XMOS Brushed DC and Stepper motor examples

Draft Date: 2011/11/18

Table of Contents

1 Introduction

Motors can be divided into different types. XMOS provides a solution for electrically commutating Brushless DC Motors and Permanent Magnet Synchronous Motors, in the *sw_motor_control* open source repository.

The sw_basic_motor_examples provides two alternative uses for the *XP_MC_LVM2* power and control boards. The first provides speed control of two brushed DC motors. The second provides position control of a two coil stepper motor.

2 Supported hardware

The XMOS *XP_MC_LVM2* kit contains a control board and a power board. These can be used to control both the DC and stepper motor examples.

Brushed DC motors come in many standard voltages. Stepper motors are more commonly 12V. The motors which were used to build these demonstration applications are both 12V.

The brushed DC motors are Hennkwell HG37D670WE12-052 DC gear motor. It has 12V coils, a built in 2 channel QEI encoder, and a 52:1 gear head.

The stepper motor is a two coil Astrosyn 12V stepper.

NOTE - The power board in the *XP_MC_LVM2* kit is designed for 24V, but will work with a 12V power supply with a single modification. The undervoltage protection circuit needs modification. The resistor R99, initially 140 Kohm, must be replaced with a 68 Kohm one.

3 Brushed DC Motor

IN THIS CHAPTER

The directory *app_DC_motor* contains an application for speed control of two DC motors.

3.1 Physical connections

The code uses a simple unipolar PWM to control the high sides of an H-bridge, the low sides being merely switched depending on the direction of rotation. Half-bridges A and B are used.

Speed detection uses the hall effect inputs, which are attached to magnetic quadrature encoders built into the motors. The quadrature encoders have two channels and no index channel, so only speed is detected. Hall signals A and B are connected to the two channels.

3.2 Software components

The following threads are running on the motor platform.

Core	Thread
0	do_wd (watchdog)
0	controller (simple control)
0	display_gpio (user IO)
1	motors (motor control)
1	pwmSinglePort

The system requires the single port PWM component from the *sc_pwm* open source repository.

3.3 Main control loop

The main control loop is run in the *motors* thread function.

3.3.1 Speed detection

The Hall sensor port is used to monitor the motor speed. The port is monitored using a port event, and as it passes one specific point in the 4 value cycle, the rotation count is increased.

The brushed DC motors that were used during development have a 52:1 gear unit on top of the spindle, and have 12 rotations of the encoder for a single revolution of the motor spindle.

Constants *GEAR_RATIO* and *ENCODER_COUNTS_PER_REV* set these values within the code.

3.3.2 External control

There are two control paths for changing the speed of the motors. The *display_gpio* thread function, which provides input and output via the LCD screen and buttons, has one channel in and out of the control loop.

The *controller* thread function has another channel input to the control loop for adjusting the motor speeds. This thread represents a more sophisticated control algorithm, although the example merely sets the initial speed.

3.3.3 PI control

The core of the *motors* function periodically measures the speed of the motor, and uses a PI controller to alter the PWM duty cycle to keep the motor speed constant.

The frequency of measurement and update of the PI controller is defined by the constant *PERIOD*, which is initially set to 10Hz, to avoid quantization problems in speed measurement.

3.3.4 PWM configuration

The application uses the single port PWM controller from the *sc_pwm* module. This is capable of controlling multiple single bit PWM control lines. In the case of the DC motor, four PWM lines are required, two for each motor.

The constant *RESOLUTION* defines the number of PWM periods in each cycle, and the constant *TIMESTEP* defines the duration of each period, in 20ns multiples. Initially, *TIMESTEP* is set to 2, meaning that the duration of each PWM period is 40ns, and the *RESOLUTION* is set to 256, giving 256 PWM steps, for a total PWM cycle time of 5120ns.

3.3.5 Extending the demonstration for more motors

An XMOS core is capable of controlling many more than two motors. The *sc_pwm* repository contains multiple versions of the PWM controller, allowing a variety of configurations to be used.

