Freescale Semiconductor
Application Note

AN2876
Rev. 1.0, 07/2005

Programming the Pulse-Width Modulator for Motor Control (PWMMC) on HC08 Microcontrollers

Using MC68HC08MR Series to Demonstrate PWM Techniques

By Jorge Zambada Tinoco
RTAC Americas
Mexico 2005

Introduction

This document is intended to serve as a quick reference for an embedded engineer to get the pulse-width-modulation (PWM) module up and running for any HC08 MCU that includes this module. Basic knowledge about the functional description and configuration options will give the user a better understanding on how the PWM module works. This application note provides an example that demonstrates one use of the PWM module within the HC08 Family of microcontrollers. The examples mentioned are intended to be modified to suit the specific needs for any application.

The complete software files for the example described in this document is available as AN2876SW.zip free-of-charge from the Freescale Semiconductor web site: http://freescale.com

PWM Module

The PWM module included in the MC68HC08MR Series of microcontrollers is a motor-control-oriented pulse-width modulator capable of generating three complementary PWM pairs or six independent PWM signals. A 12-bit timer PWM counter is common to all six channels. PWM resolution is one clock period for edge-aligned operation and two clock periods for center-aligned operation. The clock period is dependent on the internal operating frequency (f_{OP}) and a programmable prescaler. The highest resolution for edge-aligned operation is 125 ns (f_{OP} = 8 MHz) and the highest resolution for center-aligned operation is 250 ns (f_{OP} = 8 MHz). Some of the features of the PWM module are:

- Three complementary pairs or six independent PWM signals on six output pins
- Edge-aligned PWM signals or center-aligned PWM signals
- PWM signals polarity control
- 20-mA current sink capability on PWM pins
- Manual PWM output control through software
- Programmable fault protection (two FAULT input pins on the MR8 and four FAULT input pins on the MR16/32)
- Complementary mode featuring:
  - Dead-time insertion
  - Separate top/bottom pulse-width correction via current sensing (three pins on MR16/32) or programmable software bits (on all MR MCUs)

Registers Description

The main PWM module registers are:

1. **The Configuration Register (CONFIG)**
   - Determines the alignment of the PWM signals (bit 7 — EDGE)
   - Selects the top and bottom PWM polarities (bits 6:5 — BOTNEG, TOPNEG)
   - Enables or disables the complementary fashion (bit 4 — INDEP)
2. **The PWM Control Register 1 (PCTL1)**
   - Disables or re-enables the PWM pins in banks X and Y (bits 7:6 — DISX, DISY)[1]
   - Enables or disables PWM counter overflow interrupts (bit 5 — PWMINT)
   - Flags a PWM counter overflow (bit 4 — PWMF)
   - Configures the current correction bits (bits 3:2 — ISENS1, ISENS0)[2]
   - Loads prescaler, modulus, and PWM values to the working registers (bit 1 — LDOK)
   - Enables the PWM module (bit 0 — PWMEN)

Notes:
1. For for details concerning disable mapping, see the Fault Protection section of the data sheet.
2. The current distortion correction feature is not covered in this application note. Refer to the Output Control section of the data sheet.
3. The number of fault pins depends on the device and package. See the Pin Assignments section of the data sheet.

3. **The PWM Control Register 2 (PCTL2)**
   - Selects the PWM CPU load frequency (bits 7:6 — LDFQ1:LDFQ0)
   - Selects the PWM register used for the current correction when achieved by software (bits 4:2 — IPOL1:IPOL3)[2]
   - Selects the prescaler for the PWM clock frequency (bits 1:0 — PRSC1:PRSC0)
4. **The Fault Control Register (FCR)**
   - Enables or disables the fault interrupts of each fault input (bits 7, 5, 3, 1 — FINT4:FINT1)[3]
   - Selects the fault operating mode of each fault input (bits 6, 4, 2, 0 — FMODE4:FMODE1)[3]
5. **The Fault Status Register (FSR)**
   - Reflects the fault pin logic level (bits 7, 5, 3, 1 — FPIN4:FPIN1)[3]
   - Flags when a fault input as been asserted (bits 6, 4, 2, 0 — FFLAG4:FFLAG1)[3]
6. **The Fault Acknowledge Register (FTACK)**
   - Acknowledges the fault input flags when asserted (bits 6, 4, 2, 0 — FTACK4:FTACK1)[3]
7. **The PWM Output Control Register (PWMOUT)**
   - Selects the PWM output control, either manually or by the PWM generator (bit 6 — OUTCTL)
   - Determines the PWM pin logic level when configured as manual control (bits 5:0 — OUT6:OUT1)
8. **The PWM Counter Register (PCNTH:PCNTL)**
   - Displays the 12-bit up/down counter (center aligned) or up-only counter (edge aligned)
9. **The PWM Counter Modulo Register (PMODH:PMODL)**
   - Holds a 12-bit unsigned number that determines the maximum count for the up/down or up-only counter
10. **The PWM 1:6 Value Registers (PVAL1H:L, PVAL6H:L)**
    - Holds a 16-bit signed value that determines the duty cycle of the PWM
11. **The Dead-Time Write-Once Register (DEADTM) which:**
    - Holds an 8-bit value that determines the number of PWM cycles to turn off the PWM pairs when configured in complementary fashion
12. **The PWM Disable Mapping Write-Once Register (DISMAP)**
    - Holds an 8-bit value that determines which PWM pins will be disabled if an external fault or software disable occurs[1]

