UPCX 530 "Extended universal positioning controller"

Purpose of the project

Based on the DC servomotor controller developed by Lawrence Glaister and the H-Bridge expansion by Maximilien Mousset, I developed a firmware for the dsPIC30F4012-30 that can also control sensored BLDC motors using TTL level step/direction and enable signals. BLDC motors are very powerful and are commutated electrically with the help of three hall sensors that give feedback on the position of the rotor inside the magnetic field.
The controller can now either be used for DC servomotors or sensored BLDC servo motors. The type of motor connected to the device is determined by the state of the three hall sensor inputs: If all are found high on startup, the controller switches to DC-motor mode; if at least on line is found low on startup, the controller switches to BLDC motor mode.
I also added the ability to control the motor step input by an internal timer. Although not perfect, this allows easy testing of motor movements without an external step signal source.
To allow adjustments of the PID parameters and BLDC-specific parameters, I wrote new control software based on the tuning tool by Maximilien Mousset using Embarcadero Delphi XE together with a small programming dongle.

V. Besmens December 2012

Included in the project

- This PDF document
- Part list in PDF with Reichelt (a German component store) order numbers.
- Firmware source and HEX file for dsPIC30F4012-30 controller written in dsPICC (licensed version needed to build project). Refer to main.c file for details on optimization-, linker- and fuse-settings.
- Schematic, Layout (Altium designer and PDF files) and Extended-Gerber including GC-Preview (*.gwk) files for a single axis control board (56mm by 56mm in size) and programming dongle. Also included are paneled gerber files (w.o. bottom silkscreen) for nine control boards and four programming dongles on a single PCB with 1mm gap between the boards.
- Embarcadero Delphi XE source files and compiled project to adjust PID parameters. To compile the project yourself, you will need some components installed (one of them is not free, sorry). Please refer to the source code for details.

Technical data

- The maximum supply voltage for the motors with the current components used is 48V regulated. The maximum motor current is limited by the PCB trace width and should not exceed 10A per motor (this is around 7A per motor phase). This limits the motor power to 240W using 24V VMOT and 480W using 48V VMOT, although I did not test that to the limits. To get the maximum rpm use a slightly higher supply voltage than specified for the motor.
- The maximum step frequency I saw was around 110KHz at 1 encoder position per step with 2000 impulse per revolution encoders and Nanotec (www.nanotec.de) DB57S1 BLDC motors running at 24V (this results in 3300rpm). The maximum step frequency depends on the motor type (BLDC or DC), motor specifications, encoder type, cables and motor supply voltage.

What you need to rebuild the firmware sources

A pre-build HEX-File for programming the controller is included within the project files.
If you want to build the firmware from the sources, you will need Microchips C30 compiler (Version 3.00 and above) which is not free because you will need optimization level 3.
Please refer to the notes in file main.c.
Make sure that you use the right fuse settings also described in main.c.

What you need to build the controller yourself

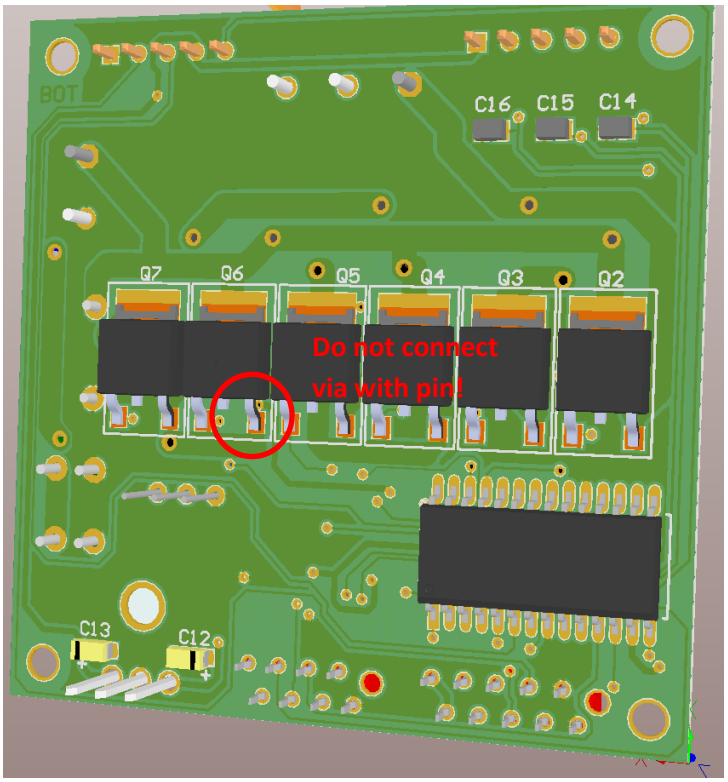
- A PCB-manufacturer that accepts Extended Gerber files or GC-Preview files to create the dual layer PCB's. The PCB's need 6mil / 6mil (0.15mm / 0.15mm) technology for structures and gaps with a minimum hole size of 0.4mm.
- The ability to solder SMD parts with 0805 size.
- The ability to build a heatsink for the bridge FET's if you want to use higher motor currents than around 5A per phase.
- A PIC programmer that is capable of programming the dsPIC 30F4012-30 (e.g. Microchip PICkit 3).
- A sensored BLDC or DC motor with TTL-Level A/B encoder

