SJSU EE138
Introduction to Embedded Control System Design
Lab 5: Motor Control Lab

Readings:

Lecture note: Chapter 5 Motors and Encoders.
Chapter 6 Digital Control.

The lab take a lot of time to complete. You will be utilizing all you have learned this semester into this lab. Do not put it off till the last week. No time extension will be given.

Lab Description:

In this project, students will develop a motor speed/position control program. This program involves EIC, TC interrupts, PORT, and PWM. ADC and DAC are optional.

Your control program should be structured as shown in Figure 5-1. The main() consists of initialization of all related peripherals and a dummy infinite loop. There should be two timer generated interrupts. One for the 7-segment display, keypad interface, and the state machine. This timer interrupt should have lowest priority and should be set to about 60Hz. The other timer interrupt is for implementing the speed and/or position control and a low-pass digital filter. This timer interrupt should have a medium priority level and should be set to 200Hz. Encoder quadrature signal should be decoded by EIC as in Lab 4. This interrupt should have highest priority.

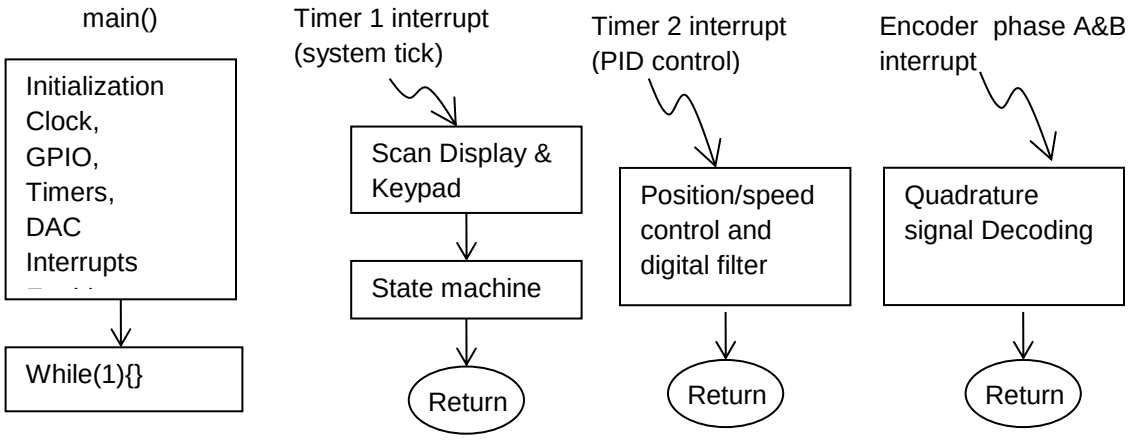


Figure 5-1 Program structure

A fully functional program for this lab is quite extensive. Because of that, the required tasks are organized into three levels and an extra credit task. Grading rubrics will be discussed by TAs in the lab. Higher level tasks are built on lower level tasks. You don't have to demonstrate lower level tasks if you complete higher level tasks.

Level-one tasks

Level-one tasks involve only motor speed control. The specific tasks are as follows

(1) Keypad will select two modes of operation: Idle mode or Speed control mode.
When the key '1' is pressed (for speed control mode), the motor should be accelerated to 1500rpm in about 5 seconds and, after that, the speed should be maintained at 1500rpm.
When the '0' key is pressed (for idle mode), the motor should be decelerated to 0rpm in about 5 seconds (if starting from 1500rpm) and, after that, the motor voltage is set to 0v.

(2) The motor speed (in RPM) should displayed on the 7-segment display at all time. The 1Hz bandwidth digital filter implemented in Lab 4 should be included so the displayed value is steady.

(3) The speed must be controlled by a **PI speed control loop**. Note that the raw calculated speed should be used in the PI control loop, NOT the filtered one.

(4) The program must be based on Interrupt and state-machine using the program structure shown in Figure 5-1.

(5) The state machine should have 4 states: Idle, Accel, Decel, and Spd_ctrl.

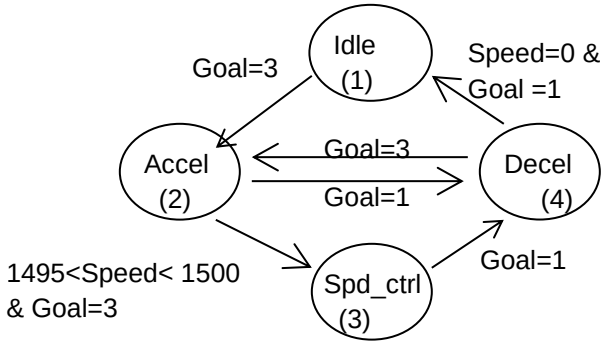
Idle (1): The motor voltage should be set to zero. PI speed control is disabled.

Accel (2): This is a controlled acceleration period. The motor speed is ramped up to 1500rpm over a period of about 5 seconds. The state machine exits this state only when the MEASURED speed (not the speed command) is within in 5 rpm of 1500rpm.

Spd_ctrl (3): In this state, the speed is maintained at 1500rpm.

