External Interrupt Control and Timer Counter Interrupt

Lab 4

Readings:

Atmel SAMD20 DataSheet

- Section 19: EIC - External Interrupt Controller

In class Lectures

- Lecture 11 Interrupt (***hint:** for the interrupts to work properly NVIC must be set correctly and the correct interrupt handler must be used. look at this lecture it is very important*)
- Lecture 12 Motor_driver

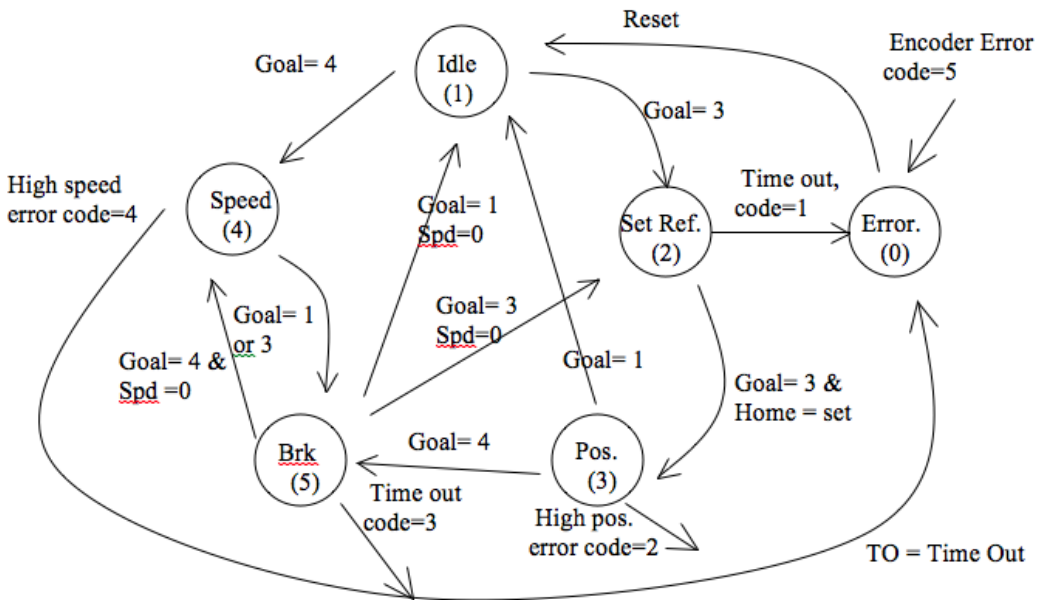
Lab Description:

In this project, students will develop a motor control program. The program should be able to control the speed and position of the provided motor using a keypad and/or a POT. To complete this successfully students will have to implement the following, EIC and TC interrupts, PORT, ADC, DAC, and PWM. It is important to organize the code into states so that it is easier to debug and follow the program.

State diagram:

The following example state machine for the control program is only for reference. You can design your own state machine. There are 5 states in this state machine:

0 – Error State
1 – Idle State
2 – Position Initialization State
3 – Position Control State
4 – Break State
5 – Speed Control State



State Transition Diagram

Idle State --- This is the power-on reset state. Motor is not energized. From this state, user can enter position control or speed control.

Position Initialization State --- This state is always entered before entering the position control state. In this state, the motor is commanded to move in one direction slowly until the home position key on the key pad is pressed. This signals the home position (0 angle position). After a home-key is detected, the machine enters the position control mode. If the machine stays in this state for more than few seconds, the machine enters the Error state with an Error code of 01.

Position Control State --- The motor is under position control. If the position error is greater than a certain value for more than few seconds continuously, the machine enters the error state with an error code of 02.

Break State --- This state is always entered before leaving the speed control mode. If the speed does not reach 0 after few seconds, the machine enters the error state with an error code of 03.

Speed Control State --- The motor is under speed control. If the speed error is greater than a certain value for more than a few seconds continuously, the machine enters the error state with an error code of 04.

