GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

ECE 4550 — Control System Design — Fall 2017

Lab #5: PWM Actuation and QEP Sensing for DC Motors

Contents

1 Background Material

1.1 Introductory Comments

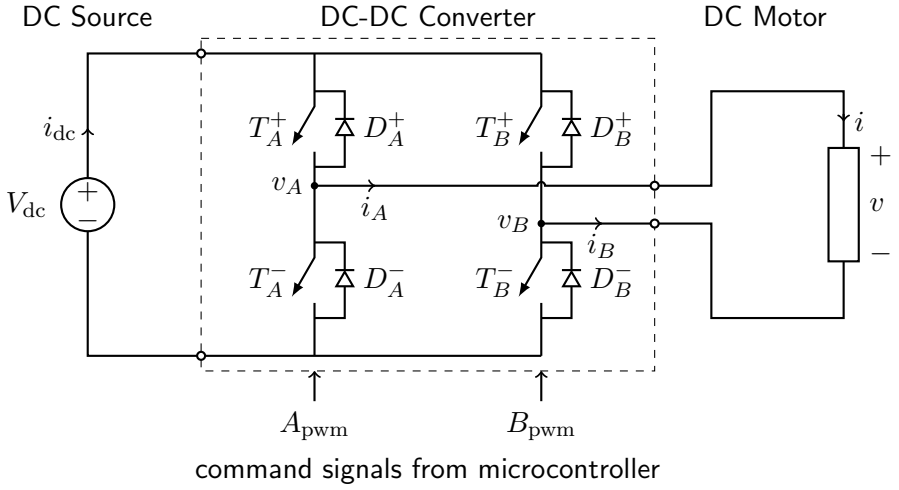
The objective of this lab is to learn the principles of operation of the voltage actuator and the position sensor used in DC motor control systems, and also to learn how these plant components are interfaced to the microcontroller using the PWM and QEP peripheral modules. This lab represents

© 2017 by David G. Taylor. All rights reserved.

a significant transition towards motion control applications and, as a consequence, we will be using the motor control motherboard instead of the peripheral explorer motherboard. Although the motor control motherboard has been designed for relatively low-power motion control applications, it is important to realize that all the concepts being taught in this lab apply equally well to other applications requiring much larger power flows; the technology involved is essentially unchanged.

1.2 PWM Voltage Actuation System

A full-bridge DC-to-DC converter is an actuator used to apply a desired voltage, in the average sense, to a DC motor. This control-enabling circuit incorporates four transistor switches and four diodes as shown below. In this depiction, the third control terminal of each transistor switch is not explicitly displayed. Power is assumed to be supplied by an ideal DC voltage source, and the DC motor is connected across the midpoints of the two converter legs, also known as half bridges. The left leg is called leg A, and its output node has voltage v_A measured with respect to the negative supply terminal. The right leg is called leg B, and its output node has voltage v_B measured with respect to the negative supply terminal. The motor voltage v is determined by $v = v_A - v_B$.



The converter achieves its desired function using switched-mode operation, i.e. the transistors are operated as switches rather than as linear amplifiers. If a transistor is *turned on* then it functions as an approximate *short circuit*, with ideally zero voltage across its terminals (through which the current may be large). If a transistor is *turned off* then it functions as an approximate *open circuit*, with ideally zero current through its terminals (across which the voltage may be large). When the transistors change from one state to another, there will be only a very short time interval over which voltage and current are simultaneously large. Since the power dissipated as heat—the product of voltage and current—is limited even when the level of power being processed is large, switched-mode operation provides advantages with respect to efficiency, cost, mass and volume.

One consequence of switched-mode operation is that, on an instantaneous basis, the converter can only provide three possible motor voltages: $v = \pm V_{\rm dc}$ or $v = 0$. However, since the transfer function relating motor voltage v to motor current i has a low-pass characteristic, it is possible to do a good job of controlling the *instantaneous* current i by merely controlling the *average* voltage $\bar{v}$; the only flaw in this strategy will be a ripple component in the resulting current, but the magnitude of this ripple component will be small if the switching frequency is sufficiently high. The transistor switches of each converter leg must be operated in complementary fashion to avoid short-circuiting the supply; consequently, each converter leg is operated according to a single logic signal.

© 2017 by David G. Taylor. All rights reserved.

Operation of converter leg A via command signal A_{pwm} is summarized as follows:

1. $A_{\text{pwm}} = 1$ implies that $v_A = V_{\text{dc}}$, with the following power device operating states:
   (a) T_A^+ is a short circuit, T_A^- is an open circuit, D_A^- is an open circuit;
   (b) T_A^+ conducts if $i > 0$, D_A^+ conducts if $i < 0$.
2. $A_{\text{pwm}} = 0$ implies that $v_A = 0$, with the following power device operating states:
   (a) T_A^- is a short circuit, T_A^+ is an open circuit, D_A^+ is an open circuit;
   (b) T_A^- conducts if $i < 0$, D_A^- conducts if $i > 0$.

