Freescale Semiconductor
Application Note

Document Number: AN3844
Rev. 0, 11/2009

MC56F8006 New Features: PDB, PGA, ADC, and PWM

by: XiangJun Rong and KuangXin
RegionTechnical Information Center

1 Introduction

This application note introduces the new MC56F8006 digital signal and controller (DSC) features that include pulse width modulators (PWM), programmable delay blocks (PDB), programmable gain amplifiers (PGA), and analog-to-digital converters (ADC).

- It describes the ADC triggering mode
  - The PWM reload synchronization event triggering the PDB then the PDB triggering the ADC after a delay
  - The PDB triggering the PGA and the PGA triggering the ADC
  - The PDB triggering the ADC at a constant cycle
  - The external signal triggering the ADC

Contents

- It provides the configuration of the corresponding register, the example code, and introduces the PWM features:
  - PWM asymmetric mode in center alignment mode
  - PWM asymmetric mode in edge alignment mode

Both modes are important for phase shift full bridge switch mode power supply (SMPS) control.

2 New MC56F8006 Features and Advantages

2.1 MC56F8006 Advantages

The MC56F8006/8002 is a Freescale hybrid digital signal processor (DSP) and microcontroller (MCU). The MC56F8006/8002 uses the MC56F56800E core that is based on a dual Harvard style architecture consisting of three execution units operating in parallel. This makes it suitable to both the DSP and control application fields.

For the MC56F8006/8002 peripherals, see Table 1. The peripherals include on-chip PWM, PDB, PGA, ADC, QuadTimer, serial communication interface (SCI), serial peripheral interface (SPI), inter-integrated circuit (I2C), and so on. The MC56F8006/8002 is well-suited for many applications because of its multiple peripherals, low power consumption, and low cost:

- Board mounted power supplies
- Industrial power supplies
- Power metering
- Low-end three-phase BLDC motor control
- PM synchronous motor control
- Handheld tool motor control
- Arc detection

Table 1. MC56F8006 peripherals

Feature	MC56F8006			MC56F8002		
	28-pin	32-pin	48-pin	28-pin	32-pin	48-pin
Flash memory size (Kbytes)	16			8 or 12		
RAM size (Kbytes)	2					
Analog comparators (ACMP)	1	1	3	1	1	3
Analog-to-digital converters (ADC	2					
Unshielded ADC inputs	6	7	7	6	7	7
Shielded ADC inputs	9	11	17	9	11	17
Total number of ADC inputs	15	18	24	15	18	24
Programmable gain amplifiers (PGA)	2					
Pulse width modulators (PWM) outputs	Up to 6					
PWM fault inputs	3	4	4	3	4	4

Table 1. MC56F8006 peripherals (continued)

Inter-integrated circuit (IIC)	Yes
Serial peripheral interface (SPI)	Yes
Programmable interrupt timer (PIT)	Yes
Programmable delay block	Yes
General-purpose timers (GPT)	2
Computer operating properly (COP) timer	Yes
Phase-locked loop (PLL)	Yes
Relaxation oscillator (ROSC)	Yes
Real-time counter (RTC)	Yes
1 kHz oscillator	Yes
Crystal oscillator	Yes
On-chip clock synthesis (OCCS)	Yes
On-chip clock synthesis (OCCS)	Yes
Power management controller (PMC)	Yes

2.2 PDB, PGA, and ADC Introduction

The QuadTimer module can not trigger the ADC directly like the other Freescale DSC MCUs because there is no hardware connection internally between the PWM module, QuadTimer, and the ADC module.

Compared to other DSC MCUs the MC56F8006/8002 has new modules, the PDB and PGA. The PDB module enables you to trigger the ADC from a multiple source including software, the PWM reload synchronization signal and external signal, general QuadTimer, and comparator outputs. The ADC sampling can be synchronized by a PWM reload synchronization signal and external signal, QuadTimer, and comparator with a programmable delay. Another new feature is the PGA that can sample and hold and analog signal, amplify the signals with differential analog input, and convert differential analog input into a single end output signal.

2.2.1 Programmable Delay Blocks (PDB)

Figure 1 is the timing diagram for the PDB, the triggering source of the trigger_input can be selected by TRIGSEL bits in the PDB control register.

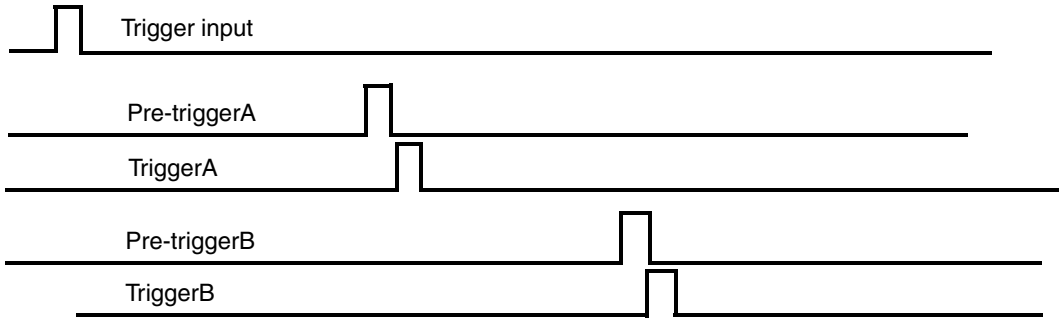

Figure 1. PDB illustration

Input Trigger Select

000 = TriggerIn0 is selected. On this device, this is CMP0_OUT
001 = TriggerIn1 is selected. On this device, this is CMP1_OUT
010 = TriggerIn2 is selected. On this device, this is CMP2_OUT
011 = TriggerIn3 is selected. On this device, this is the PWM SYNC signal
100 = TriggerIn4 is selected. On this device, this is ExtTrigger
101 = TriggerIn5 is selected. On this device, this is the output of General Purpose Timer 0 (T0)
110 = TriggerIn6 is selected. On this device, this is the output of General Purpose Timer 1 (T1)
111 = SWTRIG is selected

The pre-triggerX (pre-triggerA or pre-triggerB) is used to select the ADC (ADCCS1A or ADCSC1B) ping-pong buffer. The triggerX signal (triggerA or triggerB) is used to trigger the ADC to sample the analog channel. The latency time between trigger-input and triggerX output is specified by the PDB Delay A or B register. The PDB driving clock is the peripheral clock (32 MHz) and is divided by PRESCALER bits in the PDB control register, the dividing number can be 1, 2, 4, 8..128.

For example, if the MC56F8006 runs at 32 MHz, the PRESCALER bit in the PDB control register is binary 010, the delay number in the PDB Delay A Register is 1000 or 0 x 3E8, the delay time will be 4*1000*0.03125 μs = 125 μs.

