EEE 158: Electrical and Electronics Engineering Laboratory V

Machine Problem 2: Stepper Motor Control

Introduction

Electric motors come in a variety of designs and sizes to accommodate equally-diverse applications, ranging from the positioning motors found in printers and scanners, all the way to the heavy traction duties of railroad and mining. One such major design is the *stepper motor*, whose principal features are the ability to incrementally move by applying an appropriate number of pulses, and reverse direction by simply reversing the sequence of applied pulses.[1] Such motors are used in applications where precise movement and quick reversal are required, such as printer / scanner movements and robotic arms. Many driver ICs and modules for such devices are available, such as Toshiba's TB6600HG or Allegro Microsystems' A49xx Series.

Modern industrial control systems feature a concept known as *local/remote control*. As the scope and complexity of such systems grow, so do the need to be able to monitor and control not only a small part of a process, but also the whole process; this makes always-local monitoring and control impractical. However, even with the benefits of remote monitoring and control it is desired (and even required in some jurisdictions) to have some level of local control; for example, to enable testing during periodic maintenance.

Safety Considerations

Personal safety shall always be of utmost priority in the design and operation of control systems. Furthermore, in practical systems software must never be the only means of operation and control; hardwired logic must always be available to enable safe shutdown in the event of a failure. For this MP though, it is "waived" due to the low energy levels involved.[2] In particular, safety-critical features like a normally-closed emergency-stop tripwire or pushbutton that fully bypass any software control are absent.

Problem Statement and Objectives

One common equipment encountered in an industrial setting is the conveyor belt, used to move items around. Although such belts are usually unidirectional, there are cases where reverse operation is also required. The latter is used, for example, in conveyors attached to storage bins – one direction for filling the bin with some raw material from delivery, the other for emptying the same bin when the material is scheduled for consumption in some subsequent process.

Using a stepper motor via a driver module, you are to simulate the operation and control of such a conveyor using your PIC32CM Curiosity Nano Board.

1 More information on such motors are found in EEE 145 (*Electromechanical Energy Converters*).

2 This is **NOT** the case for the power levels encountered in courses (similar to those) housed at the Electric Machines Laboratory (eg. EEE 148 / EE 164). In fact, such courses are a prime example of the importance of safe practices due to the higher energy levels involved (eg. kW-rated machines, DC, and 3-phase systems).

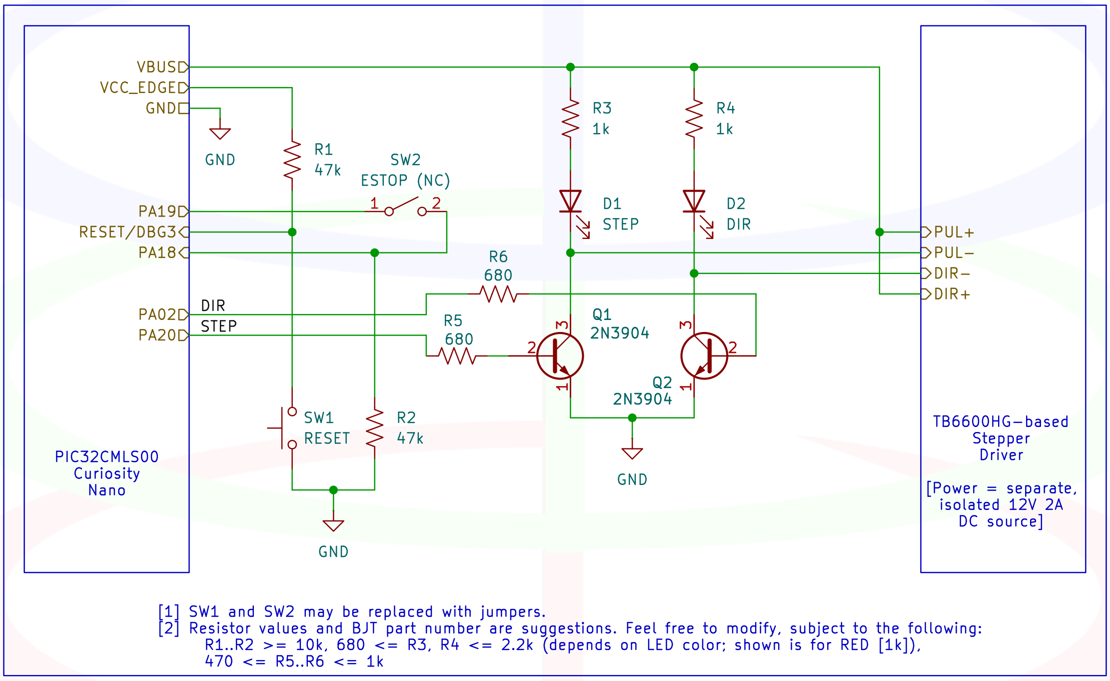
Limitations

Due to time constraints, only remote control with local “force-off” will be implemented. Furthermore, only motor-shaft turning will be observed – no actual conveyor will be driven.