Operation of converter leg B via command signal B_{pwm} is summarized as follows:

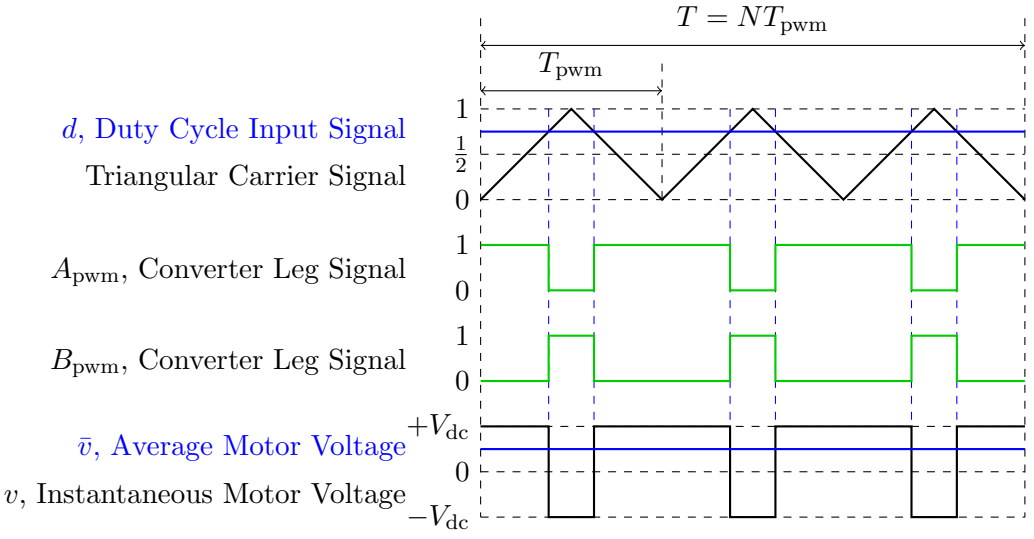
1. $B_{\text{pwm}} = 1$ implies that $v_B = V_{\text{dc}}$, with the following power device operating states:
   (a) T_B^+ is a short circuit, T_B^- is an open circuit, D_B^- is an open circuit;
   (b) T_B^+ conducts if $i < 0$, D_B^+ conducts if $i > 0$.
2. $B_{\text{pwm}} = 0$ implies that $v_B = 0$, with the following power device operating states:
   (a) T_B^- is a short circuit, T_B^+ is an open circuit, D_B^+ is an open circuit;
   (b) T_B^- conducts if $i > 0$, D_B^- conducts if $i < 0$.

Motor voltage v is related to command signals A_{pwm} and B_{pwm} as follows:

1. If $A_{\text{pwm}} = 1$ and $B_{\text{pwm}} = 0$, then $v = +V_{\text{dc}}$ (maximum-magnitude positive voltage).
2. If $A_{\text{pwm}} = 0$ and $B_{\text{pwm}} = 1$, then $v = -V_{\text{dc}}$ (maximum-magnitude negative voltage).

Pulse-width modulation is a common strategy for establishing a desired *average* motor voltage. For example, consider the bipolar PWM switching logic which uses just the two nonzero output voltage values. Since just two distinct motor voltages are available on an instantaneous basis, PWM is employed within each *switching interval* of length T_{pwm}. Control is enabled by adjusting the *duty cycle* of the converter, a value between 0 and 1 denoted by d, which can be modified at the beginning of each *control interval* of length $T = NT_{\text{pwm}}$, where N is typically an integer.

A depiction of the kth control interval is shown below (for the case $d = 0.75$ and $N = 3$):



© 2017 by David G. Taylor. All rights reserved.

Assuming that there is an integer number of switching intervals of length T_{pwm} per control interval of length T, as shown in the diagram above, the average motor voltage over the kth control interval—denoted by $\bar{v}[k]$—is given by (derive this yourself)

$$\bar{v}[k] = \frac{1}{T}\int_{kT}^{kT+T} v(t)\ dt = d[k](+V_{\text{dc}}) + (1-d[k])(-V_{\text{dc}}) = V_{\text{dc}}\,(2d[k]-1)$$

where $d[k]$ denotes the constant duty cycle assigned to the kth control interval. This result implies that $d[k] = 0.5$ or 50% corresponds to $\bar{v}[k] = 0$ V, which makes sense; if we apply $+V_{\text{dc}}$ half the time and $-V_{\text{dc}}$ half the time, then on average we are applying a voltage of zero. The maximum average voltage is $\bar{v}[k] = +V_{\text{dc}}$ which corresponds to $d[k] = 1$ or 100%, and the minimum average voltage is $\bar{v}[k] = -V_{\text{dc}}$ which corresponds to $d[k] = 0$ or 0%. Inversion of this relation yields

$$d[k] = \frac{1}{2}\left(1 + \frac{\bar{v}[k]}{V_{\text{dc}}}\right), \quad -V_{\text{dc}} \le \bar{v}[k] \le +V_{\text{dc}}$$

which provides the duty cycle required to impose some desired average voltage. The equations provided above are idealized, in the sense that they neglect parasitic phenomena such as transistor on-state resistance, diode on-state voltage drop and current sensing resistance; nevertheless, the idealized equations are generally adequate for control purposes, as we shall see.