2.2.2 Programmable Gain Amplifier (PGA) Module

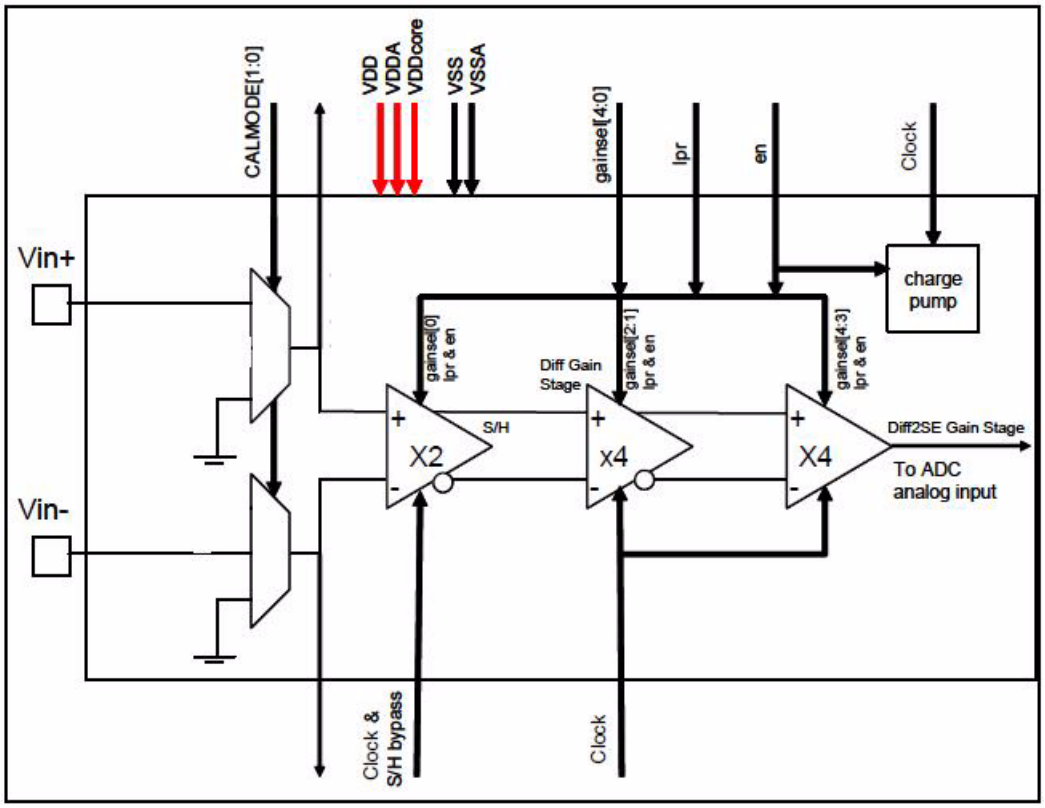

Figure 2. PGA diagram

The PGA is an amplifier implemented with switching capacitor technology. It has two functions, one is converting the differential input analog signal to a single end analog signal, the other is a programmable gain. If you use an external dedicated operational (OP) amplifier instead of the on-chip PGA to the amplifier and offset the analog signal, you can bypass the PGA and use the PDB to trigger the ADC directly. This does not have any affect on the function or performance of the triggering mode and sample. The PGA uses the switched capacitor mechanism, therefore the clock signal is required.

There are four parts for the PGA:

- Signal selection
- Sample and hold with a PGA
- Differential signal amplifier with programmable gain
- Differential input to single end signal part with a PGA

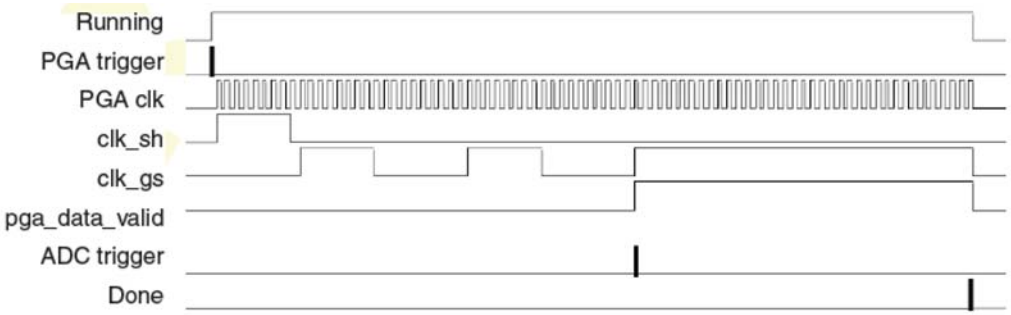

Figure 3. PGA timing

The PGA clock frequency must be the same as the ADC clock frequency although both of the clocks can be out of phase. The maximum PGA clock frequency is 8 MHz. After the ECC bit in the ADC status and the control register2 is set, the ADC module outputs the clock that is then routed to the PGA, see Figure 4. The ADIV bits in the PGA control register2 must be the same as the ADIV bits in the ADC configuration register for the PGA and the ADC to have the same frequency. If the MC56F8006 runs at 32 MHz, select the peripheral-clock/2 by setting the ADICLK bits in the ADC configuration register as 01 binary, and also setup the ADIV bits in the ADC configuration register to be 01 in binary. The ADC clock is then 32 MHz/(2*2) = 8 MHz. For the PGA and the ADC clock to have the same clock frequency of 8 MHz you must set the ADIV bits in the PGA control register2 as 01 in binary.

During the clk_sh high time, the sample/hold circuit of the PGA tracks the analog signal. On the falling edge of the clk_sh signal, the sample/hold circuit holds the analog signal. The high time of the clk_sh is the aperture time of sample/hold that is 8*(PGA clock cycle time) = 8*0.125 µs = 1 µs. See Figure 3. When the PGA is enabled, whether you set the TM bit in the PGA control register0 to use the software trigger or the hardware trigger, the PGA generates the new pre-triggerX and triggerX signal for the ADC to use. X can be either A or B. See PGA latency for Figure 3.

For example, assume that the PGA clock frequency is 8 MHz, if you set the PGA's CNTL2[NUM_CLK_GS] as $2 that corresponds to 2 assertions of clk_gs per conversion, the latency between the edge of the PGA trigger and the edge of the ADC trigger is:

(9 + NUM_CLK_GS *18)*0.125 µs = (9 + 2*18)*0.125 µs = 5.625 µs

The PGA trigger is the software trigger or hardware trigger from the PDB. You can use the software trigger by setting the SWTRIG bit in the PGA control register when the TM bit is set. When software triggering has been enabled by writing CNTL0[TM]=1, writing a 1 to the SWTRIG bit in the PGA control register initiates a conversion. See Figure 3.

NOTE

NUM_CLK_GS must be set to a minimum value of 001 when using software triggers.