Error State --- Motor is de-energized. Error code is flashing on the display until the rest button is pressed. Other than the error code 01~04 described above, the encoder ISR can force the state going into the error state with an error code of 05.

State transition table

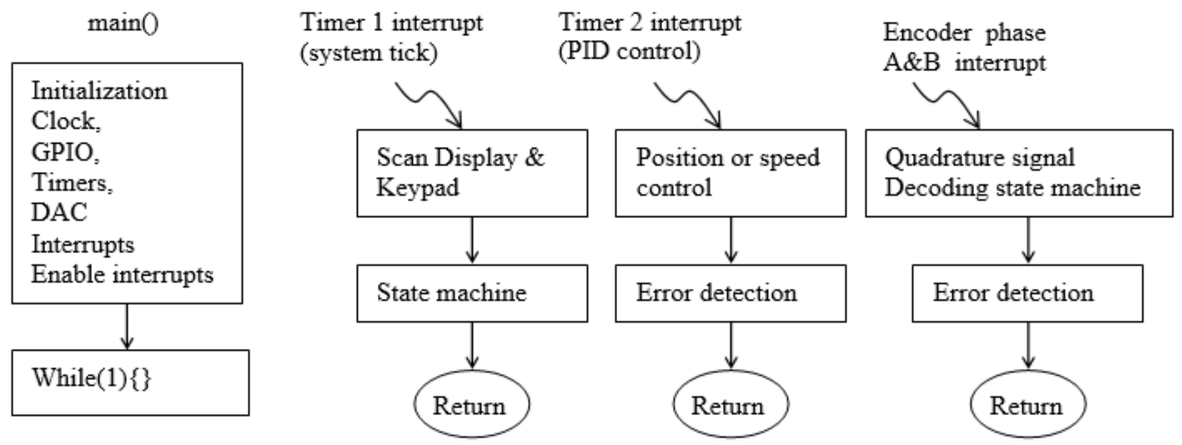
The state is mainly steered by the parameter Goal State which is set by the user interface routine. Goal State can also be set by a fault condition.

State	Name	State Operation	Exit condition	Exit operation	Next state
1	Idle	NOP, motor off	Goal = 3	Set timer	2
			Goal = 4	Speed cmd=0	4
			Goal = 0	Motor off	0
2	Set Ref.	Slow speed, search home position	Goal = 3 & home flag =1	Pos cmd=0;	3
			Timer time out	Error code = 1 (fail to init) Goal state =0	0
			Goal = 0	Set timer	5
3	Position	Position control	Goal = 1	Motor off	1
			Goal = 4	Speed cmd =0	4
			Goal = 0	Motor off	0
4	Speed	Speed control	Goal = 1	Set timer	5
			Goal = 3	Set timer	5
			Goal = 0	Motor off	0
5	Break	Speed control W zero speed	Goal = 1 & speed=0	Motor off	1
			Goal = 3 & speed= 0	Set timer	2
			Timer time out	Error code = 3	0

			Goal = 0	Motor off	0
0	Error		Reset	Clear error	1

Program structure:

The main.c should consist of only an initialization portion followed by an infinite dummy loop. There should be two timer generated interrupts. One of the timer interrupt is for the display and keypad interface and the state machine. This ISR should have lowest priority and should be set at no less than 50Hz. The other timer interrupt is for the position/speed control algorithm. This ISR should have a median priority level and should be set to 100Hz~200Hz.Encoder quadrature signal should be decoded by encoder signal generated interrupt via EIC. This interrupt should have highest priority.



There is no explicit communication between any of these functions. All communication is done by the state variable. For example, the position/speed control ISR changes its control mode strictly base on the current value of the state variable. The ISR detects high error condition and, when detected, the ISR set the state variable value to 0 and that forces the state machine going into the Error state.