For DC motor control, we will synthesize the leg command signals A_{pwm} and B_{pwm} from the duty cycle command signal d using the logic displayed above. A duty cycle value—shown as a constant signal—is compared to a triangular waveform with period T_{pwm}. If the duty cycle value (blue) is greater than the triangular waveform value (black), then the leg-A command signal (green) is high and the leg-B command signal (green) is low, resulting in motor voltage $v = +V_{\text{dc}}$. On the other hand, if the duty cycle value (blue) is less than the triangular waveform value (black), then the leg-A command signal (green) is low and the leg-B command signal (green) is high, resulting in motor voltage $v = -V_{\text{dc}}$. The scaling of the two signals being compared will be implementation dependent, but the duty cycle—which by definition is between 0 and 1—will always be obtained as the ratio of the command signal value to the carrier signal range.

The above synthesis of leg command signals is referred to as symmetric PWM; the use of a triangular reference waveform results in leg command signals that exhibit symmetry with respect to the boundaries of each switching interval. An alternative synthesis, known as asymmetric PWM, uses a sawtooth reference waveform instead of a triangular reference waveform. For motor control, we will use symmetric PWM exclusively, since it simplifies the task of sampling motor current for over-current protection or improved control (topics beyond the scope of these labs).

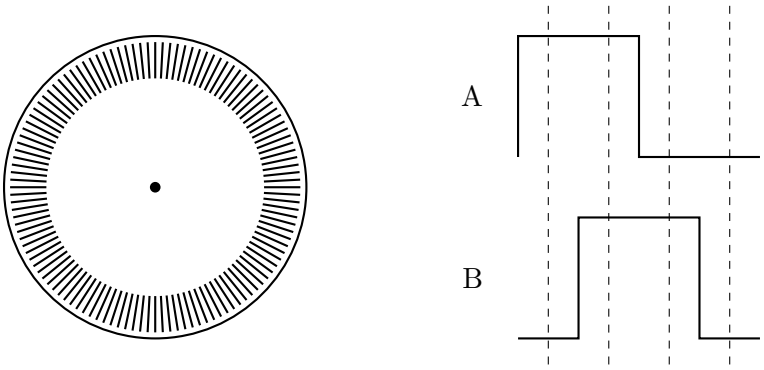
Equivalent interpretations of duty cycle $0 < d < 1$:

1. d is the fraction of time interval T_{pwm} that input signal is greater than carrier signal.
2. $1 - d$ is the fraction of time interval T_{pwm} that input signal is less than carrier signal.
3. d is the fraction of time interval T_{pwm} that legs operate with $A_{\text{pwm}} = 1$ and $B_{\text{pwm}} = 0$.
4. $1 - d$ is the fraction of time interval T_{pwm} that legs operate with $A_{\text{pwm}} = 0$ and $B_{\text{pwm}} = 1$.
5. d is the fraction of time interval T_{pwm} that instantaneous motor voltage is $v = +V_{\text{dc}}$.
6. $1 - d$ is the fraction of time interval T_{pwm} that instantaneous motor voltage is $v = -V_{\text{dc}}$.

© 2017 by David G. Taylor. All rights reserved.

1.3 QEP Position Sensing System

A quadrature encoder is a sensor for measuring the angular position of the rotor. This sensor, which may make use of either optical or magnetic sensing principles, is composed of two parts; a lined disk that is attached to the rotor shaft, and an electronic circuit attached to the stator frame that is capable of detecting rotation of the lined disk. Two separate sensing channels, A and B, are present; these channels are spatially displaced by $\frac{1}{4}$ of the angular pitch between two adjacent lines on the disk and, consequently, they are said to be in phase quadrature. As the rotor rotates, the two quadrature outputs of the optical or magnetic detector circuitry, A and B, will periodically change states due to the lines on the rotating disk. Note that there are four detectable edges in one cycle of the outputs, implying that a quadrature encoder with x lines on its disk will resolve $4x$ possible angular positions within one full revolution.



The digital outputs provided by the quadrature encoder are not directly usable, since they represent motion in an encoded form; the QEP peripheral module has hardware to decode the motion information. If the outputs are presently reading 00, then rotation in one direction will trigger a change to 01 and rotation in the other direction will trigger a change to 10. The decoder hardware operates on the basis of the state transition rules listed below. Given the present state of the two outputs, transition to one set of new outputs will increment a position counter, whereas transition to another set of new outputs will decrement the counter. Transitions between certain outputs, such as between 00 and 11, are not possible and would be interpreted as a fault. The angular position of the rotor, θ, is related to the variable counter value according to

$$\theta = \text{counter value} \times \left(\frac{\text{radians}}{\text{increment}} \right).$$

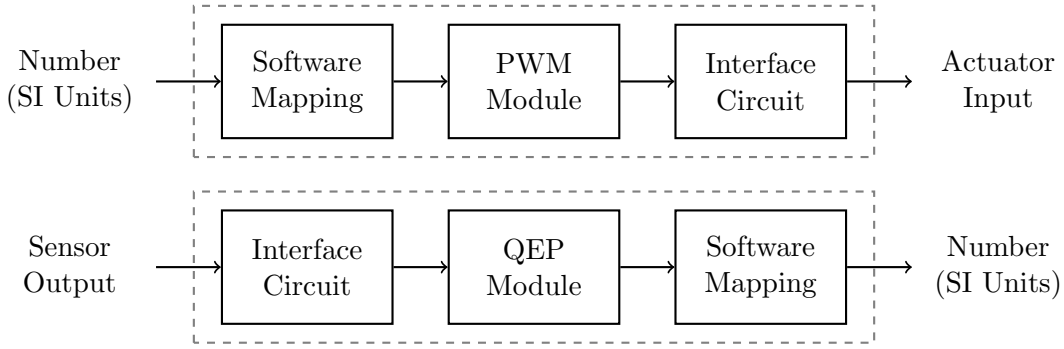
		new AB			
		00	01	10	11
old AB	00	0	−1	+1	×
	01	+1	0	×	−1
	10	−1	×	0	+1
	11	×	+1	−1	0

counter actions (× = fault):
increment if +1
decrement if −1
no change if 0

© 2017 by David G. Taylor. All rights reserved.

