AN3453

Sensored 3-Phase BLDC Motor Control Using Sinusoidal Drive

Introduction

Author: Aldrin Abacan, Microchip Technology Inc.

The sinusoidal current drive has been the one of the most used methods in industrial applications for driving BLDC motors. Compared to the six-step commutation (trapezoidal drive), the sinusoidal current drive provides higher efficiency, lower torque ripple and lower acoustic noise. It is the preferred option by users for low-speed and low-noise motor control systems. For practical applications, both the maintenance and implementation cost are also a consideration in choosing the right motor type and motor controller.

This application note describes how to implement a sinusoidal current drive on a sensored 3-phase BLDC motor setting, using a low-cost 8-bit microcontroller. By using Microchip's Core Independent Peripherals (CIPs), the sinusoidal current drive functions with less processing from the CPU, by reducing the complex firmware mathematical operations required. This application note also highlights the use of the Direct Memory Access (DMA) peripheral to move data from a Sine Look-up Table (LUT) to the PWM peripherals, using unused CPU cycles. By implementing these functions, the CPU gains additional bandwidth to handle other system processes. Figure 1 shows the basic block diagram of implementing sinusoidal current drive using PIC18-Q43 family devices.

Figure 1. 3-Phase BLDC Motor Control Block Diagram

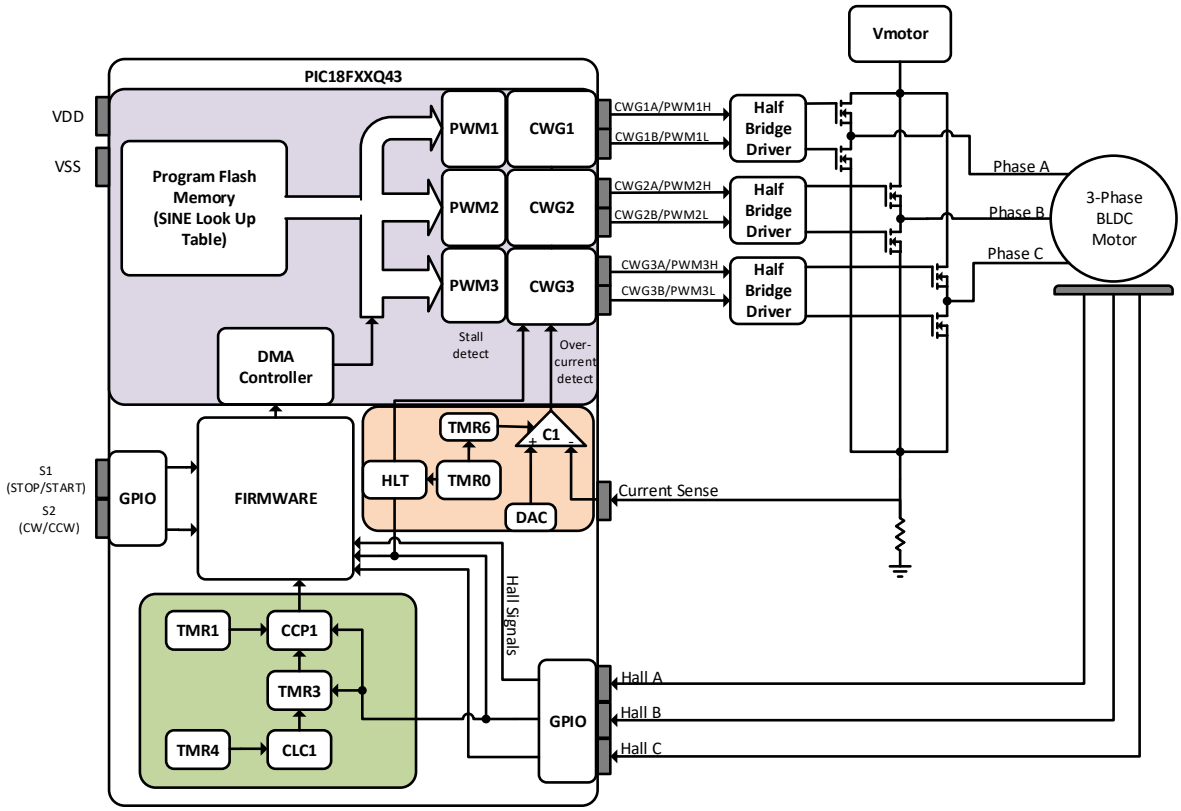


Table of Contents

1. Overview

In this application, the implementation of the sinusoidal current drive on PIC18-Q43 can be divided into three functions; Drive Signal Generator, Angular Position Detector and Fault Detection Control. The drive signal generator provides the necessary signals for the implementation of the sinusoidal current drive. The drive signals produced in this stage are connected to an external MOSFET driver, which is used to directly drive the BLDC motor. The drive signals are used in gating the power supply to provide modulated voltage. The angular position detector provides periodic interrupts based on the motor angular position. These interrupts are used to trigger a change in drive signal applied voltage. The Fault detection controller monitors events such as overcurrent condition and stall condition. Once a Fault is detected, the controller disables the drive signal generator and all the interrupts, which stops the motor from running. The following are the list of peripherals used to successfully perform these functions.

- Drive Signal Generator
  - Direct Memory Access (DMA) controller
  - Complementary Waveform Generator (CWG)
  - 16-Bit Pulse Width Modulation (PWM)
- Angular Position Detector
  - TMR1/3/4
  - Configurable Logic Cell (CLC)
  - Compare (CCP)
- Fault Detection
  - Digital-to-Analog Converter (DAC)
  - Comparator (CMP)
  - TMR0/2/6

Figure 1-1. Control Diagram

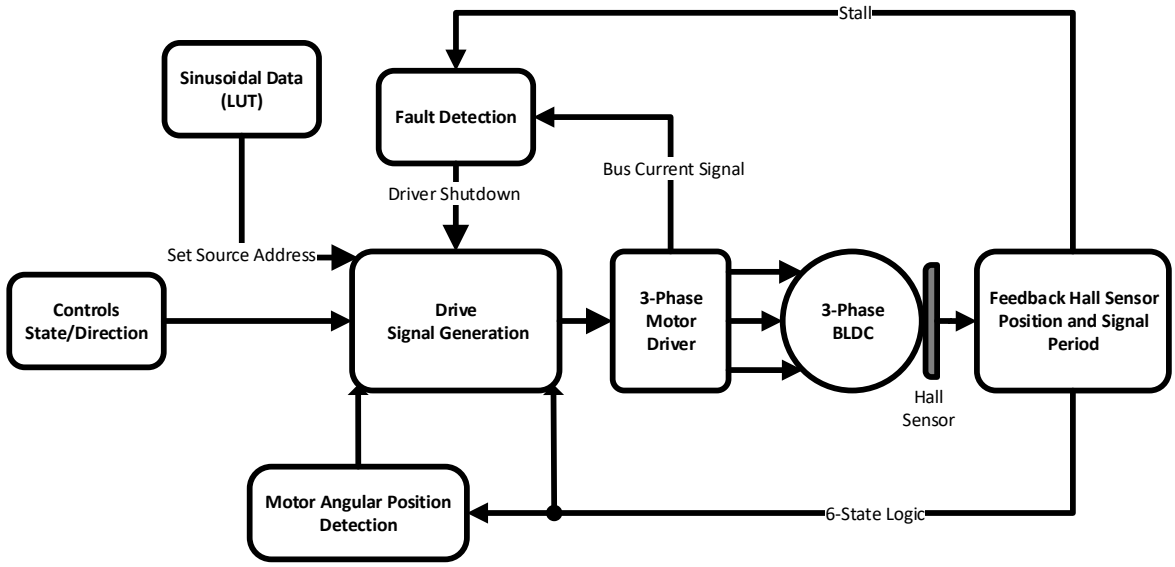


Figure 1-1 shows the functional block diagram of the 3-phase BLDC motor operation. Input control signals are available for stop/go commands and direction control. These input controls determine when the motor will rotate and the commutation sequence the drive signal generator will follow. The sinusoidal data provides PWM duty cycle values stored on Programmable Flash Memory (PFM), which will be loaded to the drive signal generator, depending on the rotor position. The motor used in this application is DMB0224C10002, which has a built-in Hall effect sensor as feedback for the motor position.

2. Sinusiodal Drive - Sensored BLDC Motor Principle

This application note focuses on the implementation of sinusoidal current drive operation on a 3-phase BLDC, using an 8-bit PIC® MCU. Hall effect sensors are used as feedback to identify the rotor position, which is required for the BLDC driving algorithm. For a more detailed explanation of BLDC motor control fundamentals, refer to AN885: BLDC Motor Fundamentals.