The *pwm_multibit_port* module allows the PWM to control multiple channels using a 4, 8 or 16 bit port. Since each brushed DC motor uses 2 pins, an 8 bit port would allow the control of 4 motors.

4 Stepper motor

IN THIS CHAPTER

A stepper motor has a pair of coils. The controller can move the spindle a very precise angle by putting current in each coil alternately. To commutate the rotor, the following pattern can be used.

- Coil A, current +
- Coil B, current +
- Coil A, current -
- Coil B, current -

The XMOS implementation supports micro-stepping, where the current in each coil is controlled propertionally, to introduce angular steps of a fraction of the coil angle.

4.1 Physical connections

A stepper motor has two independent coils. Consequently, it requires 4 half-bridges. The software is configured to have the first coil attached to motor 1 port A and motor 2 port A, and the second coil on motor 1 port B and motor 2 port B.

The stepper motor used during development of the algorithm is from Astrosyn, and is a 12V unit.

4.2 Software components

The following threads are running on the motor platform.

Core	Thread
0	do_wd (watchdog)
0	controller (simple control)
1	motors (motor control)
1	adc_7265_triggered (adc)
1	pwmSinglePortTrigger

The system requires the ADC component from the general motor control repository *sw_motor_control*.

4.3 Main control loop

The main control loop has a period of 20 Hz. There are two states which it can be in, speed control or step counting. The *doSpeed* and *doSteps* variables control which mode is current active.

Running outside of the *singleStep* function is the main *motor* control loop. If it is running in speed control mode, an outer timer is used to call the singleStep at regular intervals to rotate the rotor at the correct speed. This loop also contains channel communication endpoints for receiving control commands for an external control thread.

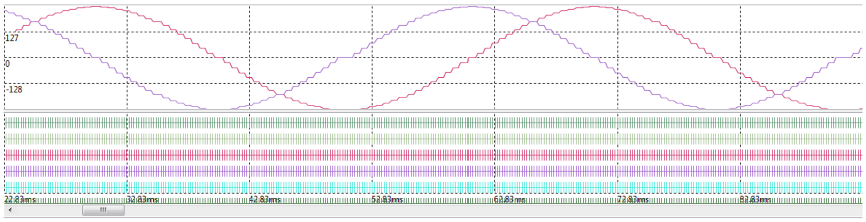
4.3.1 Single step control function

The function *singleStep* moves the spindle by a single position. It has arguments that are channel links to the PWM and ADC modules, a microstepping period, and a step position.

Although the *singleStep* function moves the rotor by one whole step, it performs a series of microsteps between the two end points. The number of microsteps are determined by a compile time constant *STEP_SIZE*. This is the number of positions through the *COS_SIZE* cosine table which each microstep will take. By default, the cosine table has 256 entries, so a *STEP_SIZE* of 256 will give no microstepping, 128 will give one intermediate microstep, and 16 will give 16 microsteps.

During the *singleStep* function, a timer provides control for the stepping operation, which generates reference coil currents for the specific microstep. A second timer, running faster, uses the ADC to measure the actual coil currents and uses a PI controller to adjust the PWMs to produce the correct current. This fast inner timer is a torque controller.

The diagram shows an XSCOPE trace of the reference currents for the two coils. The number of microsteps is set to 16, which can be seen by examining each signal and noticing that in each quarter phase there are 16 steps.



4.4 ADC configuration

In the default version of the code, the ADC query function is set up to return a pair of hard coded values for the measured currents, and the PI controllers are disabled.

The *XP-MC-LVM2* board has a dual ADC with a multiplexor that allows the selection of the sampled channels. Since there are two coils, a dual ADC will be able to sample the two currents without the aid of the multiplexor. However, since the board is also designed to perform sampling for a complex FOC, the ADC channels measure the current in each half-bridge, rather than in either of the pair of half-bridges which control each coil.

Instead of making the stepper motor code more complex by performing a pair of samples, the code is left simple, but without torque control.