Notes:
1. For for details concerning disable mapping, see the Fault Protection section of the data sheet.
2. The current distortion correction feature is not covered in this application note. Refer to the Output Control section of the data sheet.
3. The number of fault pins depends on the device and package. See the Pin Assignments section of the data sheet.

Application Example

In this example PWM application, a three-phase AC induction motor will be driven by the microcontroller using an inverter topology. As shown in the Schematics section of this document, the PWM pins are connected to the gate driver IC. This gate driver is used to convert the MCU voltage levels to the required gate-to-source voltages of the switching transistors. The IGBTs used for this example are in the range of 10 V to 20 V typically. As a protection scheme, there is a sense resistor, where its voltage drop will represent the total current fed to the motor's coils. An amplifier and comparator will generate a FAULT signal to protect the power stage and the motor against an overcurrent condition.

With this topology, a set of duty cycle values are stored in FLASH as constants in order to generate the waveform shown in Figure 1.

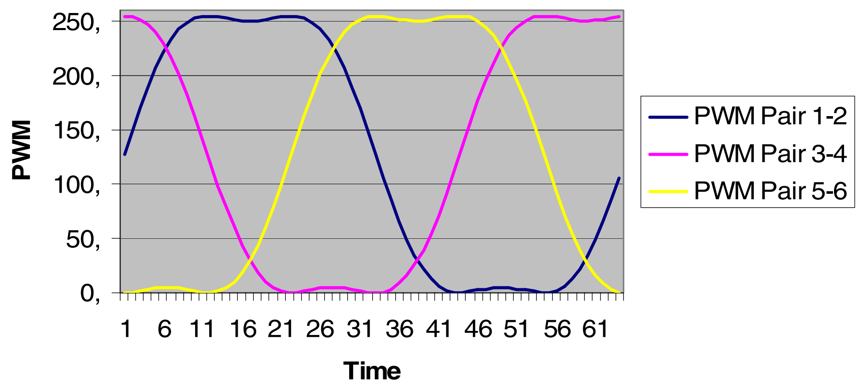

Figure 1. Third Harmonic Waves

When these waveforms are fed into the inverter transistors, the currents seen by the motor coils appear as shown in Figure 2:

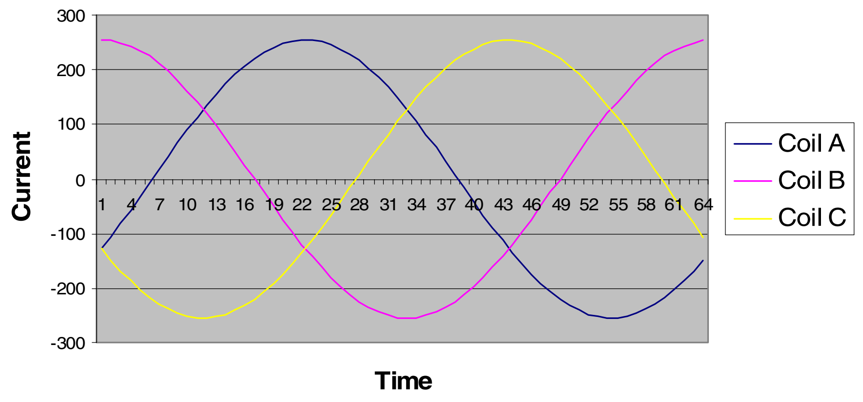

Figure 2. Coil Currents

As an example, a 60-Hz signal with these shapes will be generated by the microcontroller using the PWM at a 15.625 kHz. This frequency is just above the audible noise level and low enough to avoid electrical noise generation. The number of PWM cycles to achieve a 60 Hz signal would be:

15.625 kHz/60 Hz = 260.4167

For convenience, a power-of-two value of 256 PWM cycles can be used. At this point, a table of 256 duty cycle values is needed to generate a 60-Hz signal with a PWM frequency of 15.625 kHz, but there is a trade-off between the number of different duty cycle values and the CPU usage. Therefore, a different PWM load frequency will be used to relieve the CPU. With a PWM load frequency of each 4 PWM cycles, a table of 256 / 4 = 64 values is to be stored in FLASH for each PWM pair and CPU use is reduced significantly.