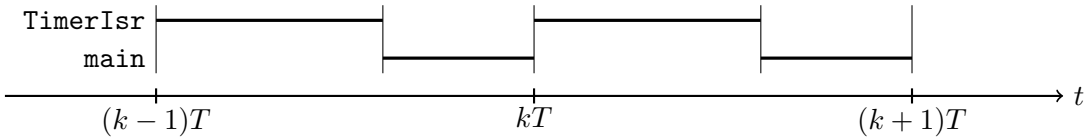
1.4 Variations on Interrupt-Based Code Flow

As you contemplate your microcontroller code development, keep in mind that the actuating system and the sensing system are related by a reciprocal structure; the actuating system begins with a floating-point command value in SI units, whereas the sensing system ends with a floating-point measurement value in SI units. Both systems require circuitry at the microcontroller pins to interface the internal microcontroller modules to the necessary external components.



A timer interrupt will be used to establish time periodicity for both writing to the actuators and reading from the sensors. If the motor control motherboard is utilized without attempting motor current measurement (as is the case in this lab), then a code flow involving one interrupt service routine would be sufficient, as shown below.

Code Flow for PWM Actuation with QEP Sensing



The tasks performed in the ISR for this implementation would be as follows.

```
interrupt void TimerIsr (void)
{
    // 1. Write out previously stored voltage command.
    // 2. Read in new angular position measurement.
    // 3. Compute and store desired next voltage command.
    // 4. Acknowledge interrupt.
}
```

1.5 Relevant Microcontroller Documentation

The overall objective of these lab projects is to teach you how to do embedded design with microcontrollers in a general sense, not just how to approach one specific application using one specific microcontroller. Therefore, the guidance provided herein focuses more on general thought processes and programming recommendations; step-by-step instructions of an extremely specific nature have been intentionally omitted. Use fundamental documentation as your primary source of information as you work through details of implementation. Being able to read and understand such documentation is an important skill to develop, as similar documentation would need to be consulted in order to use other microcontrollers or other application hardware. By making the

© 2017 by David G. Taylor. All rights reserved.

effort to extract required details from fundamental documentation yourself, you will have developed transferable skills that will serve you well in your engineering career. For this lab, review:

- Technical Reference Manual
  - §3.1, for an overview of the PWM module.
  - §3.2.2, for a description of the time-base submodule.
  - §3.2.3, for a description of the counter-compare submodule.
  - §3.2.4, for a description of the action-qualifier submodule.
  - §3.4.1, for the `TBPRD` and `TBCTL` register field descriptions.
  - §3.4.2, for the `CMPA` and `CMPB` register field descriptions.
  - §3.4.3, for the `AQCTLA` and `AQCTLB` register field descriptions.
  - §7.1, for an overview of the QEP module.
  - §7.3, for a description of the quadrature decoder submodule.
  - §7.4, for a description of the position counter and control submodule.
  - §7.9, for the `QEPCTL`, `QPOSCNT`, `QPOSINIT` and `QPOSMAX` register field descriptions.

1.6 Target Hardware Schematic Diagrams and Data Sheets

- Motor Control Motherboard
  - Page 1, for motor connections and motor driver chip.
  - Page 2, for microcontroller daughtercard and encoder connections.
- Motor Driver Chip
  - Page 1, for an overview of the chip.
  - Page 4, for the pin assignment.
  - Page 10, for system block diagram including connections to half-bridge power stages.
- Quadrature Encoder
  - 250 lines on disk, 1000 counts per revolution (2π radians).
- Power Supply
  - Voltage setting: we will use 24 V.
  - Current limit: we will use 3 A.

This lab uses a newer type of F28069 microcontroller daughtercard that incorporates a JTAG emulator. The motor control motherboard incorporates one motor driver chip to implement three half-bridges. As discussed in §1.2, we will only use two of the three half-bridges. There are reset pins on the motor driver chip for each of the half-bridges; power is enabled on the half-bridges when these GPIO pins are driven high. Furthermore, logic within the motor driver chip permits one PWM signal to operate both switches of one half-bridge in complementary fashion, so only two PWM signals from the microcontroller are required to operate all four switches of the converter, i.e. command signals A_{pwm} and B_{pwm} as described in §1.2.

© 2017 by David G. Taylor. All rights reserved.

The motor-mounted encoder connects to the motor control motherboard using a five-wire cable. The connectors on both ends of this cable are simple five-pin header connectors. Unlike connectors intended for commercial use, these inexpensive connectors do not fit with a unique orientation. Consequently, special care must be used to guarantee that the connectors on both ends of the cable are properly installed; *the equipment will be damaged if you do not install both connectors correctly*. The pin assignment for both connectors is shown below.

