Lab 15 Period Measurement

This laboratory assignment accompanies the book, Embedded Microcomputer Systems: Real Time Interfacing, by Jonathan W. Valvano, published by Brooks-Cole, copyright © 2000.

Goals
- Write a general purpose device driver for period measurement,
- Use the 6812 input capture for measuring period,
- Interface the tachometer of a DC motor.

Review
- Valvano Section 2.7 on device drivers,
- Valvano Section 6.1.4 concerning period measurement,
- Valvano Section 6.4.1 concerning using period measurement to calculate frequency,
- Valvano Section 8.5 concerning transistors, motors and optocouplers,
- Chapter on the input capture/output compare in the Motorola MC68HC812A4 Manual,

Starter files
- INTERP12.C, SCI12H, SCI12A.C, OC3TEST.C, OC3.H, OC3.C

Background

The objective of this problem is to interface the tachometer of a DC motor controller. Your system will measure the speed of the motor in rotations/sec. The interrupt driven serial drivers SCI12A.C will be used to communicate with the PC. You will power the motor with an external power supply, labeled as +12V in Figure 15.1. The speed will be defined as the frequency of the tachometer output in Hz. With 0 to 12 volt applied to the motor, the speed will range from 0 to about 800 Hz. You will estimate the motor speed (**x'**) by measuring the period of a squarewave coming from the tachometer. The 6812 will measure the period using input capture and the software will convert period to frequency. Your system should be able to measure from 50 to 1000 Hz. The desired speed resolution at 400 Hz is 0.1 Hz. Refer to Section 6.4.1 for a discussion of how the frequency resolution is related to the fundamental period resolution. Your system should also be able to handle the stopped motor condition (0 Hz). If the frequency happens to be between 0 and 50 Hz, then your system can return any value between 0 and 50 Hz. An optical isolator (6N139 or IL-5) isolates the motor power from the microcomputer.

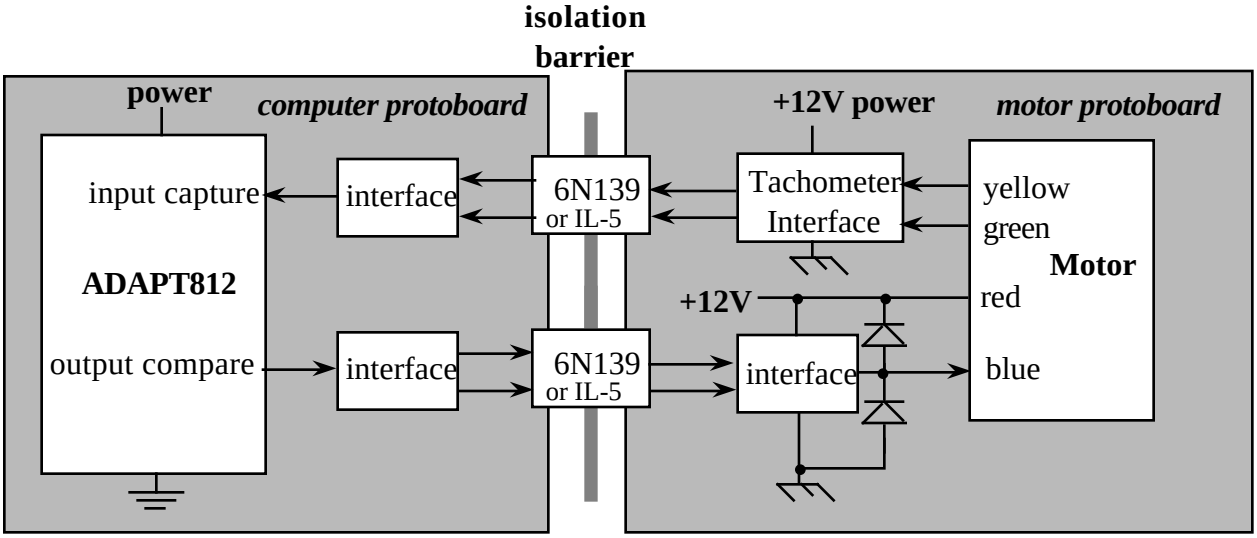

Figure 15.1. Block diagram of the motor tachometer system.