Another important parameter, known as dead-time insertion, must be considered on these applications. The dead-time value will depend on the motor coil's characteristics, the layout, and the gate driving circuitry. For this particular example, a dead-time value of 2 microseconds is inserted between the turn off of a channel (e.g., PWM 1) and the turn on of its complementary channel (e.g., PWM 2) to prevent a short circuit condition. That value can be computed as follows:

PWM module input clock (8 MHz with prescaler of 1) * 2 μs = 16.

Schematics

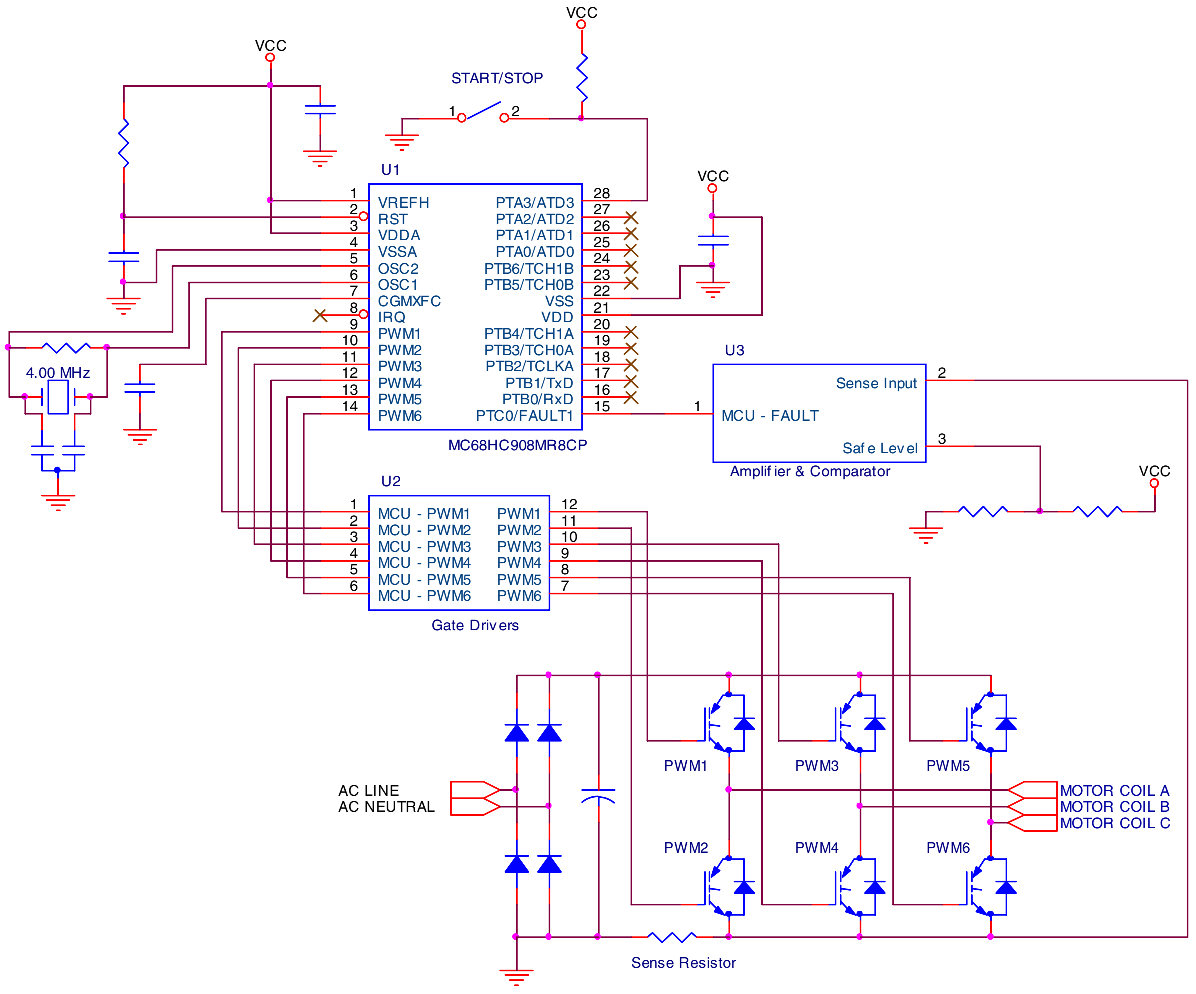

Figure 3. Example Schematic for Three-Phase Inverter

Code and Explanation

The following C code is an example using an MC68HC908MR8 microcontroller. This code configures and uses the PWM module to start and stop a three-phase AC induction motor. The software consists of the following steps:

- Initialize the PWM module.
- Check the input pin. If the pin is open, the motor is stopped. If the switch is closed, the motor is continuously running.
- The PWM duty cycles are reloaded continuously so that the output frequency seen by the motor coils is a three-phase sinusoid 60-Hz signal on a PWM frequency of 15.625 kHz. See the flowchart in Figure 4.

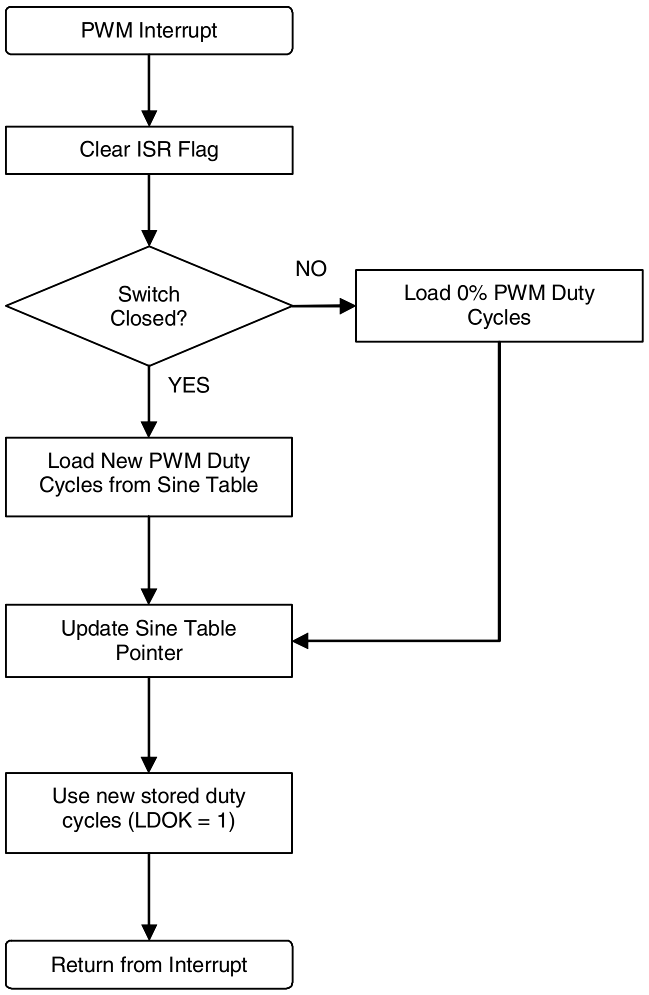
PWM Interrupt
Clear ISR Flag
Switch Closed?
NO
Load 0% PWM Duty Cycles
YES
Load New PWM Duty Cycles from Sine Table
Update Sine Table Pointer
Use new stored duty cycles (LDOK = 1)
Return from Interrupt

Figure 4. PWM Interrupt Service Routine Flowchart

1. Initialization Routine

Following these steps, the user will be able to use the PWM for a typical motor application.

1.1 Configure the PWM fashion and polarity

```
CONFIG = 0x01;  /* Center aligned mode enabled, all PWM pins in positive polarity,
                   complementary mode enabled, COP disabled */
```

1.2 Initialize the PWM counter modulus to have a frequency of 15.625 kHz

The software initializes the PLL[1] to speed up the operating frequency to the maximum of 8 MHz with the external crystal of 4.00 MHz. In this step, the prescaler is also selected.

```
PMOD = 0x0100;  /* With a FOP of 8MHz, the PWM Frequency would be:
                                         Fbus           8 MHz
                   PWM Frequency = --------------- = --------- = 15.625 kHz
                                    '2' x PMODH:L     2 x 256
                   The '2' factor of the divisor is because a center-aligned
                   operation is used, so the PWM counter counts up and down. */
PCTL2 = 0x80;   /* Reload PWM values every 4 PWM cycles. Select a prescaler by 1 */
```

1.3 Initialize the disable mapping registers. When a FAULT occurs, turn off every PWM output. Configure the FAULT1 pin as automatic operation without interrupts.

```
DISMAP = 0xFF;  /* Disable all PWM pins when a FAULT occurs */
FCR = 0x01;     /* Fault 1 in Automatic mode without interrupts */
```

