Assignment 3 PWM, Cooling Fan Control and Pulse Measurement – CSE 325, Fall 2010

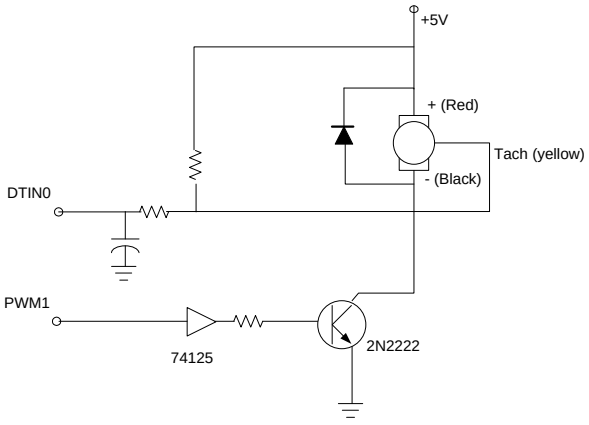
Assignment Objectives

1. To learn software development on the embedded M5211DEMO board
2. To learn event driven embedded software with interrupt service routines
3. To configure pad signals for multiplexed IO pins of microprocessors
4. To generate PWM signals using a hardware timer for fan speed control.
5. To measure pulse width for fan speed monitoring.

DC motors have found many applications in computer and communication devices. Examples include vibrators in cell phones and cooling fans in PCs. For instance, the following vibrator is designed for pagers and cell phones, and costs less than $2.

The operation of a DC motor is based on the magnetic field generated when an electrical current flows in an inductor. Hence, by controlling the current flowing through the motor, we can adjust the motor speed. This can be done by varying the driving voltage or switching on and off the current to the motor using a pulse width modulator.

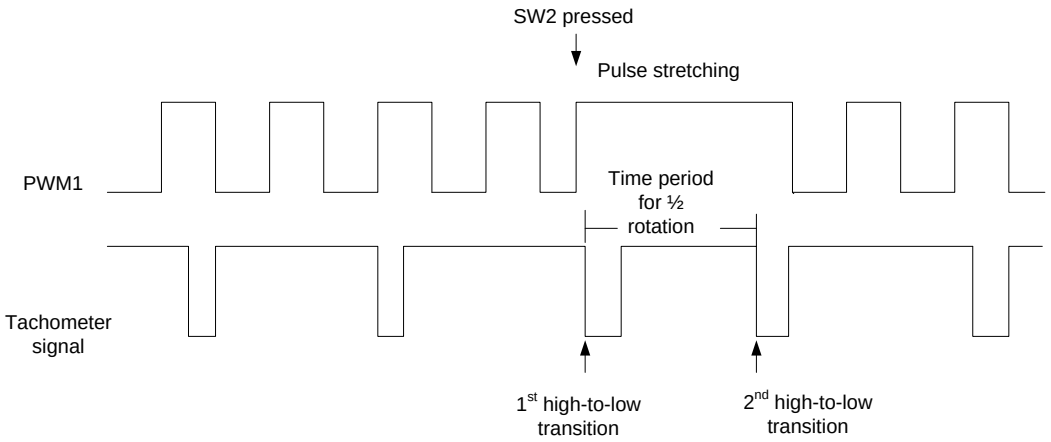
In this project, you will develop a program to control the speed of a 3-wire 5V cooling fan. The three wires are positive (red), negative (black) and tachometer signal (yellow). The motor will be driven by a NPN transistor which switches on and off based on a PWM signal from Coldfire microprocessor, i.e. PWM1. The tachometer signal, after passing a filter, will be read at pin DTIN0 of Coldfire processor. The schematic diagram is shown in the follow diagram.



The assignment asks you to develop a program that sets the motor speed with a PWM signal. The PWM signal starts with a 0% duty cycle, and the duty cycle will be incremented by

20% whenever SW1 switch is pressed. After reaching 100%, the duty cycle will be reset to 0% on the next press of SW1. This sequence should continue on every press of SW1.

If SW2 is pressed, your program should read the speed of motor rotation based on the pulse signal on the tachometer line. When power is on, there will be two pulses for each rotation. When power is off, no pulse signal will appear. Hence, to receive the correct internal of ½ rotations, we should stretch the pulse as shown in the following figure. This pulse stretching should end after we observe (detect) two consecutive high-to-low transitions. The period between the two transitions indicates the time needed for ½ rotations. The motor speed in terms of RPM can then be calculated.



The hardware settings for the project are as follows:

1. When you go into the lab, the M5211DEMO boards should already be plugged into the J5 port of the project board.
2. The AC adapter is connected to the project board and the project board is set up to run with 3.3 volt from on-board voltage regulator.
3. The demo board receives DC voltage from the project board (through M5-J1 connector). No power will be drained from USB.
4. PWM1 is connected to pins 13 of M5-J1 connector (i.e. GPT0). DTIN0 connected to pins 34 of M5-J1 connector.
5. The DC fans, 74HC125s, 2N2222 transistors, resistors and capacitors are left in a box. You will need to wire them on the breadboard of the prototype board.

Once you wire the schematic on the breadboard, you can leave it there. However, before you run your program, please make sure the components are connected correctly.

A main program, including interrupt handlers for SW1 and SW2, and a fan_control.h will be given to you. Your task is to complete the project by implementing 4 functions which are defined in fan_control.h:

- init_pwm(int dcycle): to initialize PWM1 to generate PWM signal to a duty cycle of dcycle%. The frequency of the PWM signal should be at 20KHz.

- init_Tach_in(): to initialize DTIN0 for measuring pulse width.
- set_PWM (int dcycle): set PWM1 signal to a duty cycle of dcycle%.
- int read_speed(): read motor speed and return RPM.

Your project, named “fan_control”, should be built for “console debug” target and the system clock should be initialized to 80MHz. You can use several existing routines from 5211DEMO stationary files, including clock_pll (to initialize clock), mcf5xxx_irq_enable (to enable interrupts), and mcf5xxx_set_handler (to set an interrupt handler), etc.