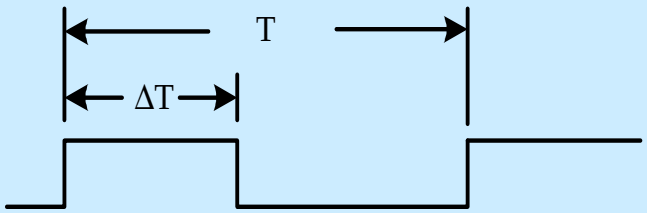
EE128
Lecture 14

Pulse Width Modulation

Instructor: Roman Chomko

Duty Cycle (1/2)



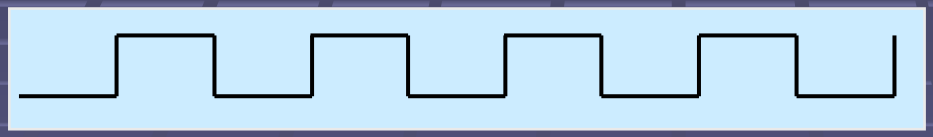
$$\text{duty cycle} = \frac{\Delta T}{T} * 100\%$$

Figure 8.13 Definition of duty cycle

duty cycle: dT/T is the percent duty cycle
duty cycle of a square wave is: 50%
duty cycle of a ground signal is: 0%
duty cycle of a "high" signal is: 100%

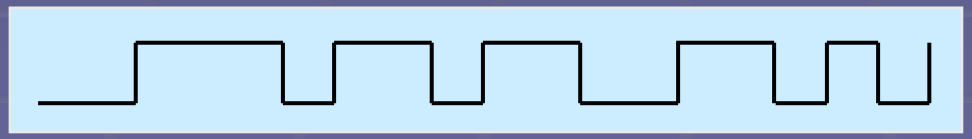
Duty Cycle (2/2)

- **Definitions:**
- **square wave: duty cycle = 50%**



- **continuous output, fixed duty cycle:**

- **continuous output, variable duty cycle:**



Output Compare Method

- Using **output compare circuitry**, we can generate square wave: pseudo-code:

- first determine how many clock cycles in a half step:

```
#define halfstep 3000
TCTL1 = 0x01;              //turn OC4 to toggle mode
while (1) {
   TC4 = TCNT + halfstep;
   TFLG1 = 0x08;           // clear OC4
   while (!(TFLG1&0x08));
}
```

by altering how long the pulse is up and pulse is down, we can dynamically control the pulse width

Motor Control with Interrupts

Motor Control: **speed is determined by the pulse width works for stepper motors, DC servo motors**

Program Structure:

```
<< init >>
main() {
```

based on tasks:
change the output compare high time; change the output compare low time;

ISR:
change pulse width based on OC high time;
change pulse width based on OC low time;
schedules next edge

this is a lot of overhead for ISR, thus separate PWM circuitry and registers have been developed

Pulse Width Modulation (PWM)

- Many applications require the generation of digital waveform.
- Output compare function can be used to generate digital waveform but incur too much overhead.
- Pulse width modulation requires only the initial setup of period and duty cycle for generating the digital waveform.

Pulse width modulation (PWM) is a scheme of expressing analog values with only TTL level pulses. This is accomplished is by changing the length of time the pulse is on in proportion to the analog level of the pulse. This is used for many purposes, such as sending analog data on a fiber optic link. The light is either on or off, so the length of the pulses varies to express different voltages

- can generate a pulse stream using the output compare circuitry or
- can use built in Pulse Width Modulation circuitry

Pulse Width Modulation (PWM)

- The MC9S12DP256 has an 8-channel PWM module.
- Each PWM channel has a period register, a duty cycle register, a control register, and a dedicated counter.
- The clock input to PWM is programmable through a two-stage circuitry.
- There are four possible clock sources for the PWM module: clock A, clock SA, clock B, and clock SB.
- Clock SA is derived by dividing the clock A by an even number ranging from 2 to 512.
- Clock SB is derived by dividing the clock B by an even number ranging from 2 to 512.
- Clock A and clock B are derived by dividing the E clock by a power of 2. The power can range from 0 to 7.