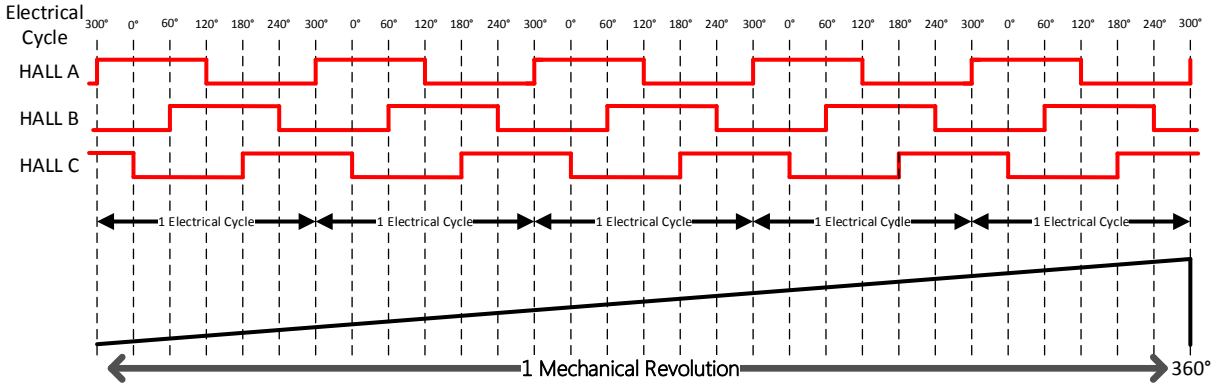
2.1 BLDC Sensored Operation

Identifying the rotor position is an important aspect in BLDC motor control. The rotor position is used to determine the proper electronic commutation. The most common method of identifying rotor position in a BLDC motor is by using Hall effect sensors. Hall effect sensors are sensing switches that produce logic level, based on the detected magnetic field. As the motor rotates, the Hall effect sensors identify the position information of the magnet poles (positive or negative polarity) installed in the motor, sending it to the controller. In this application, three Hall effect sensors are pre-installed inside the motor. These Hall sensors are distributed equally around the stator (120° apart) in a way that its output can generate six different combinations in one electrical cycle, which changes for every 60° of movement. These combinations can be translated into a number from one to six, which are represented by three binary digits. These sensors are positioned in a way that the magnets' polarity will change even before the rotor is in the position for the next commutation, preventing the rotor from being stuck.

The number of electrical cycles in one complete revolution is based on the number of pole pairs the motor has, as shown in Figure 2-1. Since a 5-pole pair motor is used in this application, a total of five electrical cycles is needed to complete one mechanical revolution (one rotation). Also, it is known that the Hall sensor will change states every electrical cycle and by measuring the time between each state change, the angular velocity or motor speed can be obtained.

In this application, the microcontroller counts the number of system clock ticks that is accumulated during a Hall period or an electrical cycle, using a series of timers. The number of clock ticks in a Hall period represents the speed of the motor, which means that the speed of the motor in clock ticks is five times the number of system clock ticks on a Hall period. This information can be used to subdivide the Hall period into smaller intervals for the rate of changing the applied voltage to the driver, which is vital to the sinusoidal current drive or simply measure its current speed.

Figure 2-1. Relationship between Hall Sensor Signal, Electrical Cycle and Mechanical Revolution



2.2 Sinusoidal Current Drive

The basic premise of the sinusoidal drive is to provide each motor winding with currents that vary sinusoidally, based on the rotor position. These currents are phase-shifted by 120° from each other, relative to its corresponding Hall sensor. In BLDC motor control, the drive signals that are used require variable voltages that change with respect to the speed and position of the motor. This variable voltage is applied using the PWM technique. By providing sinusoid-based signals through the PWM modules to the MOSFET driver, the current is generated on each motor windings. Due to the gradual changing of the applied voltage, the sinusoidal drive's torque ripple is somehow lower in comparison with the trapezoidal drive.

In this application, the sinusoidal current drive is applied using data extracted from the Space Vector Pulse-Width Modulation (SVPWM) technique. This technique is widely used for digital implementation because of its higher efficiency and more sophisticated algorithm, over regularly applying an individual sinusoidal signal to the motor windings.

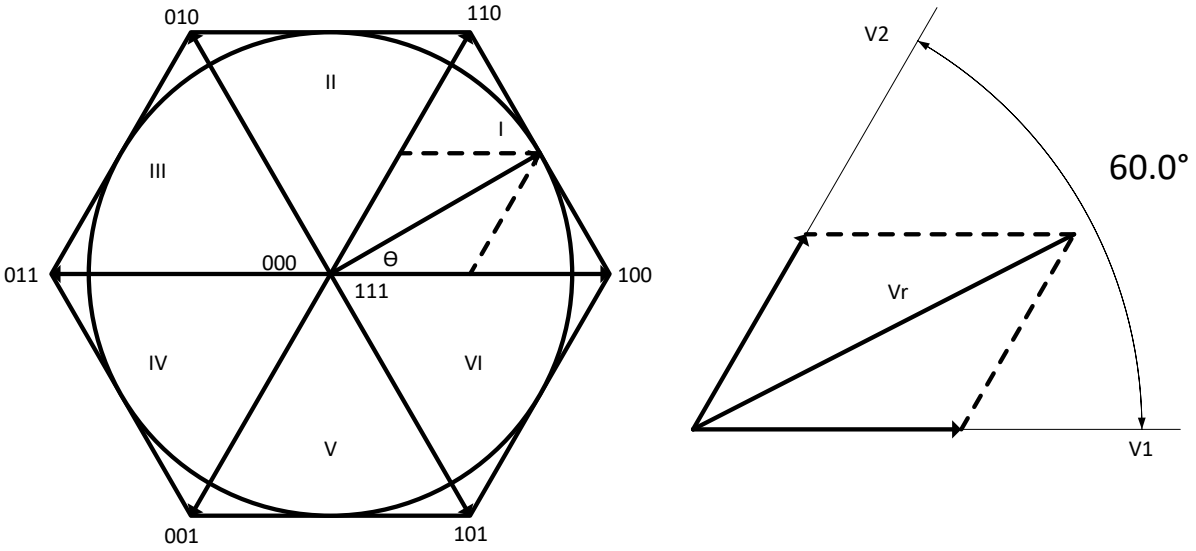
The SVPWM method is a vector-based scheme used in three-phase systems, such as motor control applications. Rather than producing pure sinusoid waveform from the drive signals for each motor terminals with respect to ground, SVPWM generates three sinusoidal line-to-line voltage (differential voltage) between two terminals. A space vector representation is created, where the spaces between unit vectors are separated by 60.0° correspond to every state in an electrical cycle, as shown in Figure 2-2. Each desired voltage can be simulated by adding the components of two adjacent active vectors and null vectors represented by the `000` and `111` logic states (located at the origin). The resulting desired voltage is represented by the manipulated duty cycle of PWM peripheral.

Table 2-1 provides the equation for the PWM switching time for each sector. Once the approximate angular position is identified, the resultant vector magnitude is calculated, with respect to the adjacent voltage space vectors and null vector T0, T1 and T2 representing conduction time within a period. The values for T1 and T2 are taken from a Look-up Table containing 172 fractional sinusoidal values from 0° to 60°.

Table 2-1. Equations for PWM Switching Time by each Sensor

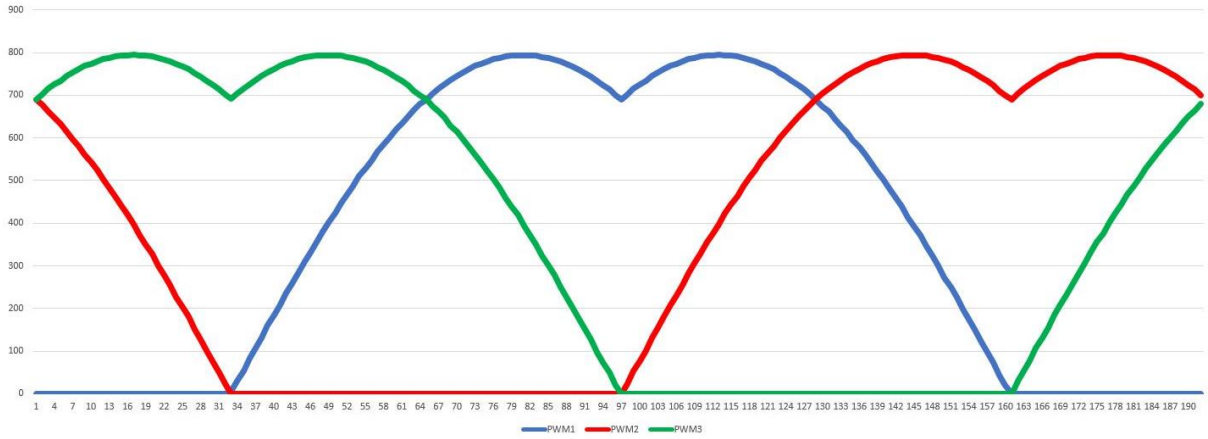
Sector	Switching Time Equation
1	PWM1 = T1 + T2 + T0/2 PWM2 = T2 + T0/2 PWM3 = T0/2
2	PWM1 = T1 + T0/2 PWM2 = T1 + T2 + T0/2 PWM3 = T0/2
3	PWM1 = T0/2 PWM2 = T1 + T2 + T0/2 PWM3 = T2 + T0/2
4	PWM1 = T0/2 P PWM2 = T1 + T0/2 PWM3 = T1 + T2 + T0/2
5	PWM1 = T2 + T0/2 PWM2 = T0/2 PWM3 = T1 + T2 + T0/2
6	PWM1 = T1 + T2 + T0/2 PWM2 = T0/2 PWM3 = T1 + T0/2

Figure 2-2. SVPWM Representation in an Orthogonal Plane



The approximate PWM output from SVPWM, with respect to its angular position, is plotted in Figure 2-3. The waveform produced has a shape of a saddle due to the third harmonic injection by SVPWM. Each PWM output is shifted 120° from each other. The angular position is scaled from 360° to 192 for more convenient implementation in this application. To reduce switching loss, each of the three phases do not switch for one-third of the time in a cycle.