The Buehler motor coil resistance is about 12 Ω, costs about $5.00, and is rated at 6,800 rpm at 12 V. The red/blue wires are used to apply power to the motor, and the yellow/green wires are the tachometer output.

The prototype for the period measurement system should be similar to the following. Feel free to adjust the names and parameters as you wish, as long as you create a general-purpose device driver. Create **Period.h** and **Period.c** files to hold the prototypes and implementations for the period measurement device driver.

```
int Period_Open(unsigned char channel, void(*fp)(unsigned short));
```

`Period_Open` will open a period measurement device. The `channel` will specify to which timer pin the signal is connected. `channel` can be 3 or 4. Although you will be using just one, make the driver flexible to handle

up to two. The other channels are reserved for other uses, like output compare. The function `fp` will be executed on every period measurement. The function `fp` will be passed the most recent measurement, as an unsigned short. This function `Period_Open` will return a true if successful, and a false if not successful. `Period_Open` will fail on an illegal channel, or if the device is already open. The channel will be armed, but interrupts will not be enabled until the user calls `Period_Set_Resolution`. The precision of the period measurement system will be 16 bits. The period resolution will be set by the function `Period_Set_Resolution`. You will use a very simple TOF overflow interrupt scheme to detect the no period condition that occurs when the motor is stopped. If you get two TOF interrupts without any input capture, then you will assume there is no period. In this situation, you should call the user function `fp` with a 0xFFFF parameter. You may wish to compare your simple solution to the 32-bit period measurement example in Chapter 6, but you need not implement a full 32-bit solution.

```
int Period_Set_Resolution(unsigned short nstime)
```

`Period_Set_Resolution` will set the period measurement resolution for all the channels. The parameter `nstime` is the desired period measurement resolution. For the MC68HC812A4 it can be 250, 500, 1000, 2000 or 4000. This function will enable interrupts and return a true if successful. It will return a false if no devices are open or the parameter is illegal.

```
int Period_Close(unsigned char channel);
```

`Period_Close` will close a period measurement device. It will disarm the channel. This function will return a true if successful, and a false if not successful. It will fail if the device is not already open. Interrupts will not be disabled.

Preparation (do this before your lab period)

Show the hardware interface between tachometer and the Adapt812 project board. The tachometer signal varies in both frequency and amplitude with the speed of the motor. The first hardware stage is to reduce amplitude to the 0 to +12 V range. The next step is to create a binary signal using hysteresis. This signal will activate/deactivate the LED stage of the 6N139 or IL-5. The output of the optocouple should be interfaced to a 6812 input capture pin. The hysteresis will reject jitter noise. You must use the electrical isolation circuits developed in a previous lab. You will extend the 6N139 isolation so that the signals are isolated in both directions.

Show the entire interface between the Adapt812 board and the DC motor. Use an external +12 V supply to power the motor. You will use pulse-width modulation using a variable duty-cycle wave, from the previous lab.

Include software listing in the preparation. Organize into modules, so that the debugging can be done piece by piece. Write a main program that allows to TA to vary the power to the motor, and measure the speed in Hz. A "syntax-error-free" hardcopy listing for the software is required as preparation. The TA will check off your listing at the beginning of the lab period. You are required to do your editing before lab. The debugging will be done during lab. Document clearly the operation of the routines.