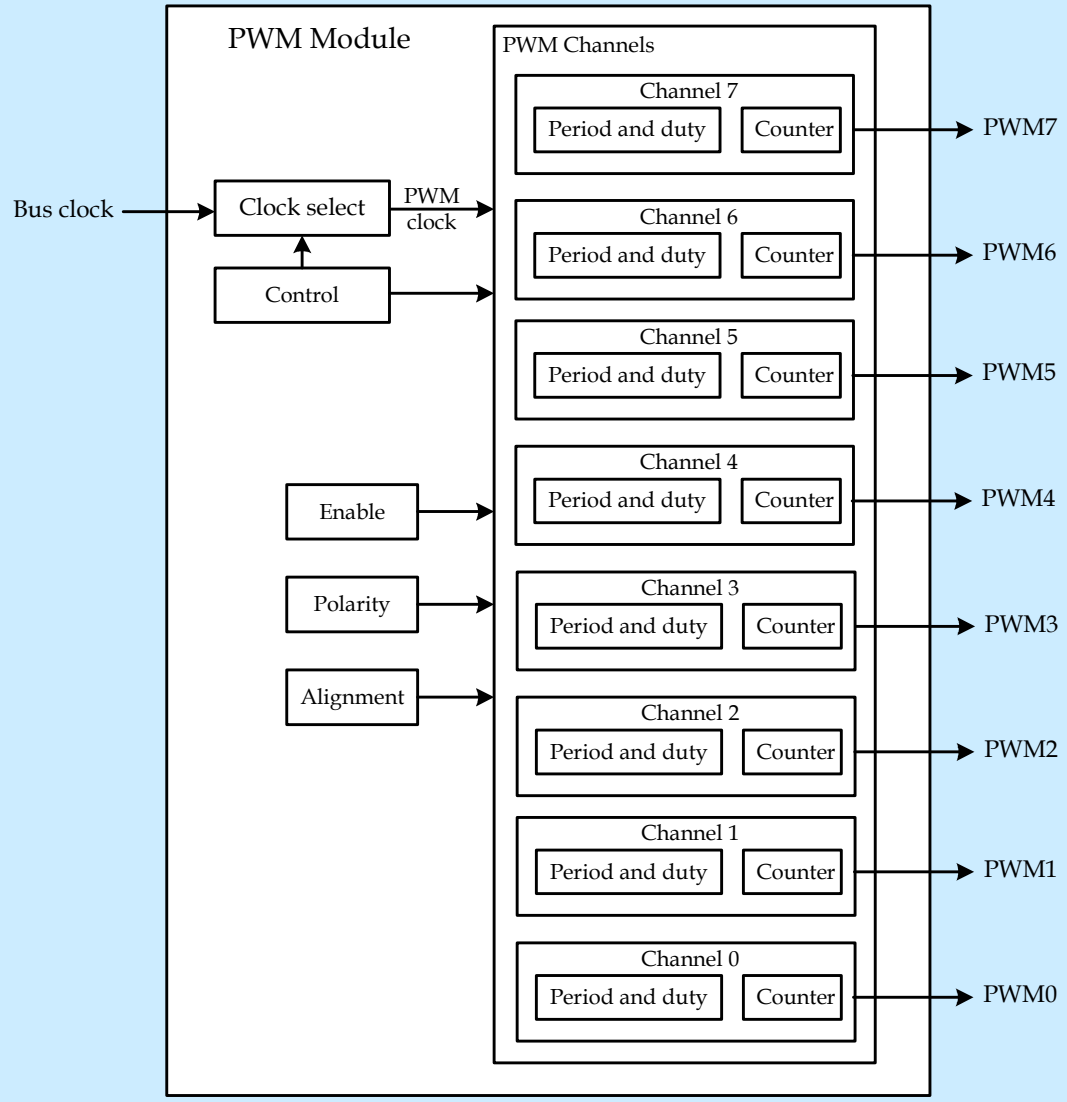


Figure 8.38 HCS12 PWM block diagram

PWM Clock Generation

- The prescale factors for clock A and clock B are determined by the PCKA2…PCKA0 and PCKB2…PCKB0 bits of the PWMPRCLK register.
- Clock SA is derived by dividing clock A by the value of the PWMSCLA register and then dividing by 2.
- Clock SB is derived by dividing clock B by the value of the PWMSCLB register and then dividing by 2.
- The clock source selection is controlled by the PWMCLK register.

	7	6	5	4	3	2	1	0
	0	PCKB2	PCKB1	PCKB0	0	PCKA2	PCKA1	PCKA0
reset:	0	0	0	0	0	0	0	0

Table 8.3 Clock B prescaler selects

PCKB2	PCKB1	PCKB0	value of clock B
0	0	0	E clock
0	0	1	E clock/2
0	1	0	E clock/4
0	1	1	E clock/8
1	0	0	E clock/16
1	0	1	E clock/32
1	1	0	E clock/64
1	1	1	E clock/128

Table 8.4 Clock A prescaler selects

PCKA2	PCKA1	PCKA0	value of clock A
0	0	0	E clock
0	0	1	E clock/2
0	1	0	E clock/4
0	1	1	E clock/8
1	0	0	E clock/16
1	0	1	E clock/32
1	1	0	E clock/64
1	1	1	E clock/128

Figure 8.41 PWM prescale clock select register (PWMPRCLK)

	7	6	5	4	3	2	1	0
	PCLK7	PCLK6	PCLK5	PCLK4	PCLK3	PCLK2	PCLK1	PCLK0
reset:	0	0	0	0	0	0	0	0

PCLKx: PWM channel x clock select (x = 7, 6, 3, 2)
0 = clock B as the clock source
1 = clock SB as the clock source
PCLKy: PWM channel y clock select (y = 5, 4, 1, 0)
0 = clock A as the clock source
1 = clock SA as the clock source

Figure 8.42 PWM clock select register (PWMCLK)

PWM Channel Timers

- The main part of each PWM channel x consists of an 8-bit counter (PWMCNTx), an 8-bit period register (PWMPERx), and an 8-bit duty cycle register (PWMDTYx).
- The waveform output period is controlled by the match between the PWMPERx register and PWMCNTx register.
- The waveform output duty cycle is controlled by the match of the PWMDTYx register and the PWMCNTx register.
- The starting polarity of the output is selectable on a per channel basis by programming the PWMPOL register.
- A PWM channel must be enabled by setting the proper bit of the PWME register.
- The overall operation of the PWM module is shown in Figure 8.44.

	7	6	5	4	3	2	1	0
	PPOL7	PPOL6	PPOL5	PPOL4	PPOL3	PPOL2	PPOL1	PPOL0
reset:	0	0	0	0	0	0	0	0

PPOLx: PWM channel x polarity
0 = PWM channel x output is low at the start of a period, then goes high when the duty count is reached.
1 = PWM channel x output is high at the start of a period, then goes low when the duty count is reached.

Figure 8.43 PWM polarity register (PWMPOL)

	7	6	5	4	3	2	1	0
	PWME7	PWME6	PWME5	PWME4	PWME3	PWME2	PWME1	PWME0
reset:	0	0	0	0	0	0	0	0

PWMEx: PWM channel x enable
0 = PWM channel x disabled.
1 = PWM channel x enabled.

Figure 8.45 PWM enable register (PWME)