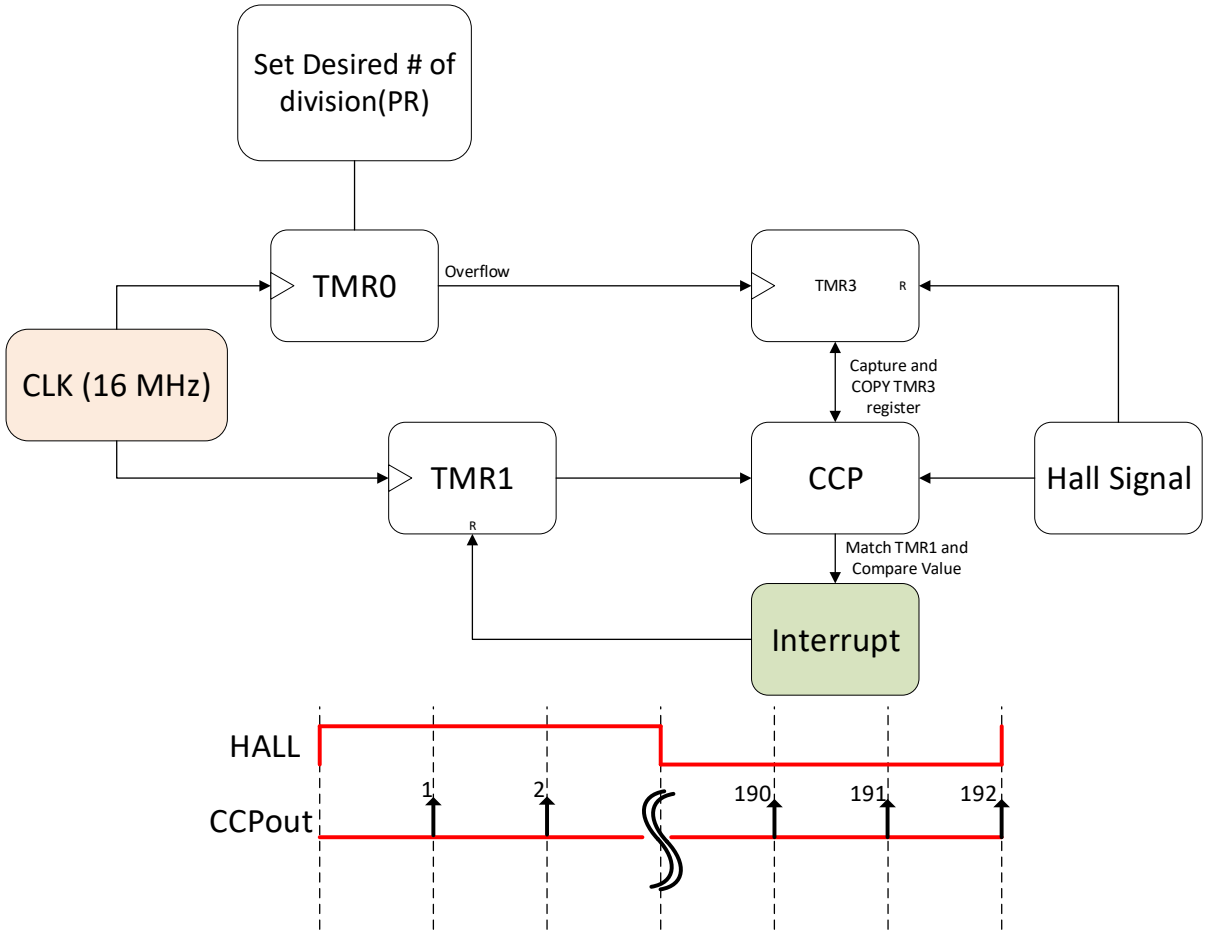
Figure 2-3. SVPWM Voltage Waveform Representation



3. Angular Position Detector

The angular position of the rotor is vital to produce waveforms that are synchronized to the motor during the application of sinusoidal current drive. Hall effect sensors provide limited information since it can only detect the approximate position in an electrical cycle. To apply the sinusoidal current drive to the BLDC motor, a constant change in applied voltage to the MOSFET driver is required, based on the exact rotor position. In this application, a motor angular position function is designed to convert the Hall signal into equally smaller signals that represent a unit angle of rotation. It subdivides the Hall signal into smaller intervals where it acts as a trigger to change the applied voltage at a certain time and position. Several peripherals, such as timers and CCP, are used to create this function. The basic idea of this method is to count the number of system clock ticks it takes to complete a Hall period and divide it to the number of units desired for the next period, assuming that it has the same clock tick count. Since it is basically a hardware division, there will possibly be remainder clocks in the calculations. These remainder clocks will be added to the last degree/phase of the period, especially when using a slower clock source. To reduce such errors, a faster clock source is used. In this design, a 16 MHz clock is used. Figure 3-1 shows the block diagram of the implemented motor angular position scheme.

Figure 3-1. Motor Angular Position Detective Scheme Block Diagram



The period register of the TMR4 is used to set the desired number of divisions per Hall signal period. Since the number of arrays declared is six with 32 elements each, the number of divisions set in T4PR is 192.

TMR3 does not have a direct input connection from the TMR4 Overflow bit. By using CLC as a buffer, the TMR4 output is rerouted to the TMR3 input. It will continuously increment until the reference edge of the Hall signal is detected. Once the reference edge is detected, the value in the TMR3 register is copied to the CCP compare value, which can also be calculated using Equation 3-1. For example, a 200 Hz Hall signal takes 80,000 clock ticks. Since

each TMR3 increment takes 192 clock ticks, the TMR3 counter would have registered a value of 416, when the Hall sensor edge is detected. The value is copied to the CCP Compare register, which is continuously compared to the TMR1 register. TMR1 shares the same clock source as TMR4, which is a 16 MHz clock. If matched, a CCP interrupt will occur to reload the TMR1 register and signals the DMA to increment the source address for the next PWM value. For the MCC configuration, the TMR4 register requires a time-based period and not a register value. It can be calculated using Equation 3-2. T4TMR is an 8-bit register that can store 255 values. Each clock cycle increments the T4TMR until it matches the T4PR. The clock source used is $F_{OSC}/4$ of 64 MHz clock.

Equation 3-1. Compare Value Calculation

$$Compare\ Value = \frac{CLK\ source}{Hall\ Frequency \times\ Desired\ number\ of\ Interrupts}$$

Equation 3-2. TMR4 Period Calculation Based on Desired Number of Interrupts

$$TMR4\ period = \frac{desired\ CNTR\ end\ value\ +\ 1}{CLK\ source}$$

$$TMR4\ period = \frac{192\ +\ 1}{16\ MHz} = 12.062\,\mu s$$

4. Drive Signal Generator

The sinusoidal data is divided into six arrays for the DMA controller to easily point its address as the source, as shown in Figure 4-1. The three windings require the same signal, phase-shifted by 120°, therefore a single set of sinusoidal data can be used.

Figure 4-1. Sinusoidal Data Look-Up Table

```
//PWM sinusoidal lookup table for six states
const uint16_t PWM_1[32] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0};
const uint16_t PWM_2[32] = {5, 30, 54, 83, 107, 131, 160, 184, 208, 236,
    259, 283, 310, 332, 355, 380, 401, 423, 448, 467,
    487, 510, 529, 547, 568, 584, 601, 620, 635, 650,
    666, 680};

const uint16_t PWM_3[32] = {690, 704, 715, 725, 736, 745, 753, 762, 769, 774,
    780, 785, 788, 791, 794, 794, 794, 794, 793, 790,
    787, 784, 779, 773, 767, 759, 751, 744, 733, 723,
    713, 699};

const uint16_t PWM_4[32] = {690, 701, 715, 725, 734, 745, 753, 761, 769, 774,
    779, 785, 788, 791, 794, 794, 795, 794, 793, 791,
    787, 784, 779, 773, 767, 761, 751, 744, 734, 723,
    713, 701};

const uint16_t PWM_5[32] = {687, 674, 661, 644, 629, 614, 594, 578, 561, 540,
    521, 502, 479, 459, 439, 414, 393, 372, 345, 323,
    301, 273, 250, 227, 198, 175, 151, 122, 98, 73,
    44, 20};

const uint16_t PWM_6[32] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0,};
```

Each array consists of 32 16-bit elements. These arrays are stored in the Program Flash Memory (PFM) of the MCU. Using the DMA controller, the address of the first element is set as the DMA source address. The data is transferred from the PFM to the PWM Duty Cycle register, which is set as the DMA destination address. The DMA controller requires the size of the destination address to be a multiple of the source address size and vice versa. The DMAxDSZ register determines the destination size, while the DMAxSSZ register determines the size of the source. The PWM Duty Cycle register is a 16-bit register, which makes the value of the DMAxDSZ register equal to 2 bytes. The value of DMAxSSZ depends on the array size stored in the PFM. It can be calculated using Equation 4-1.

Equation 4-1. DMAxSSZ Register Value

$$DMAxSSZ = element\ size\ (byte) \times\ \#\ of\ elements$$

There are two events that access or modify the settings of the DMA controller; one is during Hall state transition, and the other is during motor angular position interrupt. When a Hall state change is detected, the DMA controller source address register is loaded by the firmware, depending on both the sector position and direction. Changing the

addresses during operation requires enabling and disabling the DMA channel on the run. A lock and unlock sequence is required to do this. On the other hand, the motor angular position interrupt triggers the DMA controller to increment the DMA source address, pointing to the next element on the array until all elements have been loaded to the destination or another Hall state change is detected.