Peripheral and Coding:

```
Address        -     SAMD20 Syntax Code
---------------------------------------------------------------------------------------------------------------------------
0x40001800     -     EIC              // address for external interrupt
offset 0x00    -     CTRL.reg         // enable. side note: needs synchronization to write to
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]

Required Tasks:

***please comment your code**

You will get a passing grade if you completely achieved the following tasks.

(1) Keypad will select the control mode : Position/Speed/Idle mode.
(2) Real-time display of the speed (in RPM) and position (in degree) in the corresponding mode
- Max speed is about 4,500 rpm.
- To determine the direction the Neg. LED (PB09) will be used

(3) The potentiometer controls the position or speed. The control must be a **closed loop control**.
(4) Interrupt and state-machine based program structure.

The following requirement is required for an 'A' grade.

(1) Keyboard entry speed and position command.
(2) A display mode control key scrolls the display modes: commanded value display, measured value display mode, and control mode display (idle, pos, or spd).
(3) Position initialization by a home position key.
(4) Slew rate limited controlled for both speed and position control.
(5) Real time analog output from PA02 of the position/speed of the motor(so we can observe the actual position or speed with a scope).

Extra credit for the following tasks *(doing these tasks do not guarantee max extra credit)*

(1) Error detection and code reporting.
(2) Modeling and analysis of the closed loop dynamics including the PID control law (modeling can be done with a modeling program like LtSpice).

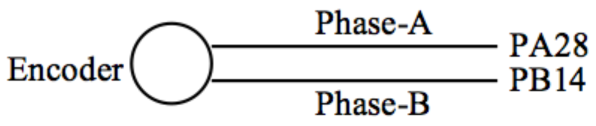
Suggested timeline:

Week 1: Set the TC interrupts needed to interrupt at the frequency required above. Check if they are set properly by OUTTGL a pin and observe the toggling frequency.

Start to read over EIC.

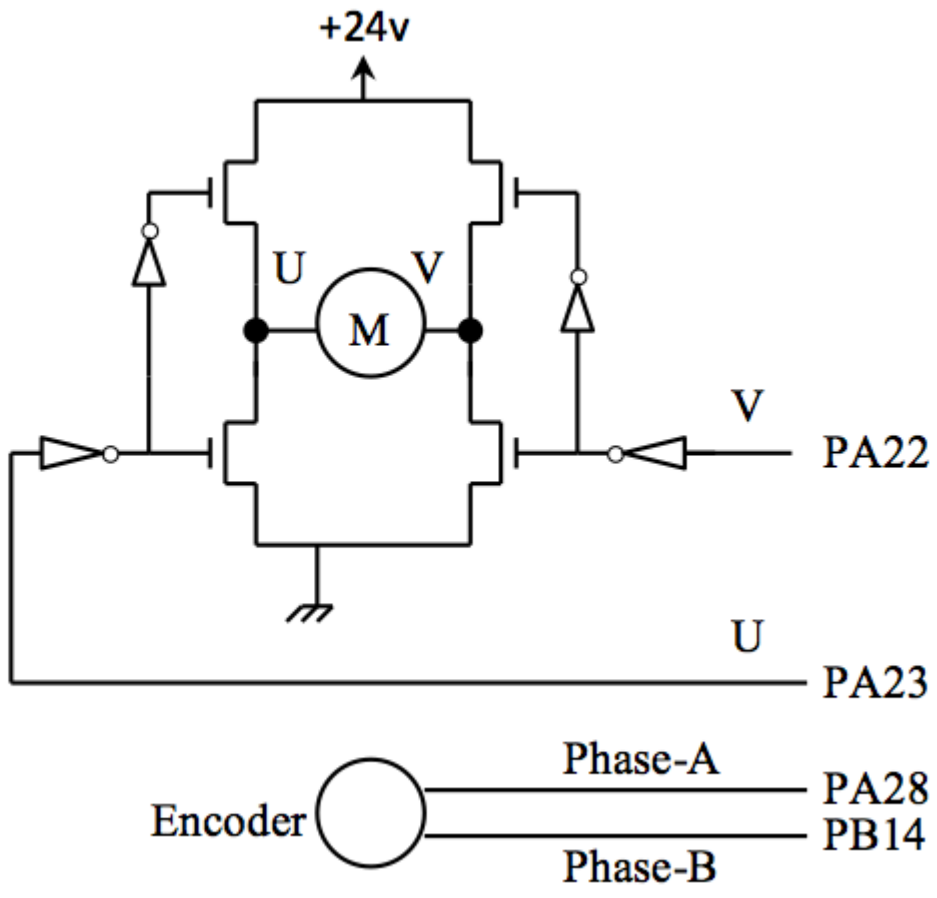