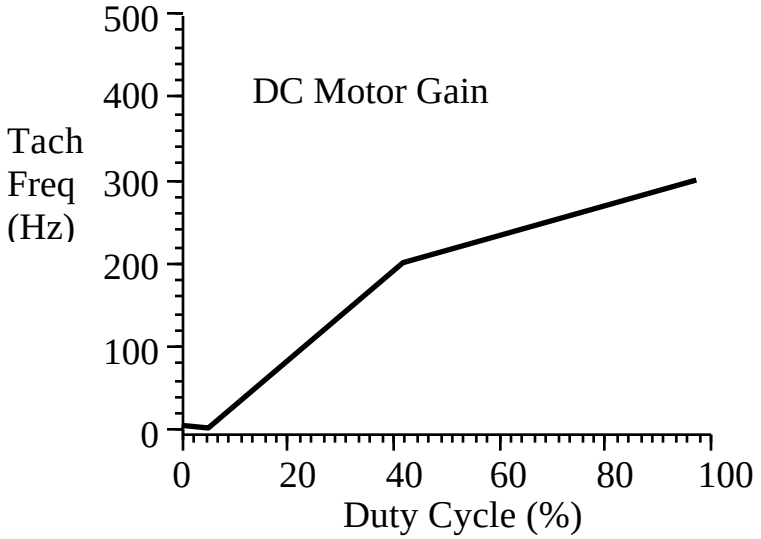

Figure 15.2. The speed of the motor is a function of the duty cycle of its input voltage (at +5V).

Procedure (do this during your lab period)

First you will measure the static response of the motor: the tachometer output, **x(t)**, (in Hz) versus applied power, **p(t)**, (in watts) response of the motor under a no load condition (nothing attached to the shaft.) Measure the shaft speed with your 6812 system. Take 10 measurements with the full range of duty cycles. Show both the raw measurements (duty cycle to the motor, voltage to the motor, current to the motor, period from the tachometer) and the calculated data (motor impedance, applied motor power, tachometer frequency) in a table. Fit the data to the linear equation:

$$\mathbf{x(t) = m\ p(t)}$$

Include a plot of the tachometer frequency versus applied power data with the linear fit. You should get something like Figure 15.2. At +12 V activation, it will run much faster.

Checkout (show this to the TA)

1. Demonstrate your hardware and software. Discuss the limitations. What happens when the motor is stopped?
2. Show your software listings to the TA. Discuss the steps if you had to convert the system from a 6812 to a 6811. What would change? What would remain the same?

Hints/clarifications

1. Be very careful to include the snubber diodes to remove back EMF. Please observe all voltages on the scope before connecting to the microcomputer.

2. For this lab you will use the pulse width modulation (PWM) hardware and software that you created in a previous lab. You will also need to design an analog interface circuit to convert the tachometer output to a CMOS compatible digital signal (square wave). Note that the tachometer output at 1200 Hz can reach 40 volts peak to peak.

3. Please add optical isolation between the tachometer circuit and the 6812. The 6812 ground and the motor ground must not be connected on your circuit. You can not use the 6812 +5V supply to power components on the motor-side of the circuit.

4. The following example illustrates how your software could be used. See OC3.C, OC3.H, OC3TEST.c for a similar example of a general purpose timer system. Assume the signal is connected to PT4. You could add other operations, such as setting and changing the duty-cycle. Remember, it is VERY bad style to perform SCI output (e.g., `OutUDec, OutString`) inside the device driver (in **Period.c**). It is just plain WRONG to perform SCI output from a background thread (e.g., `CalcSpeed`).

```
#include "Period.h"
#include "SCI12.h"
unsigned short Speed; // motor speed in rps
void CalcSpeed(unsigned short period){
  Speed=100000L/period; // I made this up, it isn't correct, you fix it
}
void main(void){ int runFlag=1; // clear to zero to stop
  Period_Open(4, &CalcSpeed);
// signal connected to PT4, call CalcSpeed on each IC
  Period_Set_Resolution(1000);  // period measured in usec
  InitSCI();
  while(runFlag){
// foreground functions
    if(InChar()=='q'){
      runFlag=0;
    }
    else{
      OutUDec(Speed); OutChar(13);
    }
  }
  Period_Close(4);
}
#include "Period.c"
#include "SCI12A.c"
```