Each DMA channel's destination address is pointed to the PWM slice register where the variable voltage is set. The switching frequency for motor control applications depends on the motor's inductance. If the motors current can change at a fast rate, the switching frequency should be high enough to avoid core saturation. In this application, the switching frequency was set to 40 kHz, catering the need to provide high frequency and high resolution for motor operation. Equation 4-2 shows the calculation of the PWM period, in reference to the frequency selected. The sinusoidal data that is used in this application are duty cycle values, based on the PWM resolution and PWM period, which can be calculated using Equation 4-3. Center-Aligned mode is selected to produce two pulses line-to-line in each period. By doing this, the ripple current is reduced while not increasing the switching loss in power devices. The PWM registers are double-buffered. It requires an external source or LD bit to be triggered. In this application, DMA is used as an external load event. Every time the DMA channel successfully transfers a message, a request to load is made. The buffer updates every other request.

Equation 4-2. PWM Period Register Value in Center-Aligned Mode PWM

$$PWMxPR = \frac{CLK\ Source}{Switching\ Frequency\ \times\ Prescaler\ \times\ 2} = \frac{64\ MHz}{40\ kHz\ \times 1\ \times 2}\ -1\ = 799 = 0x31F$$

Equation 4-3. PWM Duty Cycles Register Value

$$PWMxS1P1 = PWMxPR \times \frac{desired\ duty\ cycle\ \%}{100}$$

To drive the three-phase motor with sinusoidal currents, the CWG peripheral is used. The MOSFET driver of the three-phase BLDC motor consists of three half-bridges, one for each motor terminal. Each half-bridge consists of two switches, which require complementary PWM signals. Because of this, the CWG modules are configured in Half Bridge mode, where each generates two output signals that resemble the true and inverted versions of the input PWM. The CWGxA input is configured for the high-side switches, while the CWGxB input is configured for the low-side switches of the driver. Figure 4-2 shows the CWG output representation for each half bridge.

Figure 4-2. CWG Output

Since power MOSFETs take a finite amount of time to switch on or off, there may be a time where both switches of a half bridge turn on. It creates a low resistance path from the source to the ground that could cause a short circuit. This event is called shoot-through. In CWG, a dead-band delay is inserted to prevent shoot-through current during an output level transition. Dead-band delay is a time delay inserted between the switch transition to prevent conduction of high and low-side switches at the same time. Dead-band delays can be inserted in the rising or falling edge of the input source. Dead-band counters are set on CWGxDBR and CWGxDBF registers.

5. Fault Detection Control

Fault detection control is implemented within this system to stop motor operation when an event that might cause motor drive failure is detected. To avoid a false trigger, each detection unit should be enabled a fraction of a second after the motor start-up, using TMR0.

5.1 Overcurrent Protection

If the circuit draws more than the maximum allowable current caused by commutation failures, excessive load or internal faults, etc., the circuit might experience overheating or system failure. To prevent such conditions, an overcurrent detection is used. Overcurrent detection is implemented by monitoring the bus current of the 3-phase MOSFET driver. Resistor R_{SHUNT} is tapped at the low side of the motor driver. The voltage across R_{SHUNT} represents the current that flows through the driver, since the voltage is directly proportional to the current across the resistor. The bus current can be computed using Equation 5-1.

Equation 5-1. Bus Current Calculation

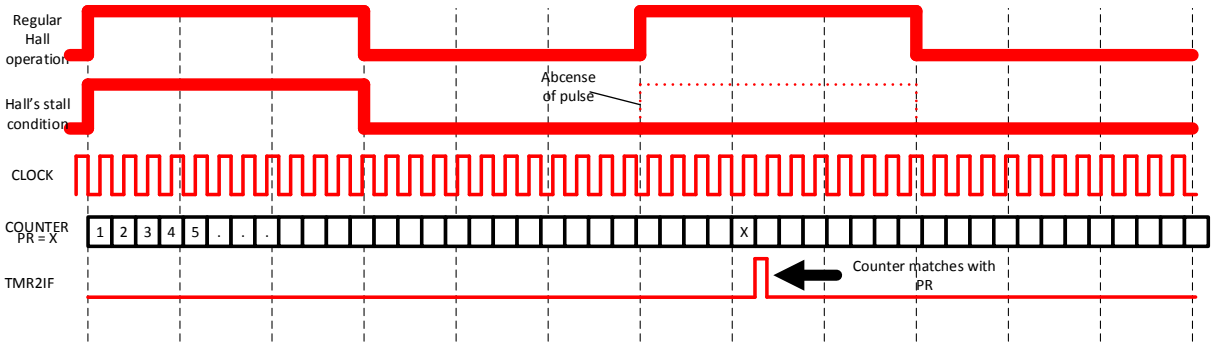
$$I = \frac{V_{DD} - V_{BIAS}}{R_{SHUNT} \times A_V}$$

The shunt resistor's voltage is continuously compared to the reference voltage provided by the MCU. During motor runtime, the bus current is sampled by the comparator (CMP) input. Every 50 ms, the comparator output voltage is monitored, to verify whether it exceeds the reference voltage set in the DAC. If the comparator output is high, the CWG peripheral will be disabled and stop the motor. During motor start-up, a high current is drawn by the motor before it stabilizes. This might cause a false trigger and stop the motor. To prevent this situation, a 500 ms delay after start-up is inserted before enabling the fault detection unit.

5.2 Stall Detection

The motor Hall sensor provides a stable periodic signal in the form of pulses, when the motor is running. The absence of these pulses over a long period indicates the motor has stopped or is in a Stall condition. This condition normally happens when the motor load torque is greater than the motor shaft torque. During a Stall condition, the motor draws current that exceeds the rated value, which also causes motor overheating and failure.

Figure 5-1. HLT Operation



To detect such a condition, a stall detection feature is also implemented. By using the Hardware Limit Timer (HLT) of the TMR2 peripheral, the absence of a Hall signal within a period of time, can be detected. It is set to operate in rollover Pulse mode. In this mode, an input signal is allowed to clear the TMR2 register every time a falling edge transition is detected. When the pulse is absent or has a long periodic time, TMR2 will continuously increment until it reaches its maximum value, setting its Overflow bit. HLT will trigger an interrupt that disables all CWG peripherals, stopping the motor. In this application, the Hall B signal, which is connected to the RC6 input, is used as the reference signal. The signal provided by the Hall sensor has an approximate period of 5 ms in a normal setup. If the timer counter matches the PR value, the interrupt flag will be set as shown in Figure 5-1. The PR value that will be set on TMR2 should be larger than the period of the Hall sensor signal, in order to identify the stall condition of the motor. To calculate TMR2 PR register values, use Equation 5-2.

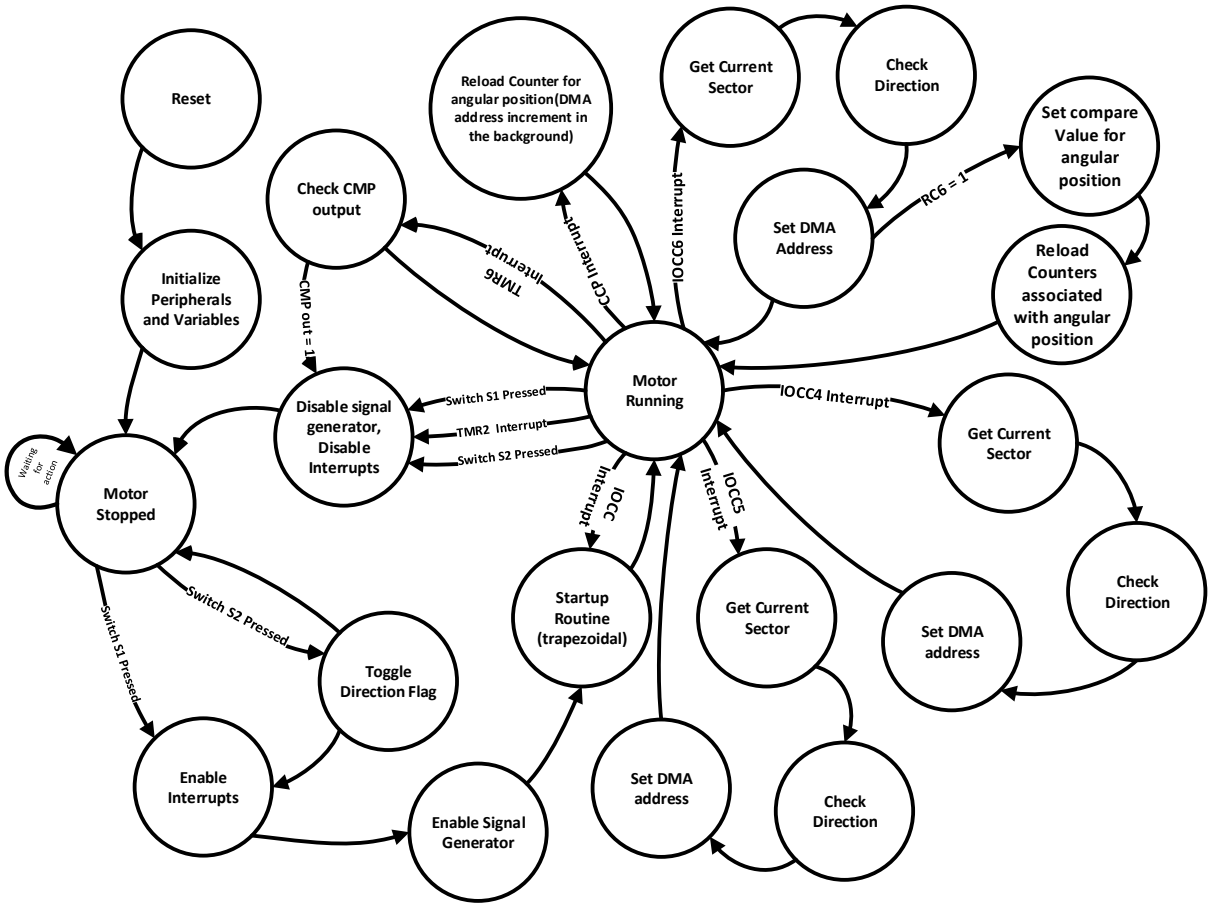
Equation 5-2. PR2 Calculation

$$PR2 = \frac{TMR2\ Clock\ Source}{Hall\ Signal\ Period\ \times\ Prescaler\ \times\ Postscaler}$$

6. Firmware Implementation

The state diagram in Figure 6-1 illustrates the motor control system and its interaction with each interrupt service available in the system. The firmware basically revolves on the state of the motor, whether it is in the Running condition or in Stop condition. Flags are used to identify the state of motor operation. From Reset, all peripherals and variables are initialized while keeping interrupts disabled. The motor is initially set in a Stop state and in a CCW direction. For the complete source code, refer to Appendix C: Source Code Listing.