Week 2: Set EIC to trigger on the rising or falling edge (not both) on one of the two Phases from the encoder. Increment a counter and display the counter onto the seven segment display to verify that it is triggering.(***hint:** this test should give you 400 for one revolution*)

Then have the EIC trigger on both rising and falling edges on one Phase and verify.(***hint:** count of 800*)
Then have the EIC trigger for both Phase A and Phase B and verify. (***hint:** count of 1600*)

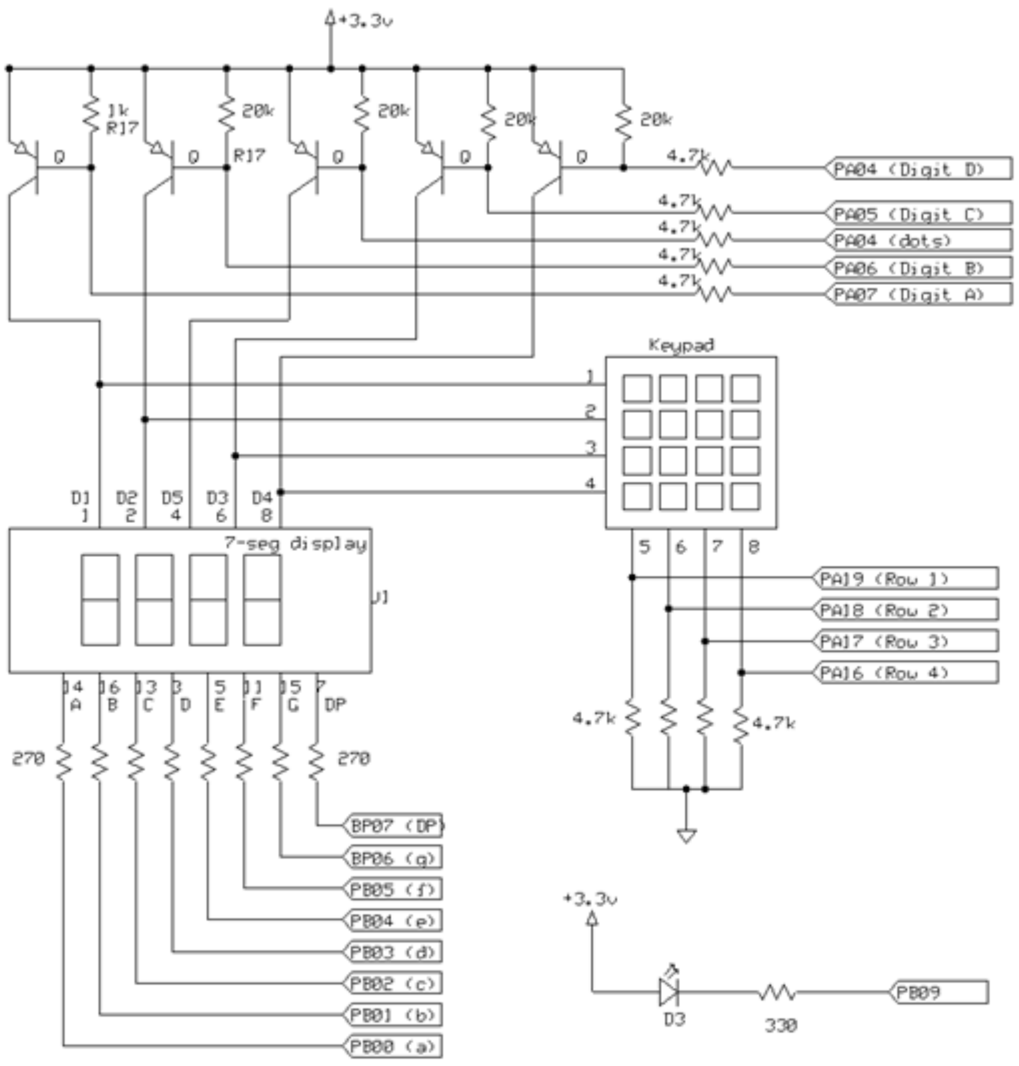


Week 3: Incorporate both Week 1 and Week 2 into one code. Verify that it is working by spinning the motor at a set speed and displaying the count value from the EIC code at a set interval using one of the TCs. Then increase or decrease the speed of the motor and observe the change of the counter displayed.

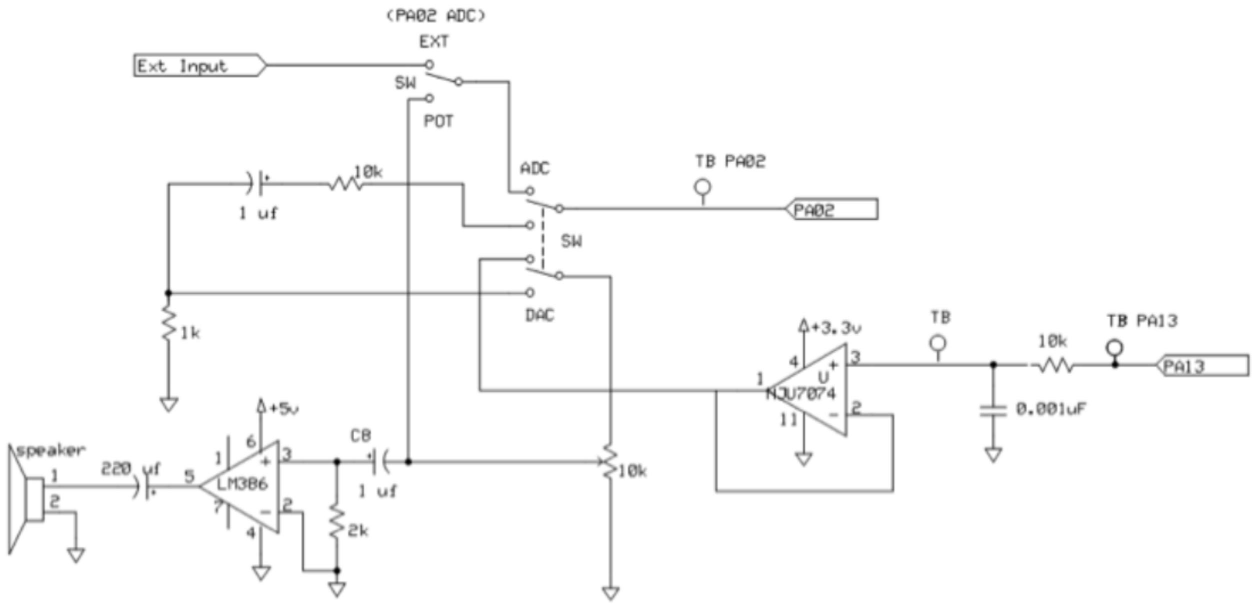
Week 4: Organize code into states and incorporate keypad and ADC POT code from previous labs.



Motor driver and Encoder



Keypad, 7-segment display, and Neg LED



Lab 2 Circuit