2.2.3 Analog-to-Digital Converters (ADC)

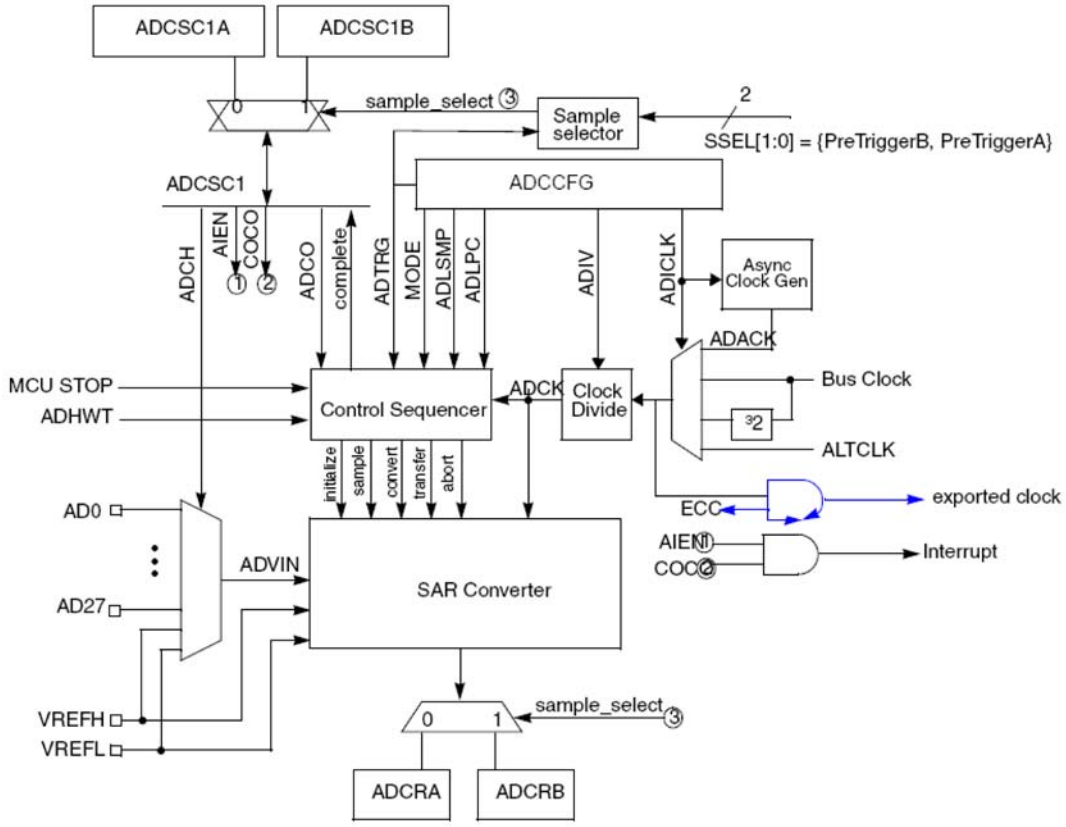

Figure 4. ADC diagram

The ADC resolution can be set as 8 bits, 10 bits, or 12 bits. The total conversion time depends on the sample time mode (determined by ADLSMP), the conversion mode (8-bit, 10-bit, or 12-bit), and the frequency of the conversion clock (f_{ADCK}). There is a table in the reference manual titled *MC56F8006 Reference Manual* that lists the condition and corresponding conversion time.

A unique feature of the ADC is its triggering mode. The ADC can be triggered by the PDB or PGA modules. The ADC software triggering sources include the PDB, PGA, and the ADC itself. The ADC hardware triggering sources include the PWM reload synchronization signal, external signal at the EXT_TRIGGER pin, comparator output, and QuadTimer output.

The ADC also supports the ping-pong mode. The ADC can sample two channels with triggering once. For software triggering, the ping-pong mode is not supported. If you use software triggering, you can only use the ADCSC1 and ADCRA register and get only one ADC sample for triggering the software once.

3 Multiple ADC Trigger Sources

3.1 ADC Sampling at a Constant Cycle

This ADC triggering mode triggers ADC in a constant cycle. All ADC samples are guaranteed to be acquired at a constant cycle by the hardware, which is preconditioned for the FIR/IIR and FFT operation. The digital signal processing is an advantage of the MC56F8006/8002. The PDB provides a mechanism to trigger the ADC at a constant cycle specified by the count modulus register. After you initiate the trigger via the software by setting the SWTRIG bit in PDB_SCR, the PDB triggers the ADC one by one in a constant cycle. Another option for the application is using the QuadTimer to trigger the PDB, and the PDB to trigger the ADC.

The following is the code for the constant cycle mode:

```
#define CONSTANT_SAMPLE 1
#if CONSTANT_SAMPLE
//configure the software trigger of PDB so that the ADC can sample at a constant cycle
setReg(PDB_SCR,0x900); //if the PDB_SCR is set as 0x800,it is okay
setReg(PDB_MOD,0x4000); //setting ADC sample cycle
setRegBitGroup(PDB_SCR, AOS,1);
setRegBit(PDB_SCR,CONT);
setRegBitGroup(PDB_SCR,TRIGSEL,7);
setRegBit(PDB_SCR,ENA);
setRegBit(PDB_SCR,SWTRIG); //start software trigger
#endif
```

The value in the PDB_MOD determines the ADC sample cycle time that the PDB_DELAY register does not affect.

The code was developed with the processor expert (PE) platform that is embedded in the CodeWarrior tools. This is the ADC initialization.

```
void AD1_Init(void)
{
  EnUser = TRUE;                       /* Enable device */
  OutFlg = FALSE;                      /* No measured value */
  ModeFlg = STOP;                      /* Device isn't running */
  /*
  setReg16(ADC0_ADCSC1B, 0x1F);        /* Disable the module */
  /* ADC0_ADCSC2:
??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,ADACT=0,ADTRG=0,??=0,??=0,??=0,ECC=1,REFSEL=1 */
  setReg16(ADC0_ADCSC2, 0x05);         /* Disable HW trigger */
  /* ADC0_ADCCFG:
??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,ADLPC=0,ADIV=3,ADLSMP=1,MODE=1,ADICLK=1 */
  setReg16(ADC0_ADCCFG, 0x75);         /* Set prescaler bits */
  /* ADC0_ADCSC1A: ??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,COCO=0,AIEN=0,ADCO=0,ADCH=0x1F */
  setReg16(ADC0_ADCSC1A, 0x1F);        /* Disable the module */
  AD1_HWEnDi();                       /* Enable/disable device according to the status flags */
}
```

For the ADC Status and Control Registers ADC0_ADCSC1B and ADC0_ADCSC1A, the ADCO (bit5) in both the ADC0_ADCSC1B and the ADC0_ADCSC1A registers must be cleared so that the ADC can convert only once for each hardware trigger. The ADCH bits are used to select the analog channels index.

For the Status and Control Register2 (ADCSC2), the ADTRG bit must be set for the hardware trigger to be selected. The Ecc bit must be set if you use the PGA, this is because the PGA clock is supplied by the ADC module when the bit is set.

ADICLK bits are used to select the clock source for the ADC configuration register. The ADIV bits are used to divide the clock that must be the same value as the ADIV bits in the PGA control register. The ADLSMP bit must be set (long sample time), and the ADLPC must be cleared (high speed configuration).

```
setRegBit(ADC0_ADCSC2,ADTRG);

//set the ADC in one conversion mode
clrRegBit(ADC0_ADCSC1A,ADCO);
clrRegBit(ADC0_ADCSC1B,ADCO);
```

3.2 PWM Reload Synchronization Signal Triggering ADC

The DSP56800/E focuses on motor control applications for a sinusoidal type motor. For example, the AC induction motor (ACIM) and permanent magnetic synchronous motor (PMSM) use the field oriented control (FOC) algorithms to control the motor. For hardware circuit on a power stage board, use three resistors to sense current signals. The three resistors are connected to the bottom branches of 3 phase bridges. For the brushless DC motor (BLDC) or the switched reluctance (SR) motor, if the internal current loop is needed the current signal also needs to be tested.