- The *green connector* should be connected to the encoder such that its black ground wire matches up with the ground label on the encoder case (pin 1).
- The *black connector* should be connected to the motor control motherboard at J4 such that its black ground wire matches up with pin 5 (the white dot indicates pin 1).

Motor-End of Cable (Green)		*Controller-End of Cable (Black)*	
Pin 1	Ground	Channel A	Pin 1
Pin 2	Index	Channel B	Pin 2
Pin 3	Channel A	Index	Pin 3
Pin 4	Power	Power	Pin 4
Pin 5	Channel B	Ground	Pin 5

2 PWM Module: Step-by-Step Guidelines

There are eight PWM modules within the F28069 microcontroller, each of which is characterized by one counter register, two counter compare registers, and two output pins. Many features of these PWM modules will not be needed for this lab; we will focus on the time base subsystem, the action qualifier subsystem and the counter compare subsystem. Note in particular that the PWM modules will not be used to generate interrupts.

2.1 Initialize the PWM Registers

2.1.1 Set the Pin Multiplexer

Microcontroller pins that have the possibility of being used as PWM module outputs are multiplexed pins. Therefore, assignment of these pins for PWM module use requires the appropriate setting of the corresponding multiplexers. Once the pin in question has been identified—in this case by consulting the motor control motherboard schematic diagram—the appropriate `GPIOz` field of the appropriate `GPxMUXy` register is used to set the multiplexer as desired. The HAL-defined variable names available for initializing these register fields are as follows:

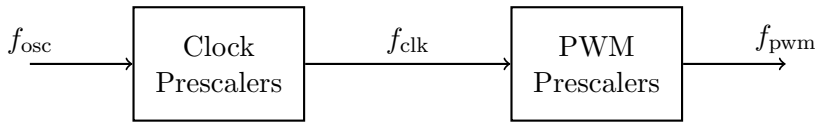
```
GpioCtrlRegs.GPxMUXy.bit.GPIOz
```

2.1.2 Enable the Module Clock

All of the microcontroller's digital modules operate on the basis of clock signals. Since various modules serve distinct purposes, it is generally necessary for them to operate at different frequencies. The PWM module is fed by the system clock signal at frequency f_{clk}, but an internally derived clock signal determines the module's counting rate; this internal clock signal is not explicitly displayed in the diagram below, but it is referred to as `TBCLK` in Technical Reference Manual and its frequency is related to f_{clk} by two prescale factors described below. Note that the frequency of

© 2017 by David G. Taylor. All rights reserved.

TBCLK represents the rate with which low-level counting operations take place inside the module, not the rate with which output voltage switching cycles occur; our main interest is in the latter rate f_{pwm}, which is programmed using a total of three parameters as described below.



There are eight separate PWM modules, and their module clock feed paths are individually enabled using the EPWMxENCLK fields of the PCLKCR1 register. It is essential to enable a module's clock prior to attempting to configure that module, since otherwise it would not be possible to successfully write to the configuration registers of that module. This initial clock-enable step feeds the modules with SYSCLKOUT but does not enable TBCLK internally, which is fine for now. The HAL-defined variable names available for accessing these register fields are as follows:

```
SysCtrlRegs.PCLKCR1.bit.EPWMxENCLK
```

2.1.3 Configure the Time Base Clock and Counter

The configuration of the time base clock and counter must necessarily take into account the choice of counter mode and counter maximum value. As previously described, we will be using symmetric PWM with a triangular carrier waveform, which implies up-down counting from zero up to some preset maximum value and then back down to zero. This mode of operation is configured by assigning the counter mode field CTRMODE of the time base control register TBCTL. The HAL-defined variable name available for accessing this register field is as follows:

```
EPwmxRegs.TBCTL.bit.CTRMODE
```

The counter maximum value, referred to as TBPRD in Technical Reference Manual, is selected so as to provide the desired resolution for the output voltage waveform. In this application, the output voltage is considered to be the average motor voltage $-V_{\text{dc}} \leq \bar{v} \leq +V_{\text{dc}}$. Since the counter increments TBPRD times from its minimum value to its maximum value, each counter increment corresponds to an output voltage increment equal to

$$\bar{v}_{\text{res}} = \frac{\bar{v}_{\text{max}} - \bar{v}_{\text{min}}}{\texttt{TBPRD}} = \frac{2V_{\text{dc}}}{\texttt{TBPRD}}$$

where the expression given on the right indicates motor voltage resolution specialized to the present application. A larger TBPRD reduces output voltage magnitude quantization, resulting in a smaller increment between two neighboring output voltages (desirable), but it also increases the output voltage switching period for a fixed time base clock frequency, which would result in larger switching ripple in motor current and hence in motor torque (undesirable). The TBPRD value is assigned using the 16-bit time base period register TBPRD. The HAL-defined variable name available for accessing this entire register is as follows:

```
EPwmxRegs.TBPRD
```

The time base clock frequency is selected so as to provide the desired switching frequency for the output voltage waveform. A larger clock frequency reduces the time it takes to complete the up-down counting process and hence the time it takes to complete each switching cycle of the output voltage, resulting in reduced excursions in motor current and motor torque over each switching

© 2017 by David G. Taylor. All rights reserved.