Decel (4): This is a controlled deceleration period. The motor speed is ramped down to 0 rpm in about 5 seconds (if starting at 1500rpm). The state machine exits this state only when the MEASURED speed reaches 0 rpm.

The following state transition logic should be used:



Note that a variable 'Goal' is used in the state machine. The 'Goal variable is NOT the state variable but it sets the destination (or goal) of the state transition sequence. Goal variable is set by the keypad input. For example, when '1' key is pressed (speed control mode), the Goal variable should be set to #3 which triggers the change of the state variable from #1 to #2 and later to #3.

At any instant of time, the control program operates in one of the 4 states. The foreground task is executed at all times. The foreground tasks include: Keypad scanning and 7-segment display, keypad input interpretation (setting Goal state, in this case) and the state machine. These foreground tasks are executed in the first timer interrupt handler routine. Note that, there is no explicit communication between any of the blocks in Figure 5-1. All communication is done through the state variable implicitly. For example, The 'speed command' variable (used in the PI control loop) is incremented when the state variable is #2. If the state variable is #4, the speed command is decremented. The speed command should be kept at 1500rpm in state #3.

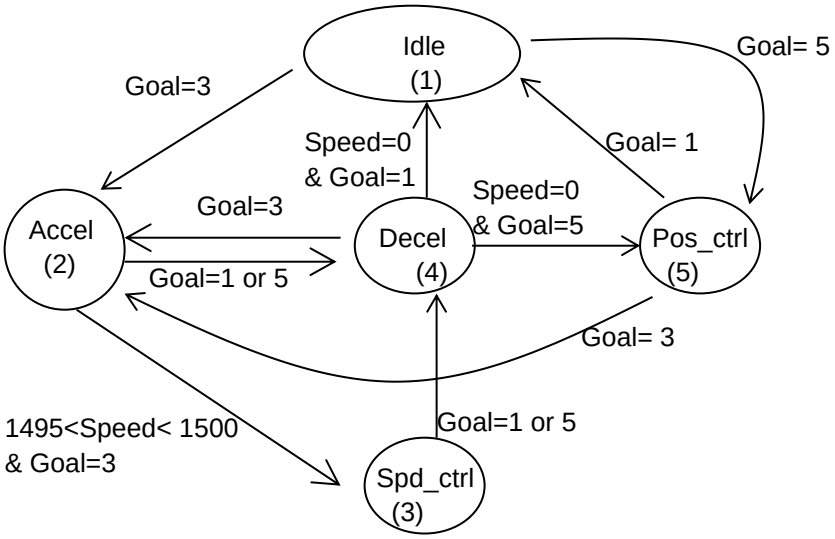
Level-two tasks (subject to change)

Level-two tasks include all the level-one tasks and the following additional functions:

(1) When key '2' is pressed, the control program should be in the position control mode. If the '2' key is pressed while the program is in the speed control mode, the motor must be decelerated to 0 before it enters the position control mode. While in the position control mode, the 7-segment display should show the measured position.

(2) In position control mode, the motor shaft's position should be maintained at 0 degree position by a closed loop PID controller. If, for example, the motor shaft is hand-turned away from the 0 degree, the position control program will bring the position back to zero. The physical zero angle position of the motor shaft can be arbitrarily set.

(5) There should be 5 states: Idle, Accel, Decel, Spd_ctrl, and Pos_ctrl. The following state transition logic should be used.



Level-three tasks (User Interface, subject to change)

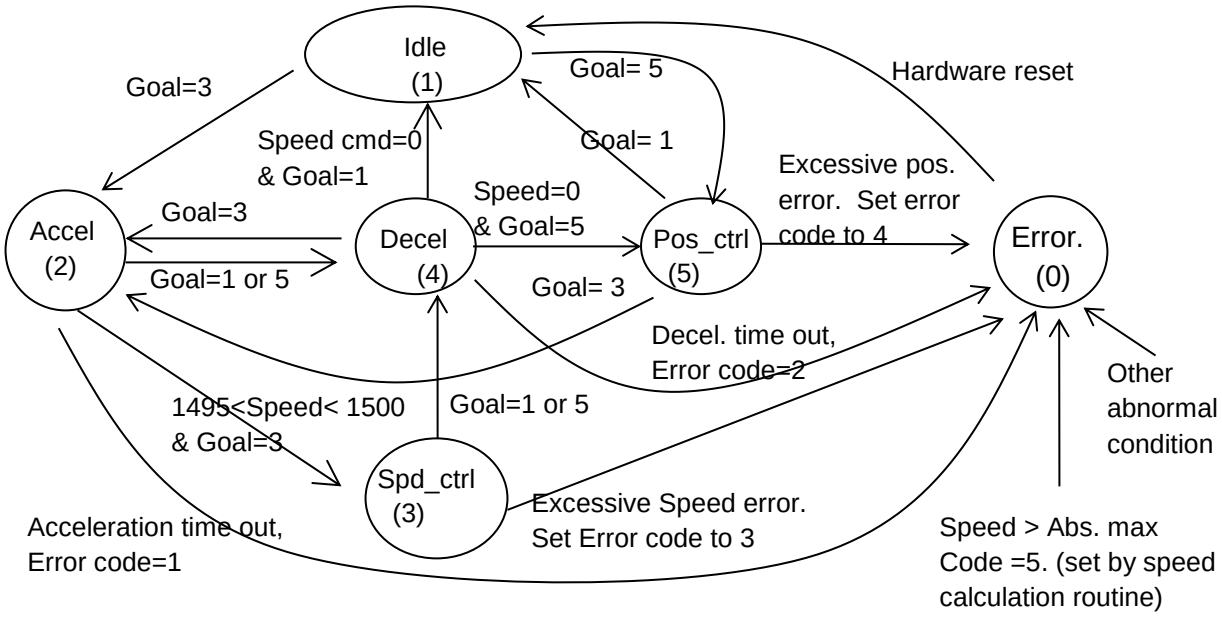
The closed loop control tasks (PI control for speed and PID for position) in Level-three are the same as that in level-two. The main difference between Level-three and Level-two is that level-three tasks include user interface that allows the user to key-in the value of speed and position command.