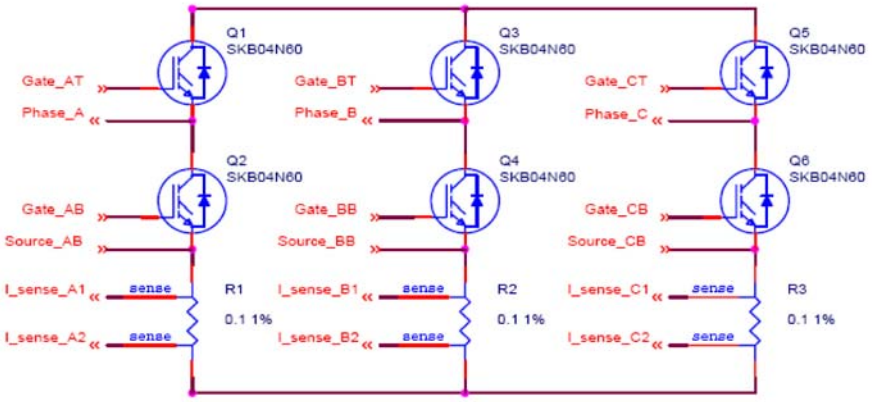

Figure 5. Three branch current sensor using resistors

The following circuit was used as a signal conditioner for the ADC to sample the current. Because the on-chip ADC can accept only voltage from the GND to the VDDA (3.3 V) an offset circuit was made

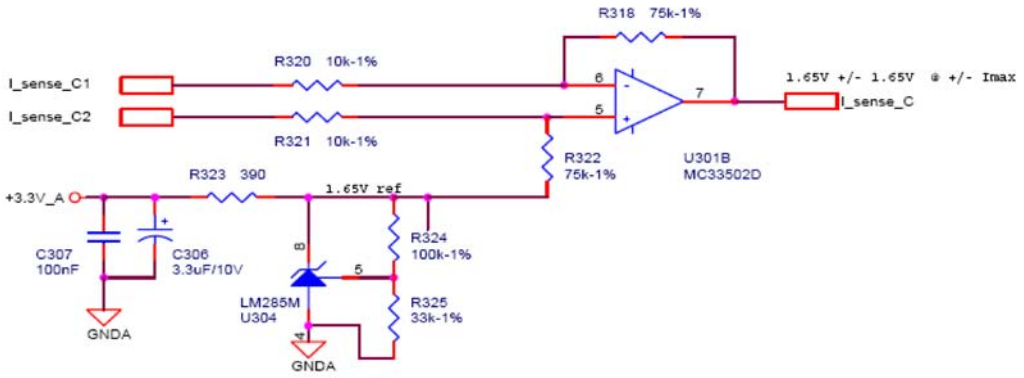

Figure 6. Signal offset circuit

The voltage shapes of two different PWM periods are shown in Figure 7. The voltage shapes correspond to center-aligned PWM sinewave modulation. The best moment of current sampling is in the middle of the PWM period.

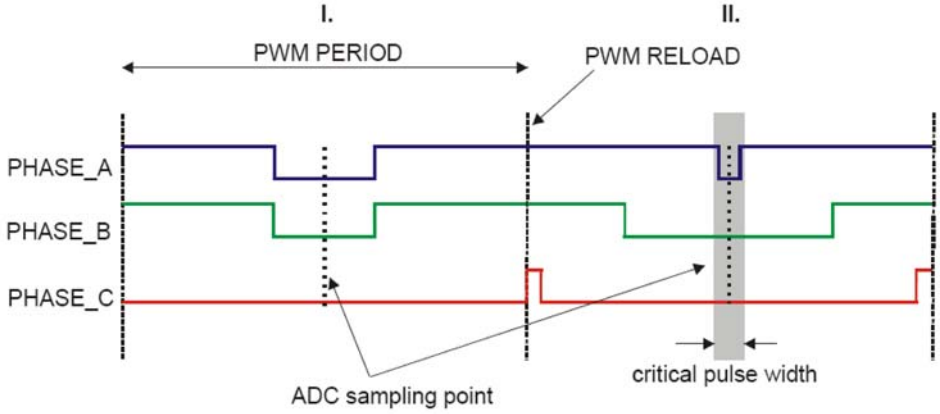

Figure 7. PWM signal timing diagram

A decision must be made about which phase the current must be sampled and calculated. The software code determines which channels should be sampled based on the sector information. The SVPWM function in PE provides the sector information.

3.2.1 PWM Synchronization Signal Triggers ADC for DSP5680x and MC56F83xx Family

For the MC56F83xx and MC56F8000 families (except for the MC56F8006/8002), the PWM reload synchronization signal triggers TimerC2. TimerC2 triggers the ADC after a programmable delay.

In the MC56F83xx and MC56F8000 families including the MC56F8013/14/23/25/37, there are no PDB or PGA blocks. There is a hardware wire connection between the PWM and TimerC2 that enables the PWM Reload synchronization signal to trigger TimerC2. TimerC2 triggers the ADC with a programmable delay.

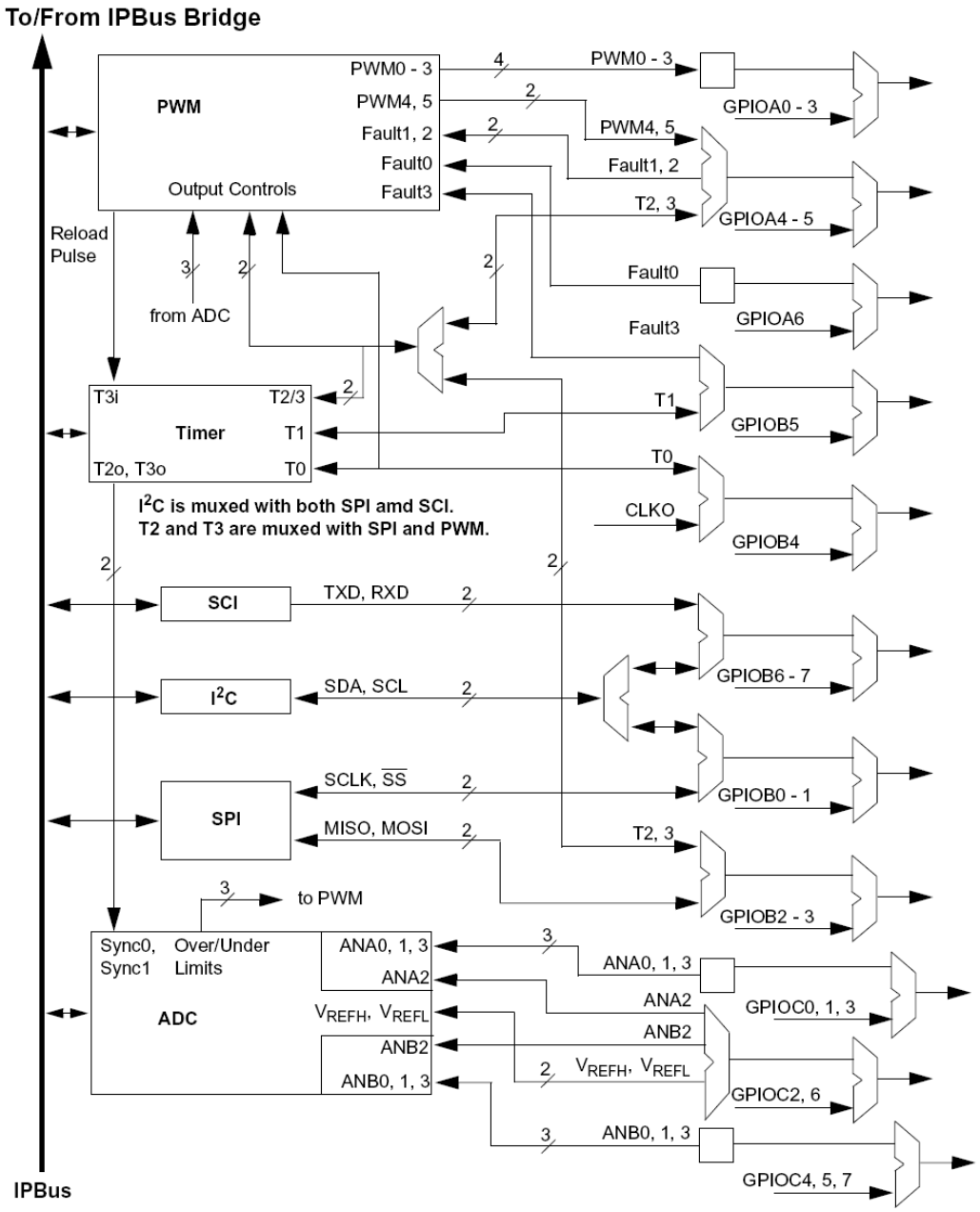

Figure 8. ADC triggering diagram

For the ADC to be triggered the PWM Reload synchronization signal is connected to the input of TimerC3, and the TimerC3 output signal is connected to ADC Sync0 or Sync1. See Figure 8.