cycle. This general trend holds true irrespective of the particular choice of counter maximum value so, in order to establish a specified output voltage switching cycle frequency equal to f_{pwm}, the time base clock frequency must be scaled proportional to the counter maximum value. The relationship used to configure the time base clock and counter is therefore summarized by

$$f_{\text{pwm}} = \frac{f_{\text{clk}}}{(2 \cdot \texttt{TBPRD})(\texttt{HSPCLKDIV})(\texttt{CLKDIV})}$$

where the factors `HSPCLKDIV` and `CLKDIV` are understood to be the effective values indicated in the register field descriptions rather than the actual field values. The factor $2 \cdot$ `TBPRD` accounts for the number of time base clock cycles needed to complete one up-down counting cycle, whereas the time base clock frequency is determined by the `HSPCLKDIV` and `CLKDIV` fields of the `TBCTL` register. The HAL-defined variable names available for accessing these latter register fields are as follows:

```
EPwmxRegs.TBCTL.bit.HSPCLKDIV

EPwmxRegs.TBCTL.bit.CLKDIV
```

At this point, you will have properly configured the parts of the PWM module responsible for generating the *black triangular carrier signal* shown in the timing diagram of §1.2.[1]

2.1.4 Set the Output Actions

Each PWM module has two output pins associated with it, denoted `A` and `B`, and actions on either or both of these pins may be programmed to take place during the switching cycles by utilizing the action qualifier control registers. More specifically: the `CAU` and `CBU` fields of the `AQCTLy` register of the `EPwmx` module may be used to automatically clear, set or toggle pin `y` of module `x` each time the module counter equals the values in the module compare registers `CMPA` and `CMPB`, provided that the module counter is incrementing (`U` for counting up); the `CAD` and `CBD` fields of the `AQCTLy` register of the `EPwmx` module may be used to automatically clear, set or toggle pin `y` of module `x` each time the module counter equals the values in the module compare registers `CMPA` and `CMPB`, provided that the module counter is decrementing (`D` for counting down). Note that there is no correspondence between the `A` and `B` labels associated with PWM output pins and PWM compare registers. For example, `CMPA` might trigger an action on either pin `EPWMxA` or `EPWMxB`, and also `CMPB` might trigger an action on either pin `EPWMxA` or `EPWMxB`. The HAL-defined variable names available for accessing these register fields are as follows:

```
EPwmxRegs.AQCTLy.bit.CAz

EPwmxRegs.AQCTLy.bit.CBz
```

At this point, you will have properly configured the parts of the PWM module responsible for generating the *green pin logic signals* shown in the timing diagram of §1.2.[2]

2.1.5 Enable the Time Base Clock

Enabling a PWM module's clock signal, as previously discussed, insures that the module receives a clock signal that in turn will allow its configuration registers to be set. However, at run time, it is necessary to clock each active PWM module with a single time base clock signal in a synchronized

[1] See also Figure 3-6 and its accompanying equation in the Technical Reference Manual document.
[2] See also Figures 3-23, 3-27 and 3-28 in the Technical Reference Manual document.

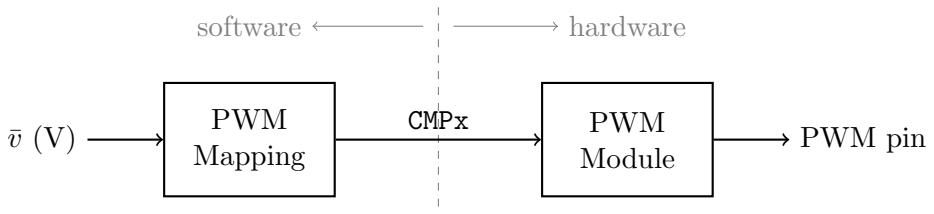
© 2017 by David G. Taylor. All rights reserved.

way. Therefore, after all preliminary configuration of all active PWM modules has been completed, the final configuration step is to enable or feed a synchronized time base clock signal to all active PWM modules. This step—which is required even if only one PWM module is being used—is performed by utilizing the `TBCLKSYNC` field of the `PCLKCR0` register. The HAL-defined variable name available for accessing this register field is as follows:

```
SysCtrlRegs.PCLKCR0.bit.TBCLKSYNC
```

2.2 Utilize the PWM Pins

The diagram below visualizes the signal flow associated with the PWM module. On the hardware side, PWM pins are connected to circuitry within the power converter and within the module itself. On the software side, the desired average output voltage is requested as a floating-point value in SI units. Registers represent the interface between the hardware and software sides; the `CMPx` registers are the means by which the desired average output voltage is requested.