Figure 6-1. Motor Control State Diagram



The device used in this application is PIC18F47Q43. It contains 8192 bytes of data memory, 131,072 bytes of program memory and 2048 bytes EEPROM. The application uses 94 bytes (1%) of its data memory and 7,190 bytes (5%) of program memory. EEPROM is not used in this application.

6.1 Button Controls

Two buttons are used as input to trigger the start/stop operation and CCW/CW direction. Both input buttons are continuously monitored after initialization as shown in Example 6-1. When switch S1 is pressed, it checks the motor state if it is in running or stall state. If it is in a Stall state, the `RunMotor()` subroutine will be called enabling CWG, Interrupt routines and CCP. If the motor is in running state and switch S1 is pressed, `StopMotor()` subroutine will be called disabling CWG, Interrupt routines and CCP, toggling the motor condition flag.

Example 6-1. Program Main Loop

```
while (1)  {

            // wait until an action is detected
            CheckTriggerButton();
            CheckDirectionButton();
}
```

Switch S2 toggles the direction flag. When S2 is pressed during a motor Stop condition, it will simply change the direction flag from CCW to CW, or vice versa. By toggling the direction flag, the commutation order based on the sinusoidal data array during Hall transition is changed. If S2 button is pressed in running condition, the motor is forced to stop before changing the commutation sequence. Once the direction flag is toggled and the position is identified, the motor will start running again.

6.2 Motor Start-up

The motor's initial speed from a Stop condition is zero. This makes the application of sinusoidal current drive difficult since there is no reference speed available. In this application, a low-speed trapezoidal drive is used as the starting commutation until the motor stabilizes for the sinusoidal drive. The trapezoidal drive is executed by configuring the CWG peripherals to Steering mode. Each time a Hall transition is detected, the PWM signal is steered to the output pin based on the 6-step drive table stored in program Flash, as shown in Figure 6-2. Each drive table array contains six elements based on the six states of Hall sensor logic. As the motor starts to turn, the PWM duty cycle is increased gradually until it establishes speed and completes a mechanical revolution.

Figure 6-2. Trapeziodal Look-Up Table

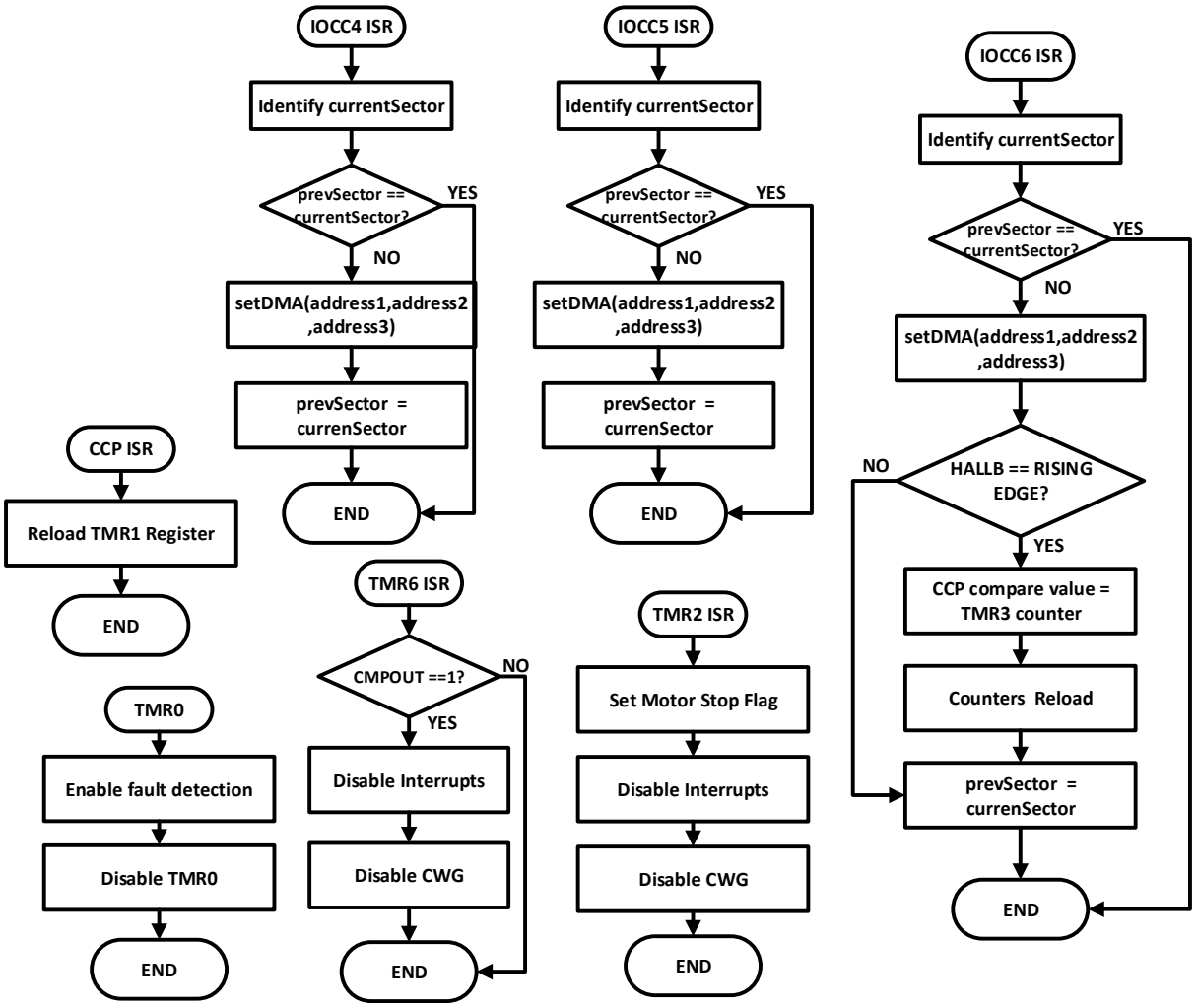
```
const uint8_t fwdDriveTable_CWG1 [7] = {0x00, 0x02, 0x00, 0x02, 0x10, 0x00, 0x10};
const uint8_t fwdDriveTable_CWG2 [7] = {0x00, 0x00, 0x10, 0x10, 0x02, 0x02, 0x00};
const uint8_t fwdDriveTable_CWG3 [7] = {0x00, 0x10, 0x02, 0x00, 0x00, 0x10, 0x02};

const uint8_t revDriveTable_CWG1 [7] = {0x00, 0x10, 0x00, 0x10, 0x02, 0x00, 0x02};
const uint8_t revDriveTable_CWG2 [7] = {0x00, 0x00, 0x02, 0x02, 0x10, 0x10, 0x00};
const uint8_t revDriveTable_CWG3 [7] = {0x00, 0x02, 0x10, 0x00, 0x00, 0x02, 0x10};
```

By the time it completes a mechanical revolution, the CWG peripherals are changed to Half-Bridge mode for the preparation of transition to sinusoidal drive. The timers used on angular transition are all reloaded and the functions on IOC ISR are changed with respect to sinusoidal function.

6.3 Interrupt Sources

After initialization, only the monitoring of the input buttons is performed in the main loop. Other processes and functions are either performed by the CIPs or triggered by interrupt resources. When a certain condition is met, an interrupt flag is set and its corresponding Interrupt Service Routine is performed. Figure 6-3 shows how these routines are executed.

Figure 6-3. ISR Flowchart



All peripherals used in the firmware are configured using MPLAB® Code Configurator (MCC). For the step-by-step procedure of how all peripherals are configured, refer to Appendix B: MPLAB Code Configurator (MCC) Peripheral Configuration.

6.3.1 Hall ISR

Each Hall sensor is connected to a digital pin of the MCU in which Interrupt-on-Change (IOC) is enabled. If a Hall transition, either a falling or rising edge is detected, Hall ISR is called. The Hall state is represented by a 3-bit Logic state which provides the current sector location. During start-up, the trapezoidal function is set as Hall ISR in which the six-step commutation is executed. When a mechanical revolution is completed, the Hall ISR is replaced with the sinusoidal function. `(uint8_t) ((PORTC >> 4) & 0x07)` is used since all Hall pins are located in the PORTC register. The DMA source address that will be set is based on the `currentSector0`. Using the `SetDMA()` subroutine, the source addresses of each DMA channel is pointed to a different set of sinusoidal data, depending on the detected location. Each PWM module is configured with an individual transfer, utilizing three DMA channels. The DMASELECT register should be set before configuring an individual channel. Figure 6-4 shows the `CheckSector()` subroutine that is used in Hall ISR to configure the DMA channels.