This is the example code used for the PWM to trigger TimerC2 and the TimerC2 to trigger the ADC based on the MC56F8013.

```
void main(void)
{
  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.                    ***/

  /* Write your code here */
  setReg(ADC_ADCR1,0x1801);
  setRegBit(ADC_ADCR2,SIMULT);
//  clrRegBit(SIM_CONTROL,TC3_INP);
  setRegBit(SIM_CTRL,TC3_INP); //This bit should be set rather than be cleared as described by
the data sheet
        AD1_EnableIntTrigger();
        PWMC1_OutputPadEnable();

  /* Write your code here */

//AD1_Measure(FALSE);
  for(;;) {}
}
The following code is used to initialize QuadTimerC2 module
void TMR4_Init(void)
{
  setReg16(TMR3_CMP1, 4096);           /* Set the Compare register 1 */
  setReg16(TMR3_CMP2, 4096);           /* Set the Compare register 2 */
  setReg16(TMR3_CAP, 0);               /* Set the Capture register */
  setReg16(TMR3_LOAD, 0);              /* Set the Load register */
  setReg16(TMR3_CNTR, 0);              /* Set the Counter register */
  /* TMR3_SCR:
TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FO
RCE=0,OPS=0,OEN=1 */
  setReg16(TMR3_SCR, 1);               /* Set the Status and control register */
  /* TMR3_CMPLD1: COMPARATOR_LOAD_1=4096 */
  setReg16(TMR3_CMPLD1, 4096);         /* Set the Comparator load register 1 */
  /* TMR3_CMPLD2: COMPARATOR_LOAD_2=4096 */
  setReg16(TMR3_CMPLD2, 4096);         /* Set the Comparator load register 2 */
  /* TMR3_COMSCR:
DBG_EN=0,??=0,??=0,??=0,??=0,??=0,??=0,TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=1,CL1=1 */
  setReg16(TMR3_COMSCR, 5);            /* Set the Comparator status/control register */
  /* TMR3_CTRL: CM=6,PCS=8,SCS=3,ONCE=0,LENGTH=1,DIR=0,Co_INIT=0,OM=5 */
  setReg16(TMR3_CTRL, 53669);          /* Set the Control register */
}
```

3.2.2 PWM Reload Synchronization Signal Triggers PDB and PDB Triggers ADC

For the MC56F8006/8002 MCU there is no internal connection between the PWM and QuadTimer, but the MC56F8006/8002 MCUs add the programmable delay block (PDB) and programmable gain amplifier (PGA) that enables the PWM reload synchronization signal to trigger the PDB, and the PDB to trigger the ADC via the PGA, or directly.

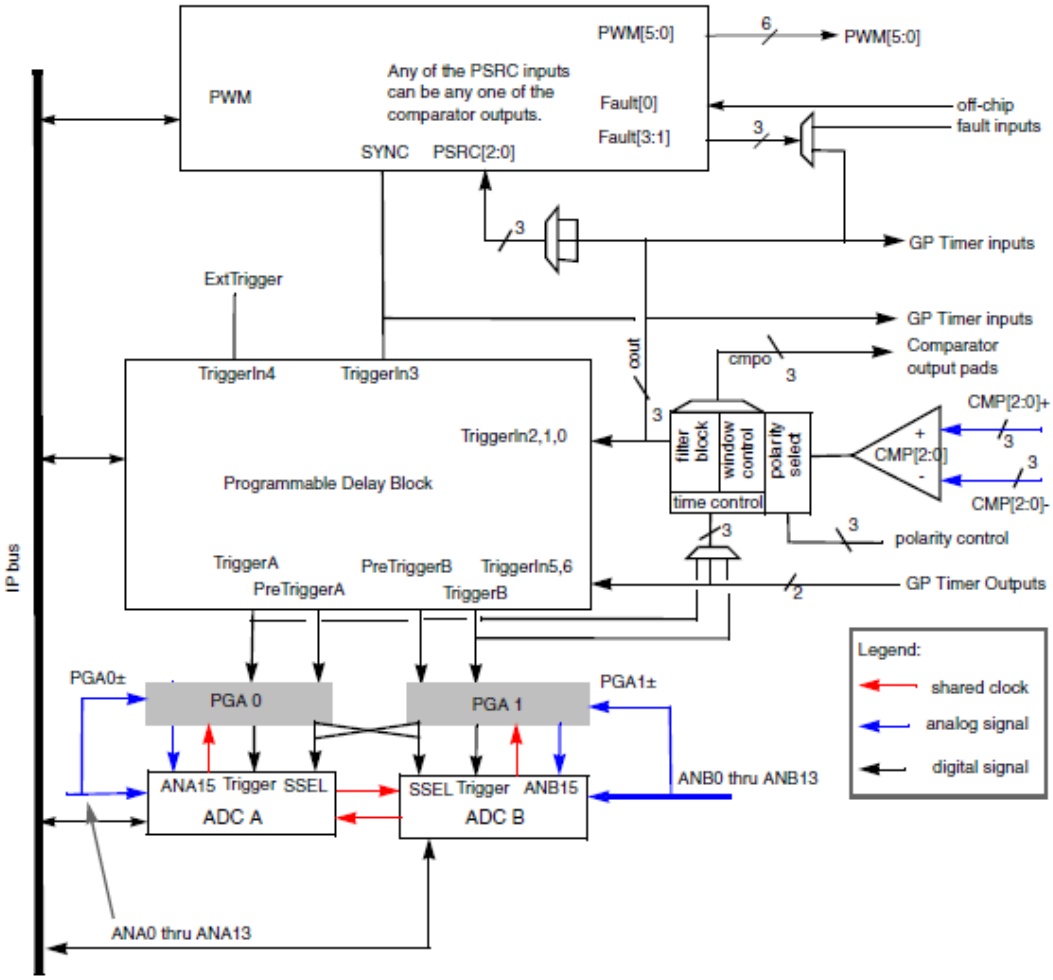

Figure 9. PDB triggering source diagram

The PWM reload synchronization signal can trigger the PDB as Trigger source3, after a delay the PDB can trigger the ADC directly or via the PGA module. See Figure 9.

This is the PDB module configuration source code for the PDB to trigger the ADC.

```
setReg16(PDB_DELAYA, 0x09C4);
setReg16(PDB_DELAYB, 0x00);
setReg16(PDB_MOD, 0x7FFF);
setReg16(PDB_SCR, 0x040C);
setRegBit(PDB_SCR,ENA);
```

The value in PDB_DELAYA or PDB_DELAYB determines the delay time, the value in PDB_MOD does not take affect for this mode.

For the PDB Status and Control register, if you use TriggerA to trigger the ADC the AOS bits must be 01 in binary, if you use TriggerB you must set BOS bits as 01 in binary. To trigger only the ADC once for each PWM reload synchronization edge both the CONT bit and SWTRIG must be cleared for the PWM reload signal. The ENA bit must be set to enable the PDB.

