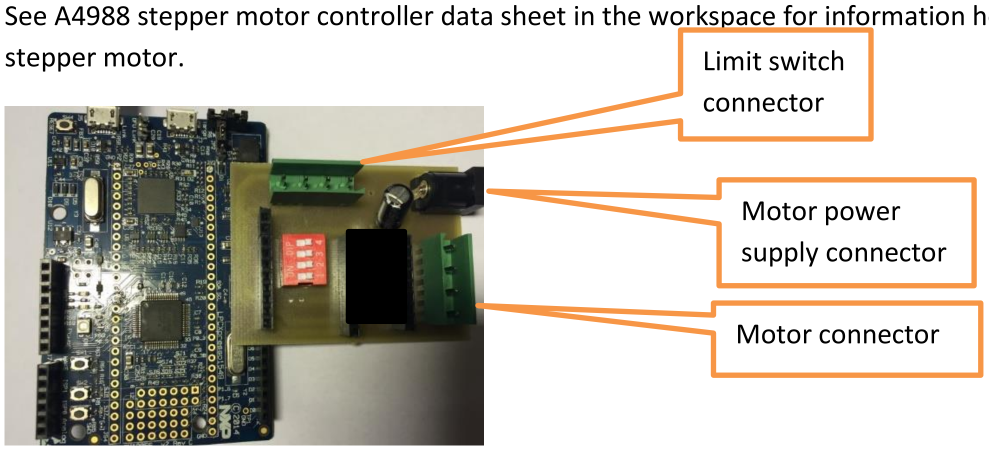
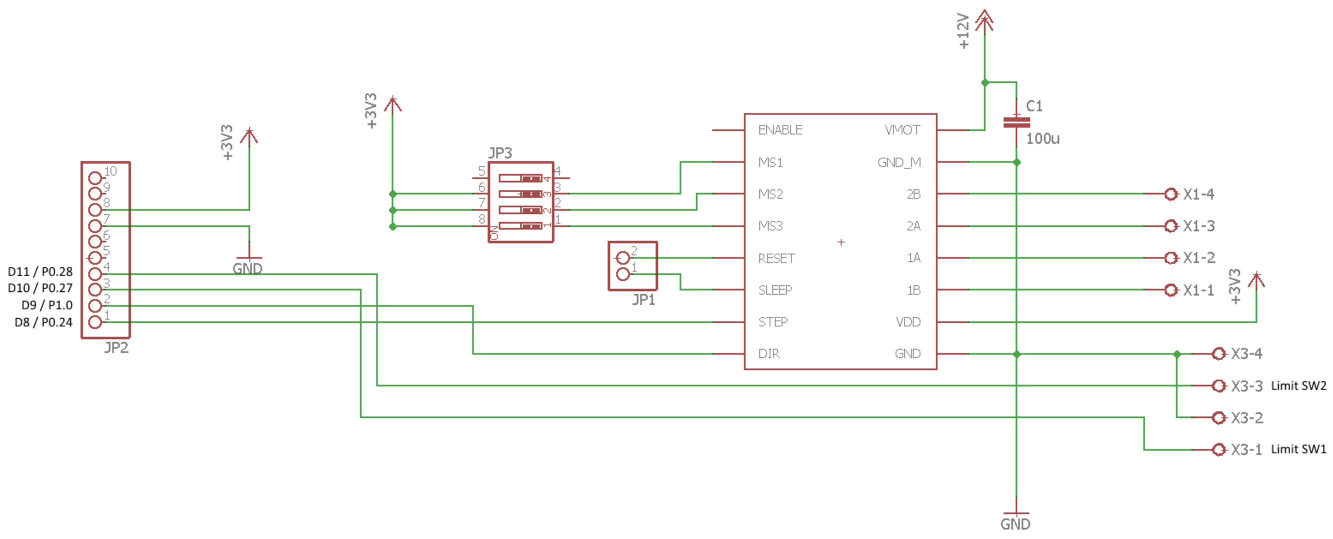
Controlling a stepper motor