Figure 6-4. Check Sector Subroutine

```
void CheckSector(void) {
    switch (currentSector) {
        case SECTOR1:
            if (FLAGS.direction == CCW) {
                SetDMA(&PWM_1, &PWM_5, &PWM_3);
            } else if (FLAGS.direction == CW) {
                SetDMA(&PWM_3, &PWM_5, &PWM_1);
            }
            break;
        case SECTOR2:
            if (FLAGS.direction == CCW) {
                SetDMA(&PWM_2, &PWM_6, &PWM_4);
            }
            else if (FLAGS.direction == CW) {
                SetDMA(&PWM_2, &PWM_4, &PWM_6);
            }
            break;
        case SECTOR3:
            if (FLAGS.direction == CCW) {
                SetDMA(&PWM_3, &PWM_1, &PWM_5);
            } else if (FLAGS.direction == CW) {
                SetDMA(&PWM_1, &PWM_3, &PWM_5);
            }
            break;
        case SECTOR4:
            if (FLAGS.direction == CCW) {
                SetDMA(&PWM_4, &PWM_2, &PWM_6);
            } else if (FLAGS.direction == CW) {
                SetDMA(&PWM_6, &PWM_2, &PWM_4);
            }
            break;
        case SECTOR5:
            if (FLAGS.direction == CCW) {
                SetDMA(&PWM_5, &PWM_3, &PWM_1);
            } else if (FLAGS.direction == CW) {
                SetDMA(&PWM_5, &PWM_1, &PWM_3);
            }
            break;
        case SECTOR6:
            if (FLAGS.direction == CCW) {
                SetDMA(&PWM_6, &PWM_4, &PWM_2);
            } else if (FLAGS.direction == CW) {
                SetDMA(&PWM_4, &PWM_6, &PWM_2);
            }
            break;
    }
}
```

Hall B is also used to identify the reference speed for the motor angular position. Each time a Hall B positive edge transition is detected, the value of the TMR3 period is copied to the CCP compare value and reloads all timers used in motor angular position.

6.3.2 CCP ISR

Every time the Timer1 register matches the compare value register, the CCP ISR flag is set. It triggers the DMA controller to increment the source address for the next array element to be loaded to the PWM duty cycle register. While the DMA performs this task in the background, CCP ISR reloads the Timer1 register.

6.3.3 HLT TMR 2 ISR

If a missing Hall pulse is detected in a period of time and the PR2 matched, the TMR2 flag is set. The motor operation is changed into a Stall state, triggering the motor to stop by calling the `stopMotor()` function and disabling the interrupts.

6.3.4 TMR6 ISR

During motor operation, the current is continuously monitored by using the Comparator (CMP) peripheral. If the V- is greater than the V+, which is set from the DAC register, the output of the comparator is set to digital high. For every 50 ms, TMR6 ISR checks the comparator output. If the CMP output is set, `stopMotor()` function is called, disabling all interrupts and CWG peripherals, stopping the motor operation.

6.3.5 TMR0 ISR

During the motor start-up, the windings behave as a closed circuit, causing the system to draw high current. Referred to as inrush current, this phenomenon might cause a false trigger on the overcurrent detector. By using TMR0, a fraction of a second is inserted before enabling the Fault detection function. When the Timer0 flag is set, the Comparator, TMR2 peripherals and its corresponding interrupts are enabled. The TMR0 is also disabled, making it a one-time execution during motor running operation and will only be in service during another start-up.

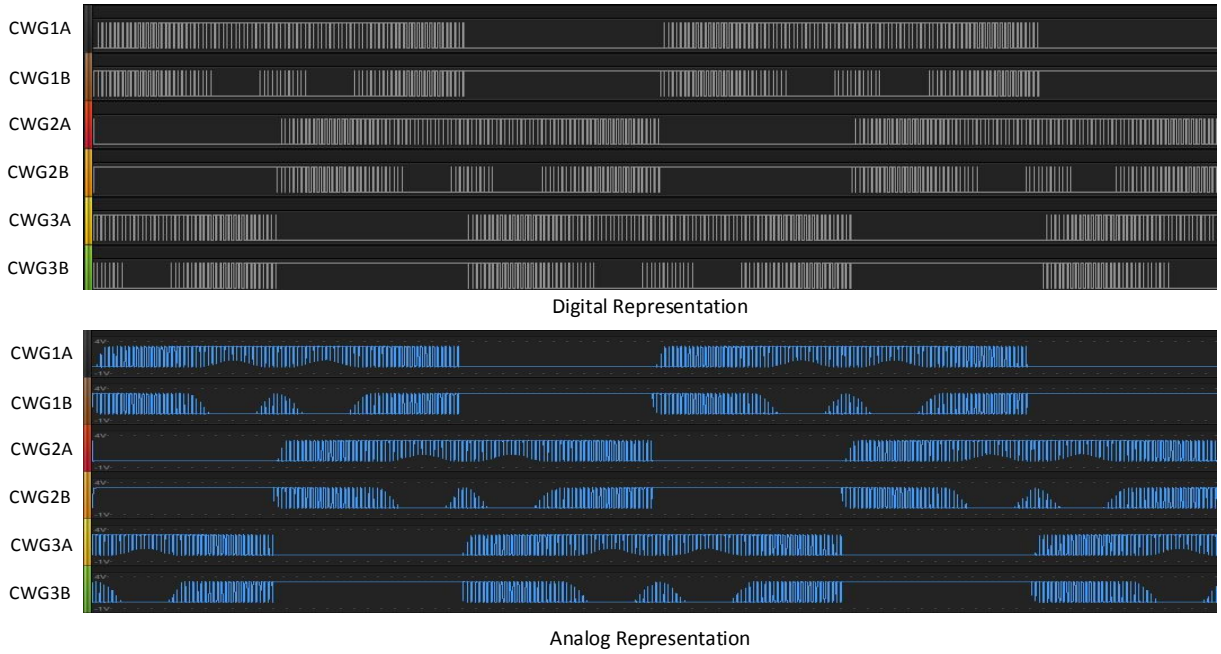
7. Sinusoidal Drive 3-Phase BLDC Performance

To present the viability of this application, both the driving voltage signal and the motor current and voltage are captured. Through observation, the output voltage and current of the sinusoidal drive are compared with the trapezoidal drive.

7.1 CWG Drive Signals for 3-Phase BLDC

Figure 7-1 shows the drive signals provided to the power stage by the CWG module while in running operation.

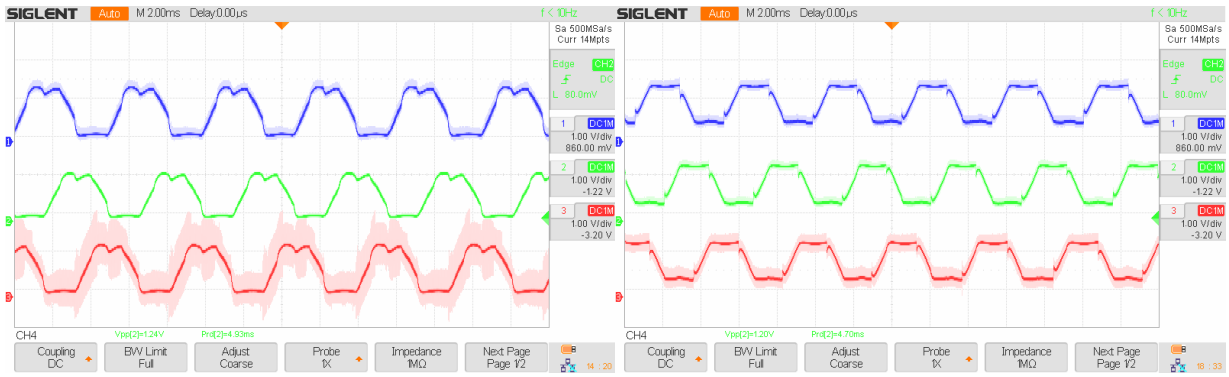
Figure 7-1. CWG Drive Signals



7.2 Phase Voltage and Line Current

Figure 7-2 shows the phase voltages of the windings by the 3-phase BLDC in both sinusoidal drive and trapezoidal drive. Three oscilloscope probes are connected to the voltage dividers of each motor winding. The difference between the driving phase voltages can be observed. The sinusoidal drive produces a saddle shape waveform where the switching voltage is smoothly varied. The trapezoidal drive, on the other hand, generates a trapezoidal waveform.