Technical Specifications

The stepper-motor driver module accepts STEP/DIR style inputs. In this method, one step rotation is effected by a rising-edge pulse on the STEP input, while the direction of rotation is determined by the DIR input at the instant a valid STEP pulse is detected. As such, “continuous” rotation may be effected by sending a continuous stream of pulses on the STEP input.[3] For this machine problem, 800 steps correspond to one full revolution, and the permissible average speed range for the stepper motor is between 3 RPM and 225 RPM.

You are expected to build a carrier circuit for your PIC32 Nano Board on a breadboard. Shown below is the reference schematic for the carrier; values may be modified to suit.



Baseline Requirements

NOTE: All baseline requirements are mandatory. Failure to comply with least one requirement shall automatically mean non-compliance with the machine problem as a whole (and hence grounds for an INC, if applicable).

- At system power-up and/or reset, the motor SHALL NOT be spinning.
- The motor’s acceleration or deceleration SHALL be gradual, except during an “emergency stop” condition (*described in the next bullet*) where the speed is not expected to be controlled. More specifically, the number of steps delivered per second SHALL NOT vary more abruptly than (10 steps/s) per 10 ms.

---

3 The very design of a stepper motor permits only discrete changes to steps. As such, a truly-smooth acceleration/deceleration is impossible.

- Two GPIO pins SHALL be externally connected via the carrier circuit, the connection of which serves as an "emergency stop" link. If the link is broken, the system SHALL enter an *Emergency Stop* state where all commands to make the motor run SHALL be ignored.
  - Exiting this mode is effected by re-establishing the link, continuously maintaining its continuity for a minimum of 4 seconds, and *then* pressing (not holding) the on-board button.
  - If the link is again broken before the 4 seconds are up, timing SHALL restart from the beginning when the link is re-established.
- The motor speed SHALL NOT exceed the maximum speed in either direction. To avoid possible overheating of the stepper driver module, if the motor is commanded to run at a speed below the minimum value the system SHALL treat it as if the request was for 0 RPM (aka. standstill).
- The STEP pulse width SHALL be at least 100 µs in each state.
- USART communication settings SHALL be 57.6 kbaud, even parity, no flow control, and 1 stop bit.

Milestones

This machine problem is divided into a series of milestones. A milestone may contain one or more of the following annotations:

- *Requires*: A milestone with this annotation will only be given credit if all listed prerequisites have 100% credit. Applies recursively.
- *Coreq*: A milestone with this annotation will only be given credit if all listed co-requisites are eligible to be given credit. Applies recursively.
- *Implies*: If a milestone with this annotation obtains 100% credit, then all listed milestones automatically get 100% credit. Applies recursively.
- F2F: A milestone with this annotation will only be evaluated in a face-to-face meeting with the handler.

If more than one annotation is present, all annotations SHALL be logically-AND'ed together. For example, consider a milestone with fully-complied prerequisites, but only partially-complied co-requisites. In this case, the milestone will not be considered to have 100% credit for prerequisite and implication purposes.

List of Milestones

- **MS 1 (+10 pts):** The system immediately enters *Emergency Stop* mode if the link between PA18 and PA19 is broken. An exit from this mode is only permitted according to the rules laid out in *Baseline Requirements*.
  - PA19 is configured as an output. The student is allowed to decide whether it is a fixed-HI output, or an intermittent/continuous square-wave output of at least 100 Hz.
  - PA18 is configured as an input, and connected to PA19 according to the provided carrier circuit. The link is considered broken if the digital I/O states of PA18 and PA19 differ from each other for more than 20 ms.
- **MS 2.1 (+7 pts):** After power-up/reset, any keyboard press has no effect until `<Ctrl>+E` is pressed, denoting presence of the terminal. If no valid keyboard input is received within at least 45 seconds, or `<End>` is pressed, the terminal is assumed absent, the motor is turned off, and the system returns to initial conditions.