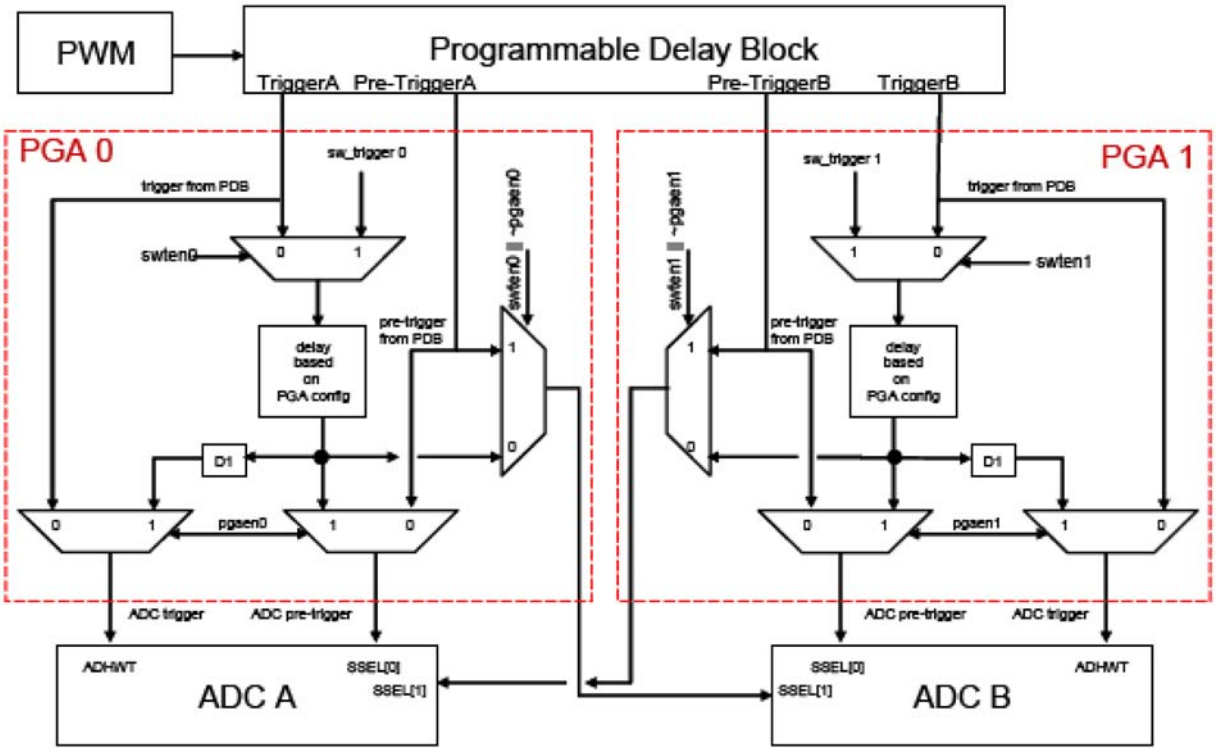

Figure 10. PGA diagram

3.2.3 PWM Reload Synchronization Signal Triggers PDB, PDB Triggers PGA, and PGA Triggers ADC

The PDB can trigger the PGA. The PGA can generate new timing to trigger the ADC after an intrinsic delay. If the EN bit in the PGA control register is set, the ADC module is triggered by the PGA directly rather than the PDB. See Figure 10.

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	TM	GAINSEL					LP	EN
Reset:	0	0	0	0	0	0	0	0

Figure 11. PGA control register 0

In this mode, the TM bit must be cleared to enable the hardware trigger, both the LP and EN bit must be set, and GAINSEL determines the gain of the PGA.

	7	6	5	4	3	2	1	Bit 0
Read:	0	0	BP	CALMODE		CPD		
Write:								
Reset:	0	0	0	0	0	0	1	0

Figure 12. PGA control register 1 (CNTL1)

The charge pump divisor (CPD) bits must be set as 2 because it is the default value. CALMODE must be set as 00 in binary.

	7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	NUM_CLK_GS			ADIV	
Write:			SWTRIG					
Reset:	0	0	0	0	1	1	1	0

Figure 13. PGA control register 2

The ADIV bits in the above register must be set at the same value as ADCCFG[ADIV] within the associated ADC module, this determines the baud rate for the PGA clk. This parameter controls how many times the gain and differential to single-ended stages of the PGA are clocked per conversion (NUM_CLK_GS+1). The default value of this field is 0x3.

In the PWM reload synchronization signal triggering the PDB, PDB triggering the PGA, and PGA triggering the ADC mode. The SWTRIG must be cleared.

```
setReg16(PGA0_CNTL1, 0x02);
setReg16(PGA0_CNTL2, 0x0F);
setReg16(PGA0_CNTL0, 0x04);
//Enable PGA
setRegBit(PGA0_CNTL0,EN);
```

3.3 External Signal Triggering PDB and the PDB Triggering ADC

In this mode, after a programmable delay by the PDB, an external signal source connected to EXT_TRIGGER pin (multiplexed with GPIOC3 or pin 46 for 48 pin LQFP package of the MC56F8006) can trigger the PDB. The PDB generates timing to trigger the ADC directly or via PGA. This way the on-chip ADC can synchronize with external signal.

The following is the code used by the external signal to trigger the PDB and the PDB triggers the ADC.

```
 Void Main()
{
#if      EXTERNAL_TRIGGER
//       set GPIOC3 as External Trigger source  to trigger PDB
                setRegBit(GPIO_C_PER,PE3);
//configure the PDB External Trigger source
         setRegBitGroup(PDB_SCR,TRIGSEL,4);
         //Result: connect GPIOC3 to a 5KHz clock signal from oscilloscope, I can see that Pin
1 of J2 toggles on 56F8006Demo board
#endif
}
```

```
/*In the ADC interrupt service routine after the conversion is complete, the GPIO_E0 pin (Pin1
of J2 on the MC56F8006 Demo board) is toggled at 5 KHZ while the EXT_TRIGGER pin (GPIO_C3)is
connected to a 5 KHz clock signal from an oscilloscope.*/

volatile word SampleValue = 0;// store the ADC sample value
void AD1_OnEnd(void)
{
  changeRegBit(GPIO_E_DR,D0);

PE_AD1_GetChanValue((unsigned int*)&SampleValue);// Get the ADC value
}
```

3.4 On-Chip ADC Ping-Pong Mode

The ping-pong mode of the on-chip ADC means that you can get two samples (or sample two different channels) using only one ADC converter by sampling once.

Each ADC converter (ADC0 or ADC1) has two ADC status and control registers (ADCCS1A and ADCSC1B) and two ADC result registers (ADCRA and ADCRB), although there is only one hardware ADC converter. In ping-pong mode, the ADC converter can do two conversions consecutively, and save the ADC sample into the ADCRA and ADCRB registers, therefore the ADC conversion time requirement has to be met. The latency between pre-triggerA and pre-triggerB must be greater than the ADC conversion time, otherwise, you can only save one ADC sample the ADCRA register.

Software triggering does not support ping-pong mode. If you use software triggering, you can only use ADCSC1 and ADCRA modules and get only one ADC sample in a software triggering.