2.2.1 Map and Write the Duty Cycle

The PWM module pin voltages, and ultimately the DC motor average voltage, will be manipulated as shown in the diagrams above. The process begins with a `float32` representation of a desired DC motor average voltage in SI units. The averaging formula provided in §1.2 shows how this average voltage value can be associated with a corresponding duty cycle value, a dimensionless quantity between 0 and 1. In the PWM module, this interval from 0 to 1 is captured by the analogous interval from the counter zero value to the counter maximum value, and the duty cycle value is captured by the counter compare value, a `Uint16` value. The purpose of the software mapping shown in the diagrams above is to cast the desired voltage `float32` value into a corresponding counter compare `Uint16` value. Once the appropriate integer value has been determined, it must then be placed in a counter compare register, `CMPA` or `CMPB`. These register values may be updated within a timer ISR, as necessary, to modify the voltage produced as a function of time.[3] The HAL-defined variable names for accessing these registers are indicated below; the variable names differ between the `EPWMxA` and `EPWMxB` pins, due to the fact that a high-resolution extension (which we don't actually use) is available on the former but not on the latter.

```
EPwmxRegs.CMPA.half.CMPA
```

```
EPwmxRegs.CMPB
```

At this point, you will have properly requested the PWM module to synthesize a desired average output voltage, as represented by the *blue duty cycle signal* in the timing diagram of §1.2.[4]

[3] Initialize the counter compare registers in configuration code so that no voltage will be applied to the motor when the power stage is first enabled; otherwise, uncontrolled excitation will occur prior to the first visit to the timer ISR.

[4] See also Figures 3-23, 3-27 and 3-28 in the Technical Reference Manual document.

© 2017 by David G. Taylor. All rights reserved.

3 QEP Module: Step-by-Step Guidelines

There are two QEP modules within the F28069 microcontroller, each of which is characterized by one decoder, one counter, and two input pins. Many features of these QEP modules will not be needed for this lab; we will focus only on the counter subsystem. Note in particular that the QEP modules will not be used to generate interrupts.

3.1 Initialize the QEP Registers

3.1.1 Set the Pin Multiplexer

Microcontroller pins that have the possibility of being used as QEP module inputs are multiplexed pins. Therefore, assignment of these pins for QEP module use requires the appropriate setting of the corresponding multiplexers. Once the pin in question has been identified—in this case by consulting the motor control motherboard schematic diagram—the appropriate `GPIOz` field of the appropriate `GPxMUXy` register is used to set the multiplexer as desired. The HAL-defined variable names available for initializing these register fields are as follows:

```
GpioCtrlRegs.GPxMUXy.bit.GPIOz
```

3.1.2 Enable the Module Clock

All of the microcontroller's digital modules operate on the basis of clock signals, and it is generally necessary to clock them at different frequencies. The QEP module, however, is clocked at the maximum possible frequency to maximize the motor speed range over which transitions in the quadrature encoder input signals will be properly detected. There are two separate QEP modules, and their module clock feed paths are individually enabled using the `EQEPxENCLK` fields of the `PCLKCR1` register. It is essential to enable a module's clock prior to attempting to configure that module, since otherwise it would not be possible to successfully write to the configuration registers of that module. The HAL-defined variable name available for accessing this register field is as follows:

```
SysCtrlRegs.PCLKCR1.bit.EQEPxENCLK
```

3.1.3 Set the Module Maximum Count

The decoder subsystem decodes the quadrature input signals into direction and pulse signals, and these latter signals govern the operation of the counter subsystem; specifically, the counter will count in the direction determined by the direction signal each time there is a pulse on the pulse signal. Associated with the counter is a programmable maximum count value stored in the `QPOSMAX` register. The `Uint32` value of the counter is, in some sense, a measure of motor position; since we have no reason to place an artificial limit on this measure of motor position, we prefer to use the largest possible value for the maximum count, which is `0xFFFFFFFF`. The HAL-defined variable name available for accessing this register is as follows:

```
EQepxRegs.QPOSMAX
```

Once the `Uint32` value of the counter is read and cast to an `int32` variable as described later, the range of values accommodated by the above assignment is -2^{31} to $+2^{31}-1$. This is clearly a very large range, so we will not need to worry about overflow; however, some applications may require overflow countermeasures to be integrated into the code for reliable operation.

© 2017 by David G. Taylor. All rights reserved.

3.1.4 Enable and Initialize the Module Counter

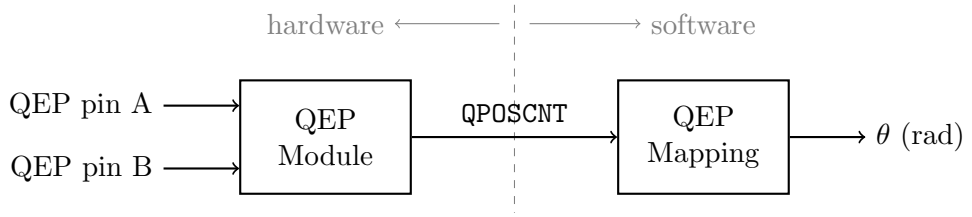
The initial value of the counter is set during counter subsystem initialization; the default value of zero should be satisfactory, but if for some reason a nonzero value is needed, the `QPOSINIT` register is used for that purpose. The counter subsystem is enabled and initialized by setting the `QPEN` and `SWI` bits of the `QEPCTL` register. The HAL-defined variable names available for accessing these register fields are as follows:

```
EQepxRegs.QPOSINIT

EQepxRegs.QEPCTL.bit.QPEN

EQepxRegs.QEPCTL.bit.SWI
```

In position control applications, it is generally necessary to perform a "move to home position" initialization task prior to initializing the counter to zero; e.g. perhaps an open-loop excitation is used to drive the motion apparatus until a trigger is issued from a home position sensing device. In our lab, since the motor is unloaded, we are free to accept the initial position as the zero position.