- **MS 2.2 (+7 pts):** Pressing `<Ctrl>+E` at any time causes the terminal to do a full refresh. If no valid keyboard input is received within at least 45 seconds, or `<End>` is pressed, the terminal is assumed absent and no terminal output is displayed until `<Ctrl>+E` is pressed again. The delay used SHALL be the same as that used for MS 2.1.
- **MS 2.3 (+1 pt):** `<Tab>` is treated as a valid input, but otherwise does nothing. *Coreq*: MS 2.1 or MS 2.2.
- **MS 3 (+10 pts):** The on-board LED blinks with the ON/OFF times indicated in the following table. *Coreq*: MS 1, (MS 2.1 or MS 2.2).

ON time	OFF time	State	Remarks
0 s	> 0 s	Emergency stop	Overrides all others
0.3 s	1.7 s	Terminal absent	
> 0 s	0 s	Terminal present	

- **MS 4 (+10 pts):** Pressing `<Left>` or `<Right>` causes a fixed-frequency 50% duty-cycle square wave to appear on the STEP output, while pressing `<Space>` turns off the square-wave output. Pressing `<Left>` also sets DIR to a fixed HI or LO; pressing `<Right>`, on the other hand, sets DIR to the configuration opposite that of `<Left>`. *Requires*: MS 1, MS 2.1, MS 2.2, MS 3.
- **MS 5 (+15 pts):** The STEP output gradually increases/decreases in <u>frequency</u> rather than being instantaneous. Use of constant-ON-time rather than standard pulse frequency modulation is allowed. *Coreq*: MS 4.
- **MS 6 (+15 pts):** Keys `<1>` through `<5>` are used to select a target speed via linear interpolation, with `<1>` being the minimum and `<5>` being the maximum. Upon system reset or exiting *Emergency Stop* mode, the internal setting corresponds to `<1>`. If pressed while the motor is off, the motor gradually accelerates to the target speed upon receipt of a turn-on command as described in MS 4. If pressed while the motor is already on, the latter accelerates/decelerates to the new speed <u>without</u> changing direction. *Requires*: MS 4, MS 5.
- **MS 7 (+5 pts):** Pressing `<Left>` or `<Right>` while the motor is running causes the motor to gradually decelerate to a stop, <u>then</u> accelerate to the <u>same</u> speed in the <u>new</u> direction. *Requires*: MS 4, MS 5.
- **MS 8 (+5 pts):** Blinking of the on-board LED is <u>amended</u> by the following table. *Requires*: MS 3, MS 6, MS 7.

ON time	OFF time	State	Remarks
0 s	> 0 s	Emergency stop	Overrides all others
0.3 s	1.7 s	Terminal absent	Overrides motor indications
0.4 s	0.4 s	Motor not at target speed	Includes target == off
> 0 s	0 s	Motor at target speed	Includes target == off

- **MS 9:** Provided a terminal is present, the following information are displayed. Under no circumstances does the terminal appear garbled. *Minimum Requires*: MS 2.2, 2.3.
  - (+5 pts) **Emergency-stop status** (additional *Requires*: MS 1)

  - **(+5 pts) Commanded motor off/forward/reverse indication** (additional *Requires*: MS 4) By "commanded", this pertains to the actual state of the DIR line.
  - **(+5 pts) Commanded motor speed** (additional *Requires*: MS 6). By "commanded", this pertains to the actual pulse frequency on the STEP line. Quantitative or qualitative display is allowed.
- **MS 10 (+10 pts):** The whole system is capable of controlling the stepper-motor driver module according to specification. This is the penultimate goal of the entire machine problem. *Requires*: MS 3, MS 6, MS 7. *Coreq*: MS 8, MS 9.

Assessment Policies

1. **This machine problem will be assessed in stages.** There shall be no option to have this MP graded with all milestones in one sitting; you must be able to demonstrate your work as you progress. It is left at the discretion of your handler how each milestone will be assessed – F2F, video recordings, or some other method.
2. **Some milestones are required for completion.** To avoid getting an INC, you must complete up to at least MS 4.
3. **Documentation is required.** The document should give a short summary of your experiences with MP2, as well as the course (if you wish to do so). In addition, the following items must be included alongside the documentation file (not as appendices within the document):
   1. Complete MPLAB X project folder; all source files MUST be inside the project folder.
   2. Pictures and/or videos of all progress towards completion of MP 2
4. **Honor, Excellence, Service.** Students caught committing academic dishonesty (including, among others, copying code from other classmates and/or the Internet, without proper acknowledgement; plaigiarism) may be grounds for failing the course and/or filing of disciplinary proceedings against the student.