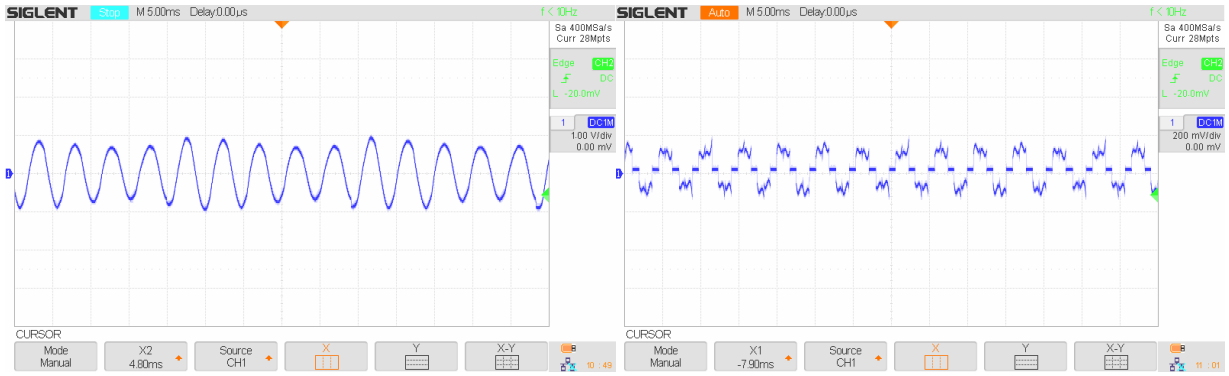
Figure 7-2. Phase Voltages



SINUSOIDAL DRIVE TRAPEZOIDAL DRIVE

Using a current probe, the phase current of the trapezoidal and sinusoidal current drive is obtained and shown in Figure 7-3. Both drive runs at 2400 RPM. Ripples are observed in trapezoidal drive. The sinusoidal drive provides a smoother line current compared to the trapezoidal drive. Ripples are more apparent in the trapezoidal drive. The sinusoidal drive current waveform follows a sinusoidal like pattern in which ripples and audible noise are reduced.

Figure 7-3. Phase Current



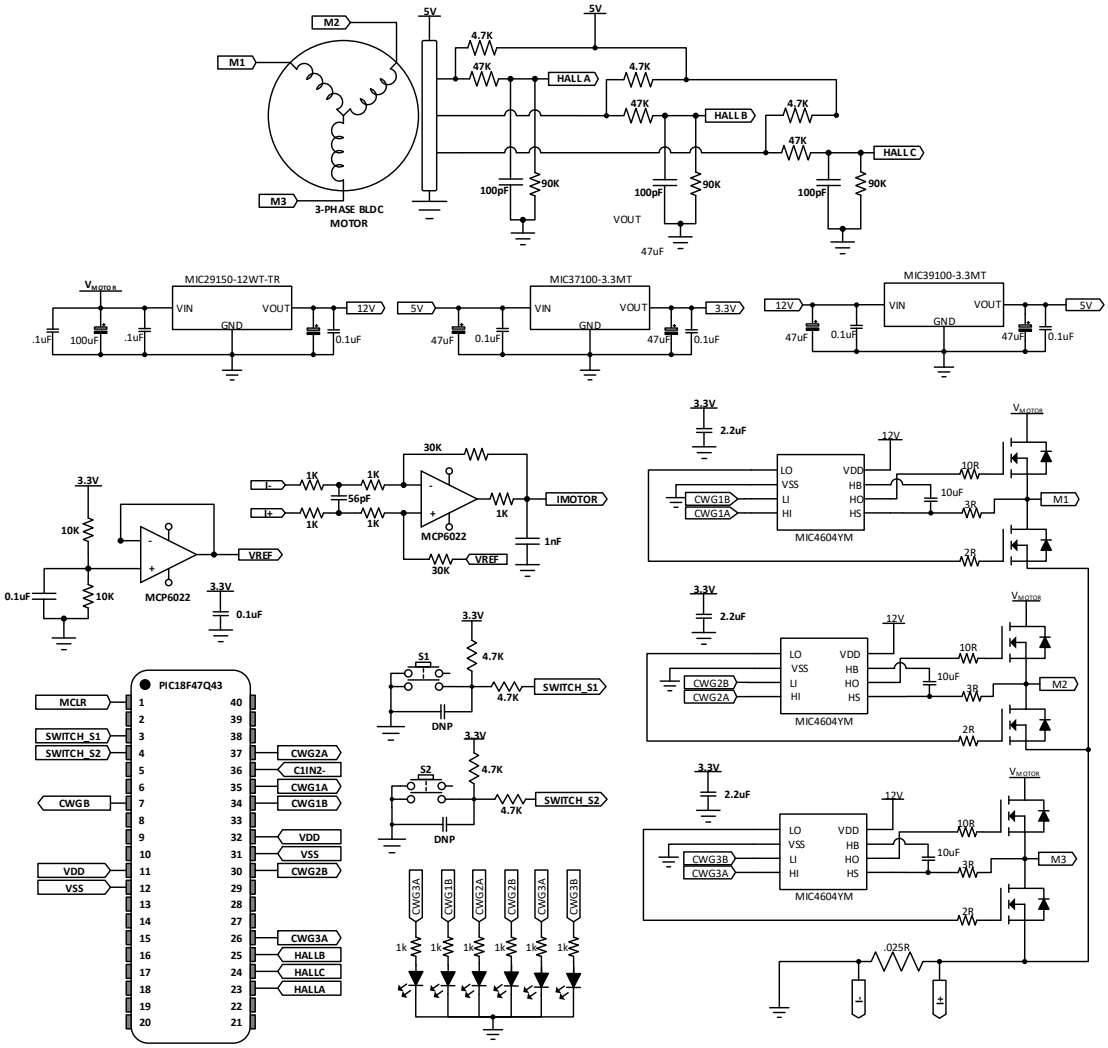
SINUSOIDAL DRIVE TRAPEZOIDAL DRIVE

8. Conclusion

The method of using DMA to directly transfer data from the Look-Up Table (LUT) to PWM can lessen real-time calculation in sinusoidal drive, BLDC motor control. Though lacking with speed control since the data from the LUT cannot be modified, this method is perfect for applications that require non-speed variables and low noise applications. By reducing the functions and instructions using the CIPs DMA controller, other tasks can be performed. By only using a low-cost 8-bit microcontroller, PIC18-Q43, sinusoidal current drive BLDC motor operation is executed successfully. Fault detection is also implemented to automatically stop the operation, when events that might cause the operation to fail are detected.

9. Appendix A Schematics

Figure 9-1. Schematic Diagram A



10. Appendix B: MPLAB® Code Configuration (MCC) Peripheral Configuration

In this section, the initialization and configuration of the peripherals utilized in this application note using MPLAB® Code Configurator (MCC) are shown. MCC is a plug-in tool of MPLAB® X IDE, which provides a graphical environment where peripheral configuration can be executed. MCC generates drivers in C code, which initializes the peripherals and provides functions that can be called on your firmware. Refer to the MPLAB® Code Configurator V3.xx User's Guide (DS40001829) for more information on how to install and set up the MCC in MPLAB X IDE.

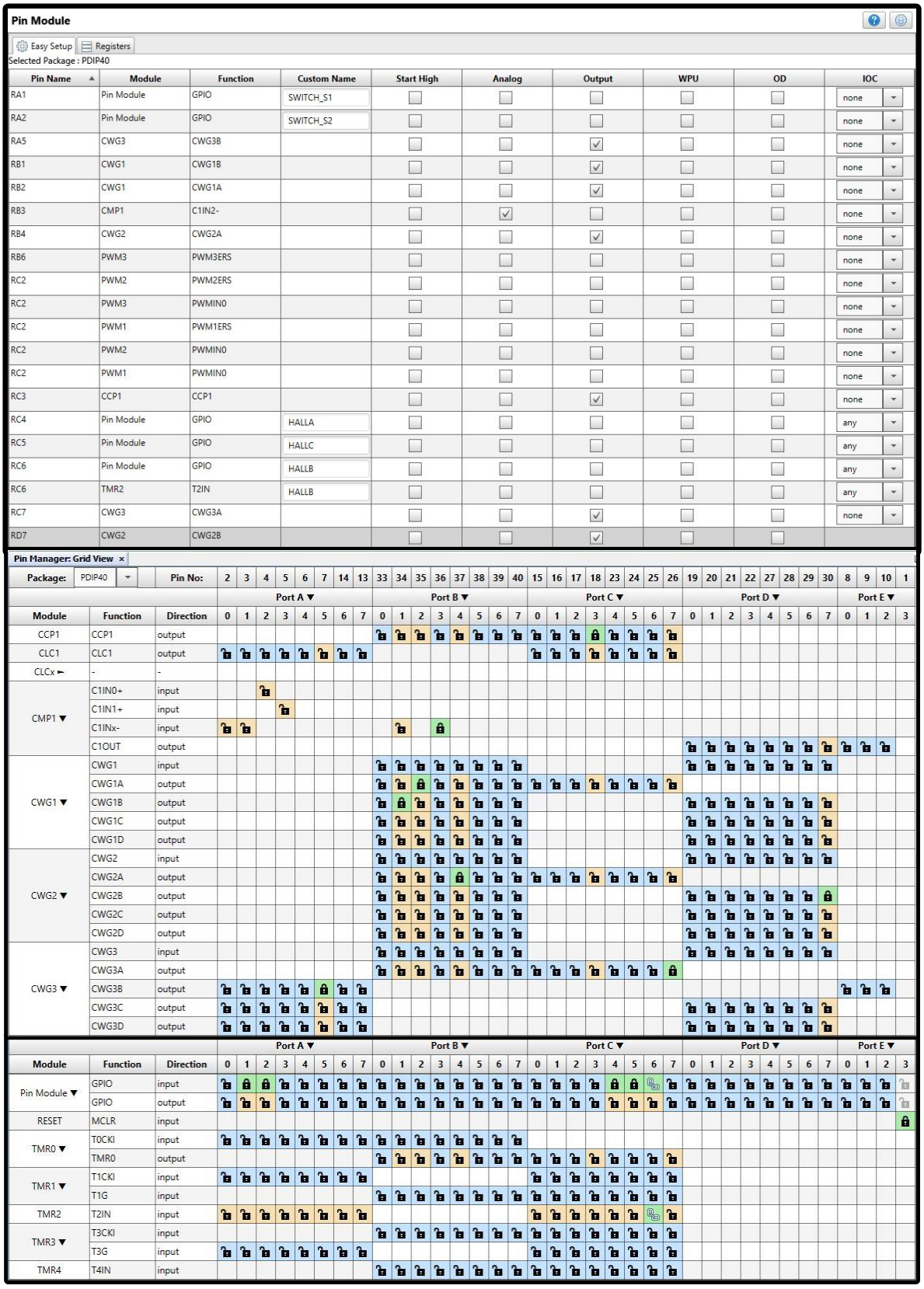
The following steps provide the MCC settings of each PIC18-Q43 peripheral used in this application note.