3.2 Utilize the QEP Pins

The diagram below visualizes the signal flow associated with the QEP module. On the hardware side, QEP pins are connected to the circuitry within the module itself. On the software side, a floating-point representation of the measured motor position in SI units is obtained. Registers represent the interface between the hardware and software sides; the `QPOSCNT` register is the means by which the measured motor position is actually extracted.



3.2.1 Read and Map the Module Counter

The value of the module counter may be read at any time, but this is typically done during periodic visits to a timer ISR. Within this ISR, the counter value is obtained by reading the `QPOSCNT` register. The HAL-defined variable name available for accessing this register is as follows:

```
EQepxRegs.QPOSCNT
```

The `Uint32` value obtained may be cast to an `int32` value, in order to naturally accommodate both positive and negative integer values; failure to perform such a cast would result in discontinuity, transitioning between the largest positive value and the zero value, when the rotor rotates through the zero position (as opposed to a simple change in sign at this position, as desired). The modeling information provided in §1.6 establishes the relationship between one increment of the module counter and the corresponding angular rotation of the motor shaft. A software mapping is utilized to cast the counter value into a `float32` representation of motor position in SI units.

© 2017 by David G. Taylor. All rights reserved.

4 Lab Assignment

4.1 Pre-Lab Preparation

Each individual student must work through the pre-lab activity and prepare a pre-lab deliverable to be submitted *by the beginning of the lab session*. The pre-lab deliverable consists of a brief typed statement, no longer than one page, in response to the following pre-lab activity specification:

1. Read through this entire document, and describe the overall purpose of this week's project.
2. Describe in specific terms how the relevant registers will be used to complete the tasks assigned in §4.2. If you will write to a HAL variable, state the numerical value to be written; if you will read from a HAL variable, state how the numerical value read will be interpreted. You will need to consult circuit diagrams in order to complete this part of your pre-lab preparation.
3. Describe the role of GPIO-1 and GPIO-3. What are these two pins connected to, and how should they be assigned? Consider §8.3.2.4 of the motor driver chip datasheet.

Please note that it is not essential to write application code prior to the lab session; the point of the pre-lab preparation is for you to arrive at the lab session with firm ideas regarding register usage and other relevant issues in relation to the tasks assigned in §4.2.

4.2 Specification of the Assigned Tasks

- The three toggle switches must remain in the center (MCU) position at all times.
- For the first task, *disconnect* the motor from the power converter.
- For the second task, *connect* the motor to the power converter.
- The required configuration settings for both tasks are tabulated below.

CPU clock frequency	f_{clk}	90 MHz
PWM switching frequency	f_{pwm}	30 kHz
Timer interrupt frequency	f_{tmr}	1 kHz
Motor voltage resolution	$\bar{v}_{\text{res}}$	32 mV

4.2.1 Actuating the Desired Input Voltage

Develop code that imposes a desired average value of the motor's input voltage $\bar{v} = \bar{v}_A - \bar{v}_B$. The PWM module should be programmed to perform switching at frequency f_{pwm}, and the desired average voltage should be specified as a `float32` value in SI units in a timer ISR that updates at frequency f_{tmr}. The desired average voltage should be periodic with piecewise-constant segments; each cycle should consist of +20 V for 0.2 s, 0 V for 0.2 s, −20 V for 0.2 s, and 0 V for 0.2 s, in that sequence. Use an oscilloscope, with two probes connected so as to simultaneously measure the two instantaneous voltages v_A and v_B, in order to confirm proper operation.

Instructor Verification (separate page)

© 2017 by David G. Taylor. All rights reserved.

4.2.2 Sensing the Resulting Output Position

Develop code that measures the angular position of the motor's rotor, using the QEP module and the same excitation as above; adopt a sign convention such that positive voltage increases angular position (to eliminate ambiguity arising from the ordering of power-flow connections between motor and power converter). The angular position should be evaluated as a `float32` value in SI units in a timer ISR that updates at frequency f_{tmr}. Use data logging to store the initial 2000 values of both the *desired* average voltage and the *measured* angular position, and export the data. Make a Matlab plot showing these two signals versus time (use `subplot`, include labels and units).

Instructor Verification (separate page)

© 2017 by David G. Taylor. All rights reserved.

GEORGIA INSTITUTE OF TECHNOLOGY
SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

ECE 4550 — Control System Design — Fall 2017

Lab #5: PWM Actuation and QEP Sensing for DC Motors

Instructor Verification Page

Lab Section	Begin Date	End Date
L01, L02	October 3	October 17
L03, L04	October 5	October 19

To be eligible for full credit, do the following:

1. Submissions required by each student (one per student)
   (a) Upload your pre-lab deliverable to tsquare before lab session begins on begin date.
   (b) Upload your `main.c` file for §4.2.2 to tsquare before lab session ends on end date.
   (c) Upload your two data arrays for §4.2.2 to tsquare before lab session ends on end date.
2. Submissions required by each group (one per group)
   (a) Submit a hard-copy of this verification page before lab session ends on end date.
   (b) Attach to this page a hard-copy of the Matlab plot requested in §4.2.2.

Name #1: _____

Name #2: _____

Checkpoint: Verify completion of the task assigned in §4.2.1.

Verified: _____ **Date/Time:** _____

Checkpoint: Verify completion of the task assigned in §4.2.2.

Verified: _____ **Date/Time:** _____