What you need to run the controller

- You will need to know how to tune a PID loop. Refer to Maximilien Mousset's website (search for YAPSC) or Wikipedia.
- A power supply with enough power to run the motor(s). If you want to run the motor(s) with 48 Volts, the supply should be regulated to keep the capacitor C5 and the driver FETs within their voltage range. For 24 Volt VMOT regulation is not required.
- Another regulated power supply with 10 to 12 Volts and 250mA current per control board.
- For first tests of the circuit use a laboratory supply where you can limit the motor current to find the right hall-sensor, motor-phase and encoder-phase connections. This is not always obvious from the motor datasheet and will usually end in a try and error procedure with 3 Hall + 3 Phase + 2 Encoder cables.
- Although not absolutely necessary a TTL frequency generator for step signal generation to test the circuit and the tuning.
- At least one programming dongle to adjust the PID settings
- A PC running Microsoft Windows XP or higher with one serial port (DB9). USB-Serial adaptors are also fine if the driver is installed correctly and you know the correct port number.
- The adjustment software (included in the project)

Notes on Building the Servo-Controller PCB

- The minimum component size is 0805.
- The board is populated on both sides.
- There are some components that are not necessary when you want to build a DC controller only. Please refer to the part-list for details.
- For some of the capacitor I first decided to use tantalum types but switched to non-pol. MLC types later. Therefore some cases do have a polarity sign. You can ignore it.
- C5 is a radial capacitor that is bended down on the board and then must be fixed with some glue (you will find a populated board 3d image with the provided files).
- J2 and J1 have their pin 1 on the opposite side of the notch!
- The driver FETs can be cooled by pressing the whole board against an aluminum plate using a silicone heat pad. You should mount the board using spring-washers in this case and mount the FETs very even onto the PCB.
- If you decide to use a linear regulator for U1 it will get very hot (around 70°C) even with the small heatsink. To reduce heat you can reduce the 12V supply to 10V (around 50°C) or place the regulator on the bottom side of the PCB and build a heatsink that cools it together with the driver FETs. As an alternative you can use the switching regulator. You can also place a 3 pin power connector at the location of U1 to supply 12V and 5V from an external source.
- Adjust the overcurrent protection by measuring the voltage at TP2. Please refer to schematic for details. Adjust it for to 1.5 to 2 times the rated motor current.
- Make sure to place the driver FETs accurately as there are some vias nearby. There is one critical connection where you can easily connect a via with a pin of Q6:



Notes on Building the Programming dongle

- The minimum component size is 0805.
- The board is populated on the top side only.
- J2 has its pin 1 on the opposite side of the notch!
- Keep the ribbon cable between the Programming dongle and the controller board short for reliable in system programming of the firmware (max 100 to 150mm).

Selecting the motors and Encoders

Refer to the Technical Data section above for maximum voltage and current.
The encoders must generate a TTL-level (5V) A/B signal with an optional Index pulse (for example Avago HEDS 5400 series encoders). The PCB is intended to be used near the motors; it does not contain differential receivers for the encoder signals but the controller has a build in digital low pass filter to reduce noise.
A BLDC servo must contain TTL-level (5V) hall sensors that generate 6 states per electrical circle. The three phases for the motor windings must be connected using "star" wiring. I did not test "delta" wiring.

Notes on Connecting a DC motor

Connecting a DC servo motor is easy. Use Pin1 and Pin2 of J4 to connect the motor leads. After adding some P-Value in the adjustment software, you will either find the motor trying to keep its position or you will find the motor running out of its position when you rotate the motor axis and the software reporting "Max error exceeded". In the second case simply swap the motor leads or the A/B leads of the encoder.