The following is the ADC pin-pong mode code. It uses the PWM module to trigger the PDB, then the ADC.

```
  setReg16(ADC0_ADCSC2, 0x05);         /* Disable HW trigger */
  /* ADC0_ADCCFG:
??=0,??=0,??=0,??=0,??=0,??=0,??=0,??=0,ADLPC=0,ADIV=3,ADLSMP=1,MODE=1,ADICLK=1 */
  setReg16(ADC0_ADCCFG, 0x75);         /* Set prescaler bits */
        //set the ADC hardware trigger mode
        setRegBit(ADC0_ADCSC2,ADTRG);
        //set the ADC in one conversion mode
        clrRegBit(ADC0_ADCSC1A,ADCO);
        clrRegBit(ADC0_ADCSC1B,ADCO);

#if     PWM_TRIGGER
        //output PWM reload sync signal to GPIOB5 pad
        setRegBit(GPIO_E_PER,PE4);
        setRegBit(GPIO_E_PER,PE5);
#if PING_PONG
  /* PDB_DELAYA: DELAYA=0x03E8 */
  setReg16(PDB_DELAYA, 0x03E8);
  /* PDB_DELAYB: DELAYB=0x07D0 */
  setReg16(PDB_DELAYB, 0x07D0);
  /* PDB_MOD: MOD=0x7FFF */
  setReg16(PDB_MOD, 0x7FFF);
          setReg16(PDB_SCR, 0x090F);
          setReg16(ADC0_ADCSC1B, 0x46);
          setReg16(ADC0_ADCSC1A, 0x48);
#endif
```

```
        setRegBitGroup(SIM_GPSA,GPS_A5,1);
        setRegBit(PWM_SYNC,SYNC_OUT_EN);
#endif
```

In the above code the PWM is 5 KHz, therefore the frequency of the PWM reload event is 5 KHz, and the ADC sample frequency will also be 5 KHz. The code sets the PDB DelayA register as 1000 in decimal that corresponds to 31.25 µs, and the PDB DelayB register as 2000 in decimal corresponds to 62.5 µs delay. Therefore, 31.25 µs latency between pre-triggerA and pre-triggerB is greater than the maximum ADC conversion time, which is 6 µs for the ADC clock in 8 MHz.

3.5 Software Triggering Mode

In software triggering modes, you can set one bit in a register by code to trigger the ADC to sample the analog signal. You can start an ADC conversion whenever and wherever, this is because three modules can trigger the ADC (PDB, PGA, and ADC). There are three methods for the software to trigger the ADC.

1. Triggering mode — Software triggering the PDB, the PDB triggering the PGA, the PGA triggering the ADC. This method is used in constant cycle sampling, as described in Section 3.1, “ADC Sampling at a Constant Cycle,” on page 8.
2. Software triggering the PGA, the PGA triggering the ADC.

Setting TM bit in the PGA control register0 enables the software trigger. Writing “1” to the SWTRIG bit in the PGA control register0 initiates the ADC sample once. In this mode, NUM_CLK_SEL must be set to a minimum value of “001” when using the software trigger. It is required for the signal to propagate through both gain stages of the PGA

3. Software triggering the ADC directly — Writing “1” to ADTRG bit in the ADCSCx register can start an ADC conversion.

This is the code for ADC software triggering.

```
#define  ADC_VECTOR_NO     12

unsigned int sample;
void delay(void);
void main(void)
{
  /* Write your local variable definition here */

  /*** Processor Expert internal initialization. DO NOT REMOVE THIS CODE!!! ***/
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.                    ***/

  /* Write your code here */

INTC_IAR0 = ADC_VECTOR_NO<<8;
 setRegBit(SIM_PCE,ADC0);
  setRegBit(SIM_PCE,ADC1);
  setReg16(ADC1_ADCSC2,0x04);
   setReg16(ADC1_ADCCFG,0x75);
         setReg16(ADC1_ADCSC1A, 0x40);
    setReg16(ADC1_ADCSC1A, 0x40);
```

```
      __EI(0);

  for(;;)
  {
         delay();
         while(getRegBit(ADC1_ADCSC2,ADACT))  {}
         setReg16(ADC1_ADCSC1A, 0x40);
    setReg16(ADC1_ADCSC1A, 0x40);
         asm(nop);
  }
}

#pragma interrupt saveall
void ADC_complete(void)
{
         //read data from register to a variable
         sample=getReg(ADC1_ADCRA);
         asm(nop);
}
void delay(void)
{
         asm(nop);
return;
}
```

4 MC56F8006/8002 PWM Module Features

4.1 Asymmetric PWM Output in Center-Alignment Mode

To reduce the IGBT/MOSFET device power consumption, ZVS/ZCS technology was used to reduce power consumption by establishing a resonant circuit called the phase shift full bridge method for large power DC/DC converter based on switch mode power supply technology. For the phase shift full bridge method, use four PWM signals to drive the H MOSFET bridge. In the application, the PWM module is configured as center-aligned and complementary operation mode. Set up the PWM module in asymmetric PWM output mode by setting the ICCx bits in PWM internal correction control register (ICCTRL). After setting the ICCx bits, the PWM module selects the even PWM value registers to use when counting up and the odd PWM value registers when counting down, this way the PWM pairs can be shifted

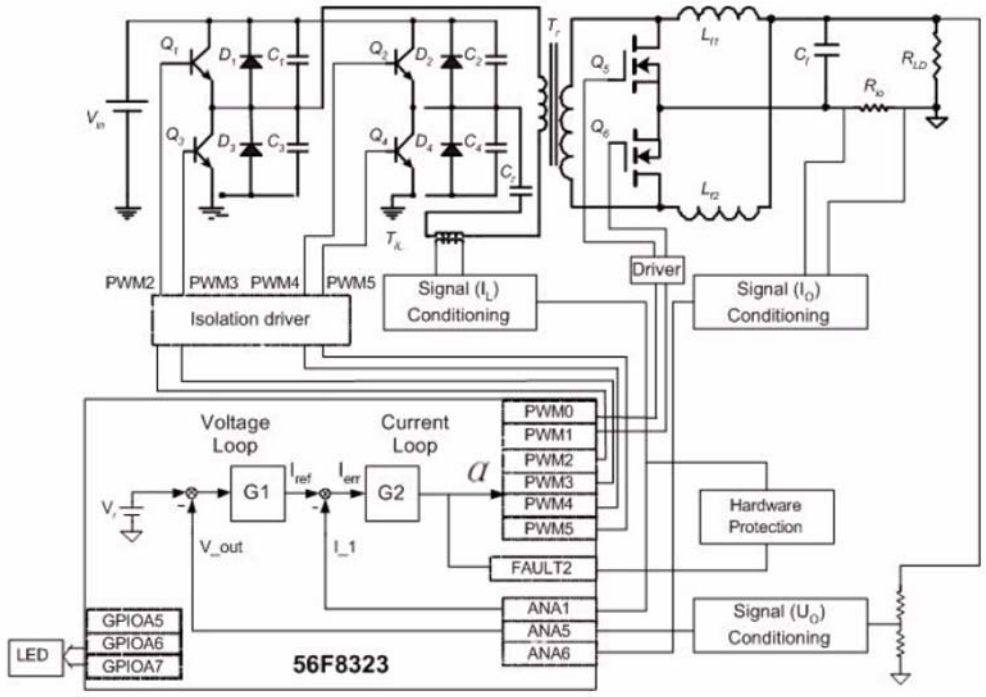

Figure 14. Phase shift full bridge DC/DC converter diagram

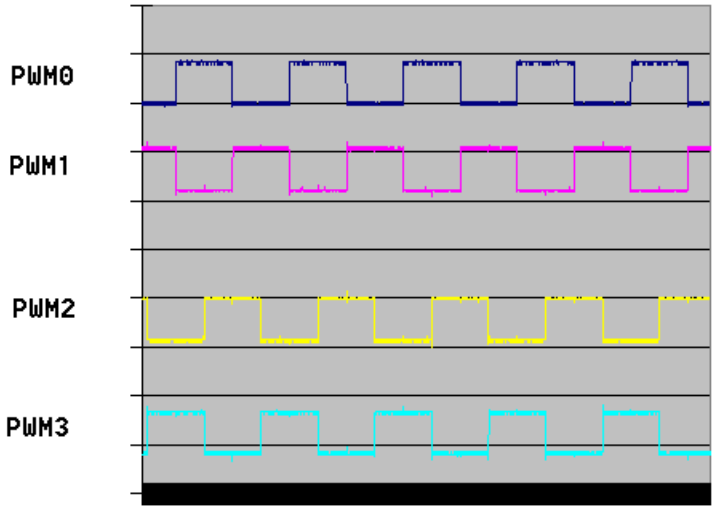

Figure 15. Asymmetric PWM output in center-alignment mode

Figure 15 is a snapshot of the MC56F8006 PWM signals when the MC56F8006 runs the following code `#define PWM_CENTER_ALIGNMENT_ASYMMETRIC 1` as a compiler option. It demonstrates the PWM output wave form when the PWM is set up in an asymmetric and center-alignment mode.