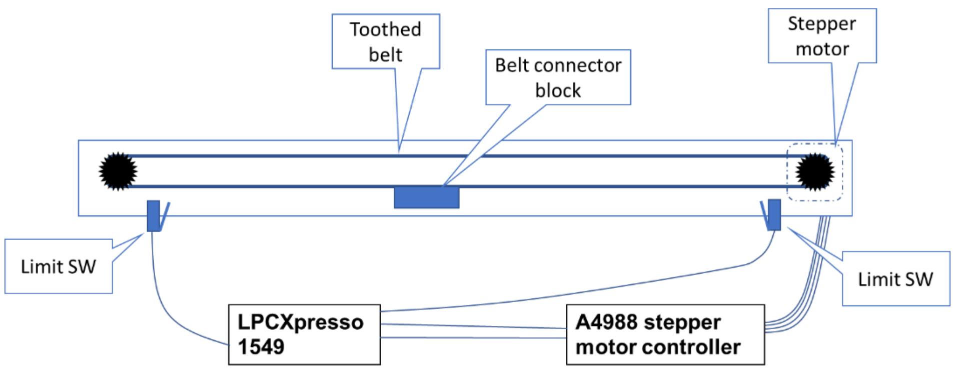
This lab requires an add-on board that is plugged into your LPCXpresso. The picture below shows proper orientation of the board. Note that the actual board contains a stepper driver module which is missing from the picture below.

See A4988 stepper motor controller data sheet in the workspace for information how to control the stepper motor.



The schematic below shows the mapping of limit switches and stepper motor to the pins of LPCXpresso. Both limit switches are grounding switches so you need to enable pull-up resistors to read limit switches.





When the motor takes steps the connector block moves with the belt. When block hits a limit switch it closes and the IO-pin state changes. When the block moves away the switch opens. When a limit closed the program should not step motor to that direction.

There are four DIP switches on the stepper board. For these exercises set all switches to OFF. In simulator set step mode to Full Step.

If you do this exercise with signal capture board and simulator then the pins to use are:

- Step P0.24
- Dir P1.0
- Limit SW1 P0.9
- Limit SW2 P0.29
- Button1 P0.8
- Button2 P1.6
- Button3 P1.8

Exercise 1

Write a program that has two tasks:

Task 1

Reads limit switches and sets red led on when limit switch 1 is closed and green led on when limit switch 2 is closed. When limit switches are open the corresponding leds are off.

Task 2

Reads LPCXpresso buttons 1 and 3. While button 1 is pressed the stepper motor runs SLOWLY clockwise and while button 3 is pressed motor runs SLOWLY counterclockwise. When the button is released the motor stops. If both buttons are pressed at the same time the motor must not run. If either of the limit switches is closed the motor must not run.

If you hit the limit switch in your tests unplug the motor power supply and turn the motor manually until limit switch is released. Then apply the power to the motor again.

Exercise 2

Write program that has at least two tasks and one binary semaphore:

Task 1

Task does the following:

1. When task starts it reads limit switches and waits until both limit switches are open. The blue led blinks during waiting.
2. Then task signals “go” using binary semaphore
3. Then task enters a loop where it reads limit switches and switches red led on if limit switch 1 is closed and green led on if limit switch 2 is closed.

Task 2

Does the following:

1. Task waits on the binary semaphore. When it gets the semaphore ("go" signal) the task continues
2. Task starts to run stepper motor in either direction and runs the motor at constant speed until one of the limit switches is closed. Then the motor direction is toggled. Note that you need to run a few steps in the opposite direction before the switch opens again or you need to know which limit switch is at each end and monitor only one switch per direction.
3. If at any time after "go" both limit switches are closed the motor is stopped immediately and task waits for 5 seconds. Then the switches are checked again and running continues only if both switches are open.

Note that limit switch is a simple IO-pin which allows multiple **readers** without locking. You can share IO-pin objects between tasks but may not create multiple IO-pin objects on the same physical pin.

Exercise 3

Write a program that does the following:

Task 1

Task does the following:

1. When task starts it reads limit switches and waits until both limit switches are open. The blue led blinks during waiting.
2. Then task signals "go" using binary semaphore
3. Then task enters a loop where it reads limit switches and switches red led on for one second if limit switch 1 is closed and green led on for one second if limit switch 2 is closed.

Task 2

Does the following:

1. Task waits on the binary semaphore. When it gets the semaphore ("go" signal) the task continues
2. Task starts to run stepper motor in either direction and runs the motor at constant speed until one of the limit switches is closed. Then the motor direction is toggled. Program must count the number of steps between limit switches. You should do multiple runs and use averaging to get an accurate value.
3. When the number of steps between the limit switches is known the task switches the blue led on for two seconds. Then the blue led is switched off and task starts to run motor back and forth so that it touches limit switches lightly but does not close them. Note that there is some hysteresis in the limit switches. When switch is closed you need to back out more than just one step to open the switch again.

If you do this exercise in the simulator the program must work with all Image widths.