Notes on Connecting a BLDC-Motor

Connecting a BLDC motor is somewhat more complicated. You will have to find the correct relationship of the 3 motor phases to the 3 hall signals after adding some P-Value as well as finding out if the A/B phases of the encoder are in correct order. There will be different combinations that will work. I used the try and error method for hall sensor / motor phase connection for some motor types:

Connector	Nanotec DB57 series*	Motionstep 57mm series**
J5 Pin1 (VCC)	Orange (VCC)	Red (VCC)
J5 Pin2 (GND)	Black (GND)	Black (GND)
J5 Pin3 (HALL1)	Green (H3)	Blue (Hall Phase C)
J5 Pin4 (HALL2)	Gray (H2)	Yellow (Hall Phase B)
J5 Pin5 (HALL3)	Yellow (H1)	Green (Hall Phase A)
J4 Pin1 (P1)	Brown (U or Phase A)	Green (Phase A)
J4 Pin2 (P2)	Blue (W or Phase C)	Yellow (Phase B)
J4 Pin3 (P3)	White (V or Phase B)	Blue (Phase C)

* www.nanotec.de

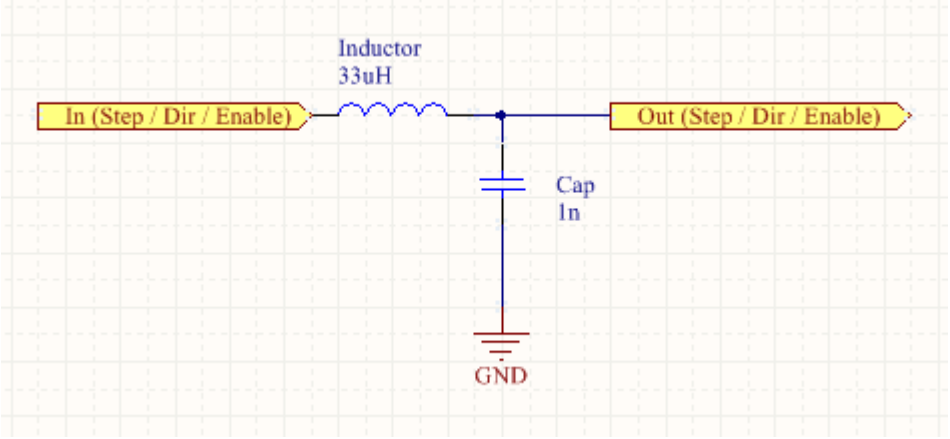
** The company "Motionstep" does no longer exist!

Notes on Control signals and Power sequence

There is no special power sequence necessary. VMOT must be available at least when the enable signal is pulled low and step impulses are fed into the device. Make good ground connection between VMOT and the 12V supply.
The unit is controlled from J1.

Pin (J1)	Signal	Type	Description
1	12V	Power	Regulated 10 to 12V 250mA when using 78S05
2	12V	Power	Regulated 10 to 15V 150mA when using TSR 1-2450
3	Step	In (TTL)	Step input on HL transition when Enable is low
4	Dir	In (TTL)	Direction input
5	Enable	In (TTL)	Device is enabled when low
6	Error	Out (OC)	Open collector error output, low on error, positioning errors can be reset by pulling enable high and low again
7	Index	Out (TTL)	Index signal from encoder if available on encoder
8	NC		No connection
9	GND	Power	GND
10	GND	Power	

The controller generates noise on control (Step/Dir/Enable) signals and is also very volatile to noise on those signals. If you find the unit working as expected using the build-in oscillator but not working stable with external control signals, you can reduce the value of R2, R3, R4 to 1K and / or place an L-C filter into to signal lines as shown below:



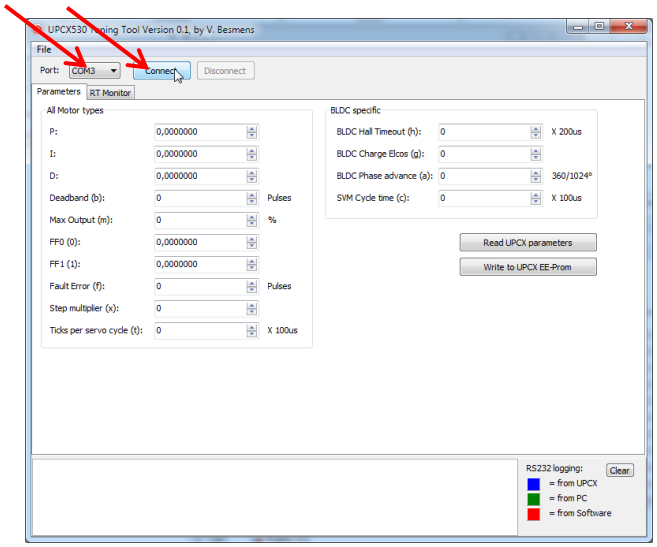
You will probably have to change the values for the inductor and / or the capacitor.