1.4 Select a dead-time of 2 microseconds. This value depends on the power electronics topology, the layout and the motor characteristics.

```
DEADTM = 0x10;  /* Deadtime of 2 us. This is dependent of the hardware topology */
                   DEADTM = Fbus x 2 us = 8 MHz x 2 us = 16 */
```

1.5 Initialize the duty cycle registers to the minimum value of 0%. In the complementary mode, it is sufficient to load duty cycle values to the odd PWM channels.

```
PVAL1 = 0x0000; /* Minimum duty cycle for pair PWM1:2 */
PVAL3 = 0x0000; /* Minimum duty cycle for pair PWM3:4 */
PVAL5 = 0x0000; /* Minimum duty cycle for pair PWM5:6 */
```

1.6 Load data to the working registers by setting the LDOK bit and enable the PWM module by setting the PWMEN bit. Enable PWM interrupts to reload duty cycle registers with new values.

```
PCTL1 = 0x22;   /* Load data to buffers by setting the LDOK bit and Enable
                   PWM Interrupts */
PCTL1 |= 0x01;  /* Enable the PWM module by setting the PWMEN bit */
```

Because an interrupt-based algorithm is being implemented, the global interrupt enable mask must be cleared as follows:

```
EnableInterrupts;    /* __asm CLI; */
```

1. PLL module is not covered in this application note. Refer to the Clock Generator Module section of the data sheet.

2. Interrupt Service Routine (ISR)

From this point on, the code execution is performed inside the PWM interrupt service routine. The ISR:

2.1 Clears PWM interrupt flag.

```
PCTL1 &= ~(0x10);        /* Clear PWM Interrupt Flag */
```

2.2 Reloads the PWM value registers with the next duty cycle value stored in a constant table. The value to be stored depends on the state of the push button. If it is read as logic 0, a value from the table is stored in the PWM channels, and if it is read as logic 1, a value of 0x0000 (0% duty cycle) is stored in the channels.

```
if ((PTA & 0x08) == 0x00) /* Check external switch connected to PTA3 */
{
    PVAL1 = PHASE_A[TableIndex];
    PVAL3 = PHASE_B[TableIndex];
    PVAL5 = PHASE_C[TableIndex];
}
else
{
    PVAL1 = 0x0000;      /* Minimum Duty Cycle on PWM1:2 pair */
    PVAL3 = 0x0000;      /* Minimum Duty Cycle on PWM1:2 pair */
    PVAL5 = 0x0000;      /* Minimum Duty Cycle on PWM1:2 pair */
}
```

2.3 Updates the TableIndex variable, which is used to point to the next value to be stored in the PWM channels. It is restarted when a maximum value of 64 is reached.

```
TableIndex++;            /* Update Table Index */

if (TableIndex == 64)
{
    TableIndex = 0;      /* Reset Table Index */
}
```

2.4 Sets the LDOK bit to use the new stored values.

```
PCTL1 |= 0x02;           /* Load new PWM values */
```

Considerations

This example code was developed using Metrowerks CodeWarrior IDE version 3.0 for HC08, and was expressly made for the MC68HC908MR8 in the DIP package. Changes to the code may be required to for use with other MCUs because pin availability differ for among devices. Table 1 shows the available pins for each member and package option for the MC68HC08MR Series of MCU.

Table 1. PWM Pin Availability

			Pins						
			PWM Pins 1–6	PWM GND Pin	FAULT1	FAULT2	FAULT3	FAULT4	ISENS Pins
Microcontroller	MR4/8	32-Pins LQFP	Yes	No	Yes	No	No	Yes	No
		28-Pins DIP/SOIC	Yes	No	Yes	No	No	No	No
	MR16/	64-Pin QFP	Yes	Yes	Yes	Yes	Yes	Yes	Yes
		56-Pin	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Conclusion

This document describes how to use the PWM module to drive a three-phase AC induction motor with an MC68HC908MR8 microcontroller. For further information on how an application of this type is implemented in detail, visit the Freescale Semiconductor web site: http://freescale.com.

References

Refer to the following documents for more information on subjects in this application note.

- AN2876SW.zip: Complete software files for the example described in this application note
- MC68HC908MR8 data sheet
- MC68HC908MR32 data sheet
- AN1712: *Get Your Motor Running with the MC68HC708MP16*
- AN2154: *Low-Cost, 3-Phase, AC Motor Control System with Power Factor Correction Based on MC68HC908MR32*
- AN2355: *Sensorless BLDC Motor Control on MC68HC908MR32 Software Description*
- DRM007: *BLDC Motor Control Board for Industrial and Appliance Applications*
- Freescale Semiconductor's Motor Control Automotive Application page