1. Select HFINTOSC as the clock source with a 64 MHz frequency in the system module.
2. For the Motor Angular Position, TMR4, CLC1, TMR3 CCP1 and TMR1 are configured. The clock source of TMR4 is set to $F_{OSC}/4$. To provide 192 interrupts in a Hall period, set the TMR4 period to 12.0625 µs according to Equation 3-2.
3. Using the CLC1 module, the TMR4 input is redirected to TMR3 input. In the AND-OR mode of CLC1, TMR4 output is selected as input to two OR gates. This configuration acts as a buffer logic that provides input to TMR3 clock.
4. The output of CLC1 is used as an input of TMR3. Every time the TMR4 output is set, TMR3 is incremented by one. It is done to count the number of TMR4 period in a Hall period. Each time a Hall period is detected, the TMR3 counter register is copied to the compare register of CCP and then reloaded with the initial value.
5. Set the CCP1 module to Compare mode. Select the TMR1 module as the timer in which the compare register value is always compared. Select pulse clear timer as the Compare operating mode. Enable the CCP interrupt to provide an ISR that will reload TMR1 every matching event.
6. Select $F_{OSC}/4$ as the TMR1 clock source to perform hardware division with TMR4.
7. The drive signal generator requires that the DMA controller, PWM and CWG modules to be configured. For the PWM module, a single slice is used for each bridge. Select HFINTOSC as the clock source for each PWM module without a prescale value. Choose the Center-aligned mode as the operating mode. Set the requested frequency to 40 kHz, to provide a 10-bit resolution for the duty cycle. In the register's tab, set the PWMxLDS to the corresponding DMA channel to automatically reload the duty cycle value to the PWM output every time the DMA successfully transmitts an element.
8. Disable the CWG module, since the motor is initialized in a Stop condition. Set the PWMxOUT1 register as the input source for the operation. Steering mode is selected as the Output mode since the motor starts in the trapezoidal drive, but changes to the Half-Bridge mode during firmware execution, if motor speed is established. The dead-band count for falling and rising events are set to 63-64 counts with the HFINTOSC clock source.
9. The DMA controller channels are initially disabled and will be manually enabled in the firmware. Set the DMODE and SMODE bits in the DMAxCON1 register to "incremented". Select the program Flash memory as the source in the DMAxCON1 register. For the DMA source address, enter the address location of the corresponding PWM slice register. In MPLAB® X IDE, the address of the register can be easily identified by clicking **Window > Target Memory Views > SFR**. An SFR tab will pop up showing a list of registers and their addressess. Set the DMAxDSZ register to 2, since the PWM duty cycle register has a 2-byte size. For the source address size, set 0x40 to the DMAxSSZ register. CCP1 is selected as an interrupt request source in the DMAxSIRQ register.
10. For the Fault detection feature, TMR0 is used to insert a delay before enabling Fault detection. Set the clock Prescaler to 1:32 and the Postscaler to 1:10. Set the clock source to LFINTOSC. Set the requested period to the desired delay sequence, before enabling the Fault detection features, which is 500 milliseconds. TMR0 is initialized as disabled, since the time set will start upon the motor running state.
11. Set the DAC output value to 2.5V. For the positive and negative reference, choose V_{DD} and V_{SS}, respectively. Disable the DAC output in the DACOUT1 register. The positive input of the CMP module uses the DACOUT1 as the reference voltage. Choose the CIN2- as the negative input of the CMP module. Select the inverted output polarity to provide an output high when the negative input is greater than the DACOUT value. Using TMR6, the comparator output is checked every 50 ms. If the CMP output state is high, a trigger will stop the motor. To apply this scheme, enable the TMR6 interrupt and select LFINTOSC as the clock source. Change the Prescaler value of TMR6 to 1:64 and set the timer period to 50 ms.
12. For stall detection, TMR2's HLT mode is used. Choose the T2CKIPPS pin as the external Reset source to use the Hall B signal to detect a Stall condition. Choose "resets at rising TMR2_ers" as a Start/Reset option to

reset the timer every time the Hall signal is detected before the TMR2 flag is set. For the TMR2 period, indicate the time in which, if no Hall signal is detected, it will trigger an interrupt. Enable TMR2 interrupt.

13. Configure the output and input pins for the peripherals as shown in Figure 10-1.

Figure 10-1. Pin Manager and Pin Module Configuration



Pin Module

Easy Setup | Registers

Selected Package : PDIP40

Pin Name	Module	Function	Custom Name	Start High	Analog	Output	WPU	OD	IOC
RA1	Pin Module	GPIO	SWITCH_S1	☐	☐	☐	☐	☐	none
RA2	Pin Module	GPIO	SWITCH_S2	☐	☐	☐	☐	☐	none
RA5	CWG3	CWG3B		☐	☐	☑	☐	☐	none
RB1	CWG1	CWG1B		☐	☐	☑	☐	☐	none
RB2	CWG1	CWG1A		☐	☐	☑	☐	☐	none
RB3	CMP1	C1IN2-		☐	☑	☐	☐	☐	none
RB4	CWG2	CWG2A		☐	☐	☑	☐	☐	none
RB6	PWM3	PWM3ERS		☐	☐	☐	☐	☐	none
RC2	PWM2	PWM2ERS		☐	☐	☐	☐	☐	none
RC2	PWM3	PWMIN0		☐	☐	☐	☐	☐	none
RC2	PWM1	PWM1ERS		☐	☐	☐	☐	☐	none
RC2	PWM2	PWMIN0		☐	☐	☐	☐	☐	none
RC2	PWM1	PWMIN0		☐	☐	☐	☐	☐	none
RC3	CCP1	CCP1		☐	☐	☑	☐	☐	none
RC4	Pin Module	GPIO	HALLA	☐	☐	☐	☐	☐	any
RC5	Pin Module	GPIO	HALLC	☐	☐	☐	☐	☐	any
RC6	Pin Module	GPIO	HALLB	☐	☐	☐	☐	☐	any
RC6	TMR2	T2IN	HALLB	☐	☐	☐	☐	☐	any
RC7	CWG3	CWG3A		☐	☐	☑	☐	☐	none
RD7	CWG2	CWG2B		☐	☐	☑	☐	☐	

Pin Manager: Grid View

Package: PDIP40 — Pin No: 2 3 4 5 6 7 14 13 | 33 34 35 36 37 38 39 40 | 15 16 17 18 23 24 25 26 | 19 20 21 22 27 28 29 30 | 8 9 10 1

Module	Function	Direction	Port A ▼ 0 1 2 3 4 5 6 7	Port B ▼ 0 1 2 3 4 5 6 7	Port C ▼ 0 1 2 3 4 5 6 7	Port D ▼ 0 1 2 3 4 5 6 7	Port E ▼ 0 1 2 3
CCP1	CCP1	output					
CLC1	CLC1	output					
CLCx ▸	-	-					
CMP1 ▼	C1IN0+	input					
	C1IN1+	input					
	C1INx-	input					
	C1OUT	output					
CWG1 ▼	CWG1	input					
	CWG1A	output					
	CWG1B	output					
	CWG1C	output					
	CWG1D	output					
CWG2 ▼	CWG2	input					
	CWG2A	output					
	CWG2B	output					
	CWG2C	output					
	CWG2D	output					
CWG3 ▼	CWG3	input					
	CWG3A	output					
	CWG3B	output					
	CWG3C	output					
	CWG3D	output					

Module	Function	Direction	Port A ▼ 0 1 2 3 4 5 6 7	Port B ▼ 0 1 2 3 4 5 6 7	Port C ▼ 0 1 2 3 4 5 6 7	Port D ▼ 0 1 2 3 4 5 6 7	Port E ▼ 0 1 2 3
Pin Module ▼	GPIO	input					
	GPIO	output					
RESET	MCLR	input					
TMR0 ▼	T0CKI	input					
	TMR0	output					
TMR1 ▼	T1CKI	input					
	T1G	input					
TMR2	T2IN	input					
TMR3 ▼	T3CKI	input					
	T3G	input					
TMR4	T4IN	input					

After setting all peripherals needed, click the “Generate code” in the resource management tab. It will generate peripheral drivers and a main.c, where all peripherals are initialized based on your configuration.

11. Appendix C Source Code Listing

The latest software version can be downloaded from the Microchip website (www.microchip.com). The user will find the source code appended to the electronic version of this application note.

The Microchip Website

Microchip provides online support via our website at http://www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to http://www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: http://www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with

your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PackeTime, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TempTrackr, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, FlashTec, Hyper Speed Control, HyperLight Load, IntelliMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, Vite, WinPath, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BlueSky, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, INICnet, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

ISBN: 978-1-5224-6008-4

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit http://www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: http://www.microchip.com/support Web Address: http://www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4485-5910 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-72400 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820