You can also try T-Filter components for noise reduction.

Adjusting the PID and BLDC Parameters / Using the Software

Adjust the PID parameters using the programming dongle connected via RS232 together with the software. For details on adjusting the PID parameters please refer to Maximilien Mousset's website and his description. You cannot use his software due to the BLDC expansion and the build in test-oscillator. But you can use a terminal program (57600Baud 8N1) as well.

The software I wrote is very simple. It has some issues and tends to block the serial connection but it does the job. If it does not react use the task manager to kill the software and restart it.
After executing the software make sure to select the right COM-Port and **make sure you press connect first**.



You can read and write the PID and BLDC parameters from and to the controllers EEProm using the "Read" and "Write" buttons.

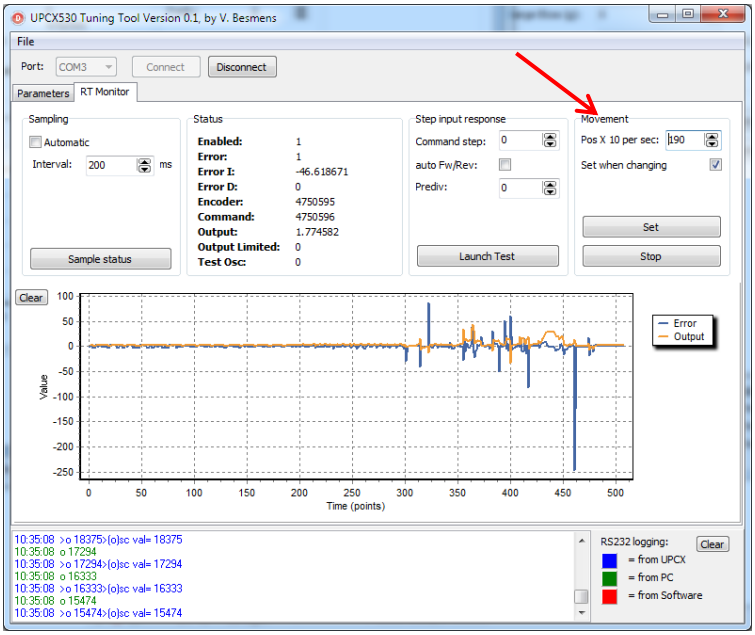
PID Parameters (first tab)
Please refer to Maximilien Mousset's website.

BLDC Paramters (first tab, affects BLDC mode only)

BLDC Hall timeout	The timeout where Hall sensor state is read again if they did not change before. 50 x 200us is the default value. Normally this value does not need tweaking.
BLDC Charge Elcos	If set to 1 the bootstrap capacitors will be charged prior to applying PWM signals to the FET's. For IR2101S you can leave this value at 0 as the IR2101S has an integrated undervoltage protection. 0 is the default value.
BLDC Phase advance	You can adjust the electrical angle the motor phase it advanced from the hall signals. Usually you can leave the value at 0. You can tweak it a little for a higher maximum speed or softer turning of the motor. 0 is the default value.
SVM cycle time	The cycle time of the space vector modulation. This is the time when the Phase values are re-calculated. A value of 4 x 100us is usually OK but you can try playing around with the value. Using a short cycle time will probably affect the overall performance and maximum step frequency of the device as the calculation needs a lot of processing time. 4 is the default value, allowed values are from 1 to 4.

Status Sampling and Step Input Response (second tab)
Please refer to Maximilien Mousset's website.

Movement (second tab)
I build an internal oscillator into the dsPIC that generates step impulses for testing purposes. This oscillator and its interrupt have a very low priority. This affects the stability of the pulses especially if fast PID or SVM update rates are used.
You can set the oscillator values using the marked settings in the picture below. Please revert the sign of the "Pos X per sec" value to alter the direction. The oscillator does not ramp up! The Positions per second are added on top of any external step pulses.



References

Lawrence Glaister's website: http://members.shaw.ca/swstuff/
Maximilien Mousset's website: http://www.max-mod-shop.com
Microchip dsPIC BLDC SVM example (AN957): www.microchip.com
Graphicode GC-Preview to review gerber files: http://www.graphicode.com
My website: http://www.vbesmens.de