4.2 Asymmetric PWM Output in Edge-Alignment Mode

When the PWM is configured in edge aligned operation mode during the complementary mode an asymmetric PWM pulse can be obtained. This is a unique feature of the MC56F8006/8002. These PECx registers to be XOR'ed together prior to the complementary logic and deadtime insertion. Using this method, three pulses can be obtained with arbitrary phase shift. For example, assume you would like to control six independent lamps with six PWM signals in both center-aligned or edge aligned mode. Without phase shift the current flowing to each lamp reaches its peak at the same time, this results in a current rush. If this asymmetric method to generate the PWM signal with arbitrary phase shift is used, power with time can be distributed.

For example, the PEC0 bit in the PWM ICCTRL controls the PWM0 and PWM1 complementary pair. Setting the PEC0 bit allows either the PWMVAL0 and PWMVAL1 to activate the PWM pulse and the other to deactivate the pulse for the PWM signal to be generated at any time.

These area new features only for the MC56F8006/8002 MCU, not implemented in MC56F83xx family and MC56F8013/14/23/25/and 37.

This is the code to set up the PWM in asymmetric center-alignment or edge-alignment mode.

```
#define PWM_CENTER_ALIGNMENT_ASYMMETRIC 1
#define PWM_EDGE_ALIGNMENT_ASYMMETRIC 0
void main(void)
{
  /* Write your local variable definition here */

  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.                    ***/
         setReg(PWM_ICCTRL,0x00);

         //set PWM center-alignment mode in asymetric mode
         //connect pin9(PWM0 output) and pin30(PWM2 output) of J1, user can observe the waveform
of the PWM signal
         //pin2 of JM60 DEBUG connector is GND

#if PWM_CENTER_ALIGNMENT_ASYMMETRIC
         PWMC1_Disable();
         asm(nop);
         setReg(PWM_ICCTRL,0x03);
           setReg16(PWM_CMOD, 0x0C80);
           setReg16(PWM_VAL0,800);
           setReg16(PWM_VAL1,2400);
           setReg16(PWM_VAL2,2400);
           setReg16(PWM_VAL3,800);
           PWMC1_Enable();
#endif

#if PWM_EDGE_ALIGNMENT_ASYMMETRIC
         PWMC1_Disable();
         asm(nop);
         setRegBit(PWM_CNFG,EDG);
         setRegBitGroup(PWM_ICCTRL,PEC,0x03);
           setReg16(PWM_CMOD, 0x0C80);
           setReg16(PWM_VAL0,800);
```

```
            setReg16(PWM_VAL1,1600);
            setReg16(PWM_VAL2,1600);
            setReg16(PWM_VAL3,2400);
            PWMC1_Enable();
#endif


          PWMC1_OutputPadEnable();

  /* Write your code here */

  for(;;) {}
}
```

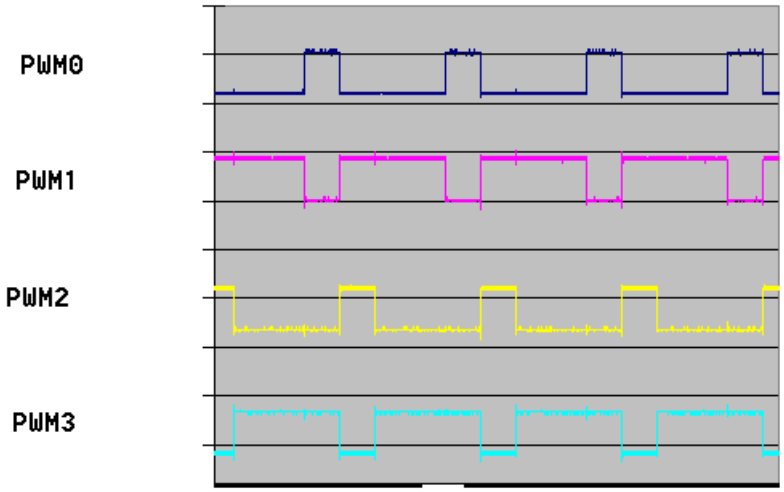

Figure 16. Asymmetric PWM output in edge-alignment mode

Figure 16 is a snapshot of the MC56F8006 PWM signals. When the MC56F8006 runs the above code by `#define PWM_EDGE_ALIGNMENT_ASYMMETRIC 1` as a compiler option, it demonstrates the PWM output waveform when the PWM is set up in an asymmetric and edge-alignment mode.

5 Conclusion

This application note introduces the new features of the MC556F8006 including the PDB, PGA, and ADC. It also introduces the multiple ADC triggering mode, for example ADC sampling at a constant cycle, the PWM synchronization signal triggering the ADC, and the external signal triggering the ADC. It also introduces the asymmetric PWM mode configuration in both center-alignment and edge-alignment mode, and the snippet code to implement the function.

Appendix A Macro Definition

Processor Expert software tools:

```
#define setRegBit(reg, bit)                                 (reg |= reg##_##bit##_##MASK)
#define clrRegBit(reg, bit)                                 (reg &= ~reg##_##bit##_##MASK)
#define getRegBit(reg, bit)                                 (reg & reg##_##bit##_##MASK)
#define setReg(reg, val)                                    (reg = (word)(val))
#define getReg(reg)                                         (reg)
#define setRegBits(reg, mask)                               (reg |= (word)(mask))
#define getRegBits(reg, mask)                               (reg & (word)(mask))
#define clrRegBits(reg, mask)                               (reg &= (word)(~(mask)))
#define setRegBitGroup(reg, bits, val)                      (reg = (word)((reg &
~reg##_##bits##_##MASK) | ((val) << reg##_##bits##_##BITNUM)))
#define getRegBitGroup(reg, bits)                           ((reg & reg##_##bits##_##MASK)
>> reg##_##bits##_##BITNUM)
#define setRegMask(reg, maskAnd, maskOr)                    (reg = (word)((getReg(reg) &
~(maskAnd)) | (maskOr)))
#define setRegBitVal(reg, bit, val)                         ((val) == 0 ? (reg &=
~reg##_##bit##_##MASK) : (reg |= reg##_##bit##_##MASK))
#define changeRegBits(reg, mask)                            (reg ^= (mask))
#define changeRegBit(reg, bit)                              (reg ^= reg##_##bit##_##MASK)
#define setReg16(RegName, val)                              (RegName = (word)(val))
```

THIS PAGE IS INTENTIONALLY BLANK

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN3844
Rev. 0
11/2009

freescale™
semiconductor