(1) In the speed control mode, the commanded speed can be entered by keypad and the same for the position control mode. Both speed command and position command can be negative.
(2) There should be multiple display modes displaying: commanded value display, measured value display mode, and control mode display.
(3) A 'user's manual' should be included in your report. This manual shows the user how to change control mode and commanded values via keypad input.

Extra Credit task (Level Four)

Level-Four tasks include all tasks in level-three and the following additional functions. Not all the functions listed below are required to earn 20% extra credit.

(1) The set speed and set position can be entered via keyboard or by turning the POT. The middle position of the POT should be considered as the 0 value.
(2) Implementing the error detection and error state logic as explained next,
(3) The set speed or position can be entered via an analog input channel so that the position or the speed command can be set by an external analog voltage (for a function generator, for example).
(4) The measured position or measured speed are output to PA02 (with the proper scaling, of course). This allows the observation of the actual position or speed with a oscilloscope.
(5) Measure and report the step response of the speed control and the position control.



In the Error state, the motor voltage should be set to zero and the error code should be displayed. The program enters the Error state in any of the following conditions. Once the program entered Error state, a hardware reset is required to bring it back to idle state.

Error code 1: Acceleration time out. If the measured speed does not reach within 5 rpm of the set speed in a certain time period (say, about 10 seconds). To detect this error, a software timer should be implemented in state #2.

Error code 2: Deceleration time out. If the measured speed does not reach 0rpm in a certain time period (say, about 10 seconds). To detect this error, a software timer should be implemented in state #4.

Error code 3: Excessive speed error. If the difference between the commanded speed and the measured speed (i.e., the speed error) exceeds a limit over a period of time (say 2 seconds).

Error code 4: Excessive position error. If the difference between the commanded position and the measured position (i.e., the position error) exceeds a limit over a period of time (say 2 seconds).

Error code 5: Over speed error. Whenever the speed of the motor exceeds the absolute maximum regardless the state the program is in. This error condition can be set at the timer interrupt handler routine where speed is calculated. This is an example of how a low-level routine can change the state transition of the control program.

Additional error codes can be included. Whenever an abnormal condition is detected, the state variable should be changed to 0 and this forces the program to go into the error state.

Peripheral Address declaration:

Address	-	SAMD20 Syntax Code
0x40001800	-	EIC // address for external interrupt
offset 0x00	-	CTRL.reg // enable.
offset 0x01	-	STATUS.reg // synchronize
offset 0x08	-	INTENCLR.reg // disables specific interrupts
offset 0x0C	-	INTENSET.reg // enables specific interrupts
offset 0x10	-	INTFLAG.reg // read what Interrupt flag triggered and to clear the flag
offset 0x18	-	CONFIG[0].reg // set filter and detection of interrupt for EXTINT[0-7]
offset 0x1C	-	CONFIG[1].reg // set filter and detection of interrupt for EXTINT[8-15]
0x40000400	-	PM //definition address for PM functionality
offset 0x1C	-	APBCMASK.reg //used to enable peripheral clocks on the APBC bus
0x42002800	-	TC2 //definition address for TC2 functionality
0x42003000	-	TC4 //definition address for TC4 functionality
0x42003800	-	TC6 //definition address for TC6 functionality

offset 0x00 - CTRLA.reg // used to setup and enable TC
offset 0x0C - INTENCLR.reg // disables overflow interrupt
offset 0x0D - INTENSET.reg // enables overflow interrupt
offset 0x0E - INTFLAG.reg // used to check over flow and to clear flag
offset 0x0F - STATUS.reg // used to check synchronization of registers
offset 0x10 - COUNT.reg // increments on every clock cycle
offset 0x14 - PER.reg //sets the top value for the 8bit mode counter
offset 0x18 - CC0.reg //sets the compare and capture for even pin then outputs to WO[0]
offset 0x19 - CC1.reg //sets the compare and capture for odd pin then outputs to WO[1]