activated the system will respond only to commands sent with the new address.

Don't use motors at 5 Volts otherwise the 7805 will not be able to supply the correct voltage to the micro.

Using a 12 motor , adjust PS voltage to reach correct nominal current feeding the motor (increasing or decreasing voltage till correct value is obtained)

When the motor is running the busy line goes high and remain high till the end of travel then return low.

If two o more motors are working together in a serial network , (every motor with his own address) than the Hold command line become very useful. If the Hold line is pulled down by the master, then every single address can be loaded with the proper speed and position, and when ready, master release the hold command line and all the motors (activated by the serial commands) will start at the same time. In this way the master can control linear interpolation of small XYZ table .
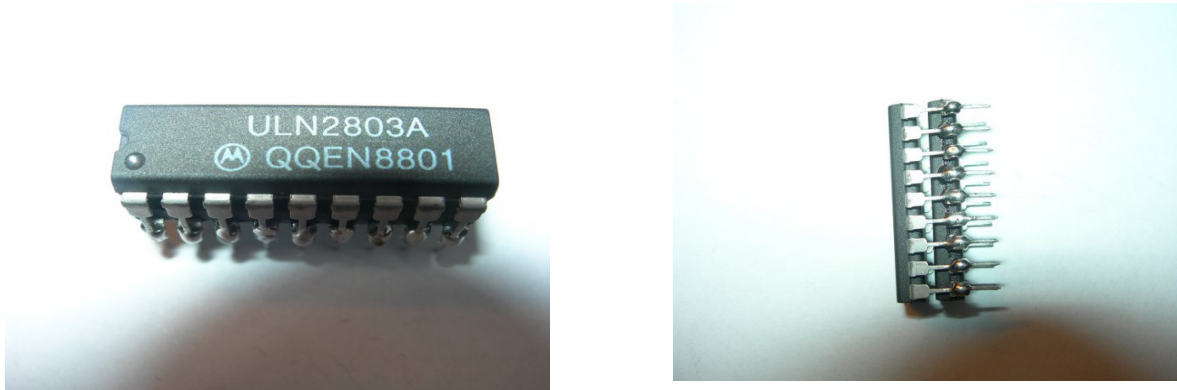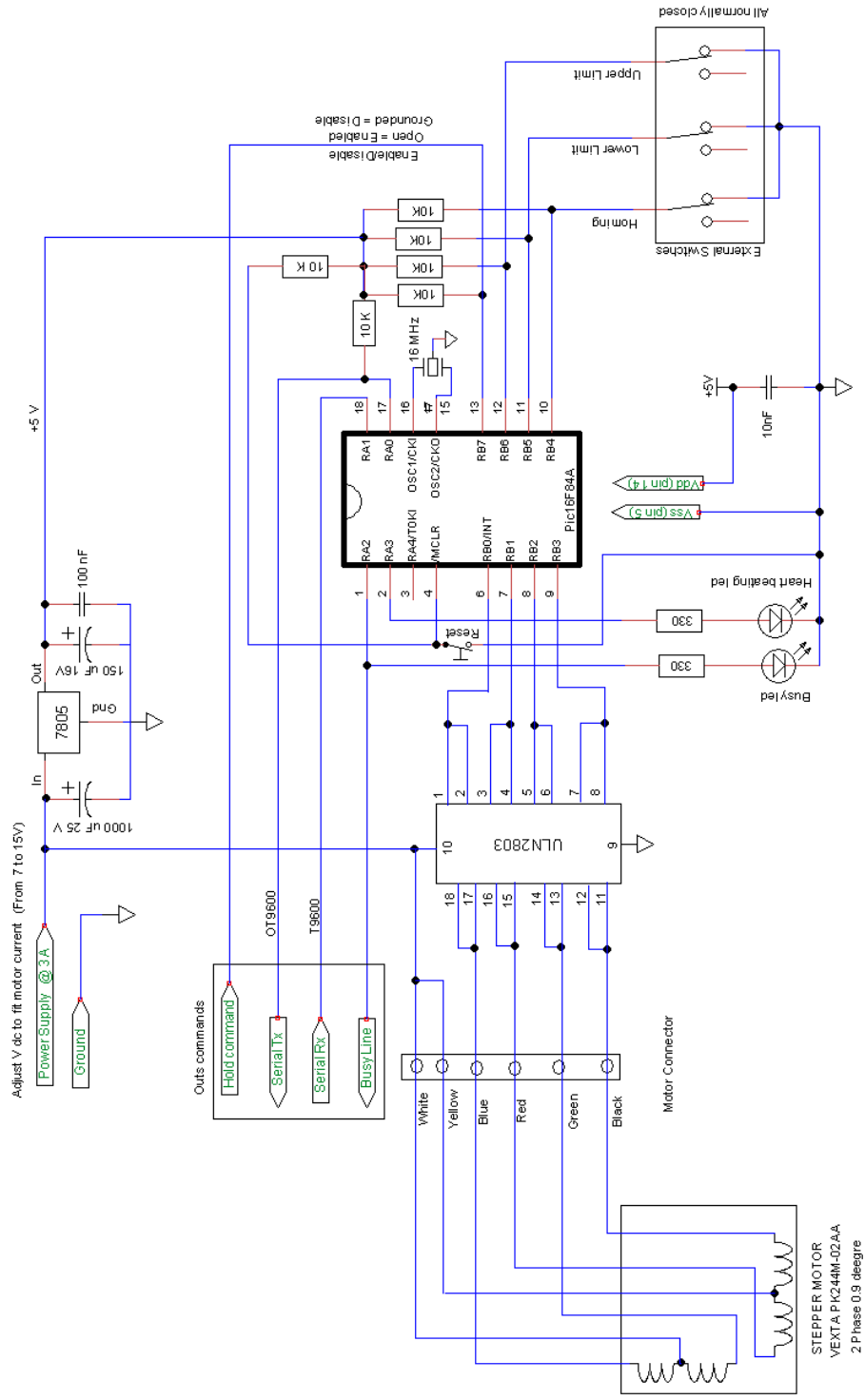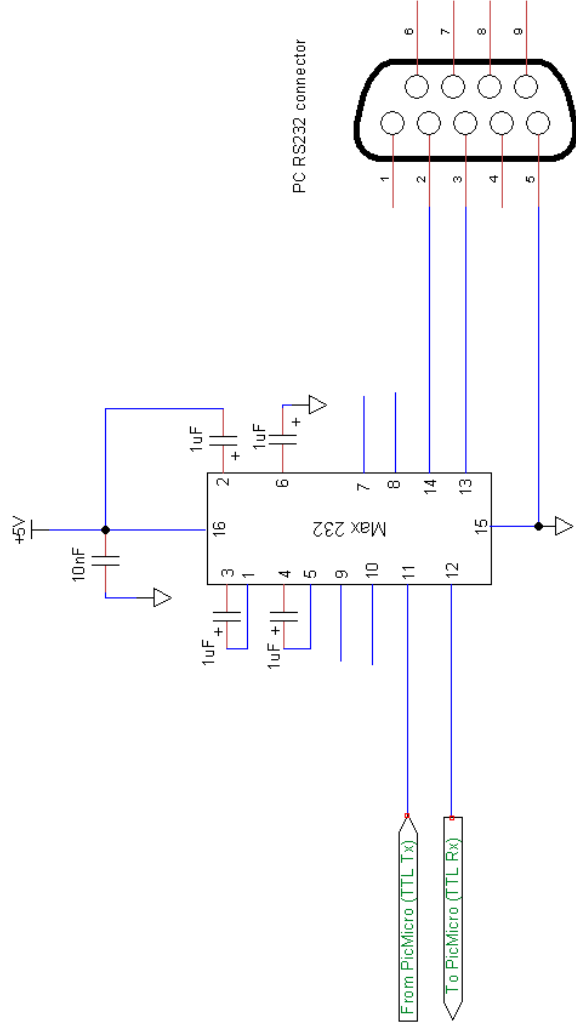

That' s All

Figure 2 & 3

Leave air gap between the two chips to improve thermal dissipation

Addendum: Using a voltage regulator (LM350; LM338 or LM317) wired as current limiter can improve the torque/speed of the motor. See last schematic added.

All normally closed

Upper Limit

Lower Limit

Homing

External Switches

Enable/Disable
Open = Enabled
Grounded = Disable

10K
10K
10K
10K
10 K
10 K

16 MHz

+5V

10nF

Vdd (pin 14)

Vss (pin 5)

Heart beating led

Busy led

330

330

RA1 RA0 OSC1/CKI OSC2/CKO RB7 RB6 RB5 RB4

18 17 16 15 13 12 11 10

RA2 RA3 RA4/TOKI /MCLR RB0/INT RB1 RB2 RB3

1 2 3 4 6 7 8 9

Pic16F84A

Reset

+5 V

100 nF

150 uF 16V

7805

Out

In

Gnd

1000 uF 25 V

Adjust V dc to fit motor current (From 7 to 15V)

Power Supply @ 3A

Ground

Outs commands

Hold command

Serial Tx

Serial Rx

BusyLine

OT9600

T9600

10 1 2 3 4 5 6 7 8

ULN2803

9

18 17 16 15 14 13 12 11

White
Yellow
Blue
Red
Green
Black

Motor Connector

STEPPER MOTOR
VEXTA PK244M-02AA
2 Phase 0.9 deegre

Title
Unipolar Stepper Motor Controller
Author
a.ratti
Document
1
Sheets
1 of 1
File
Settings\Alberto\Desktop\Stepper_Controller.dsn
Date
16-11-2008
Revision
1.0

PC RS232 connector

TTL to RS232 converter

Max 232

+5V

10nF

1uF
1uF
1uF
1uF

From PicMicro (TTL Tx)

To PicMicro (TTL Rx)

Title		
Author		
File		Document
toiDocumentiTinyCAD ExamplesMax232_b.dsn		
Revision	Date	Sheets
1.0		1 of 1

Heat Sink necessary

LM338K

V Out

V in

Adjust

R1

From +12 to +24 V

$$\text{Motor Current (I)} = \frac{1.2}{R1}$$

Adjust R1 for Motor nominal current

Uniplar Stepper Motor

White

Yellow

Red

Blu

Black

Green

ULN2803

18 17 16 15 14 13 12 11

10

9

1 2 3 4 5 6 7 8

From Pic pin 6 (PortB.0)

From Pic pin 7 (PortB.1)

From Pic pin 8 (PortB.2)

From Pic pin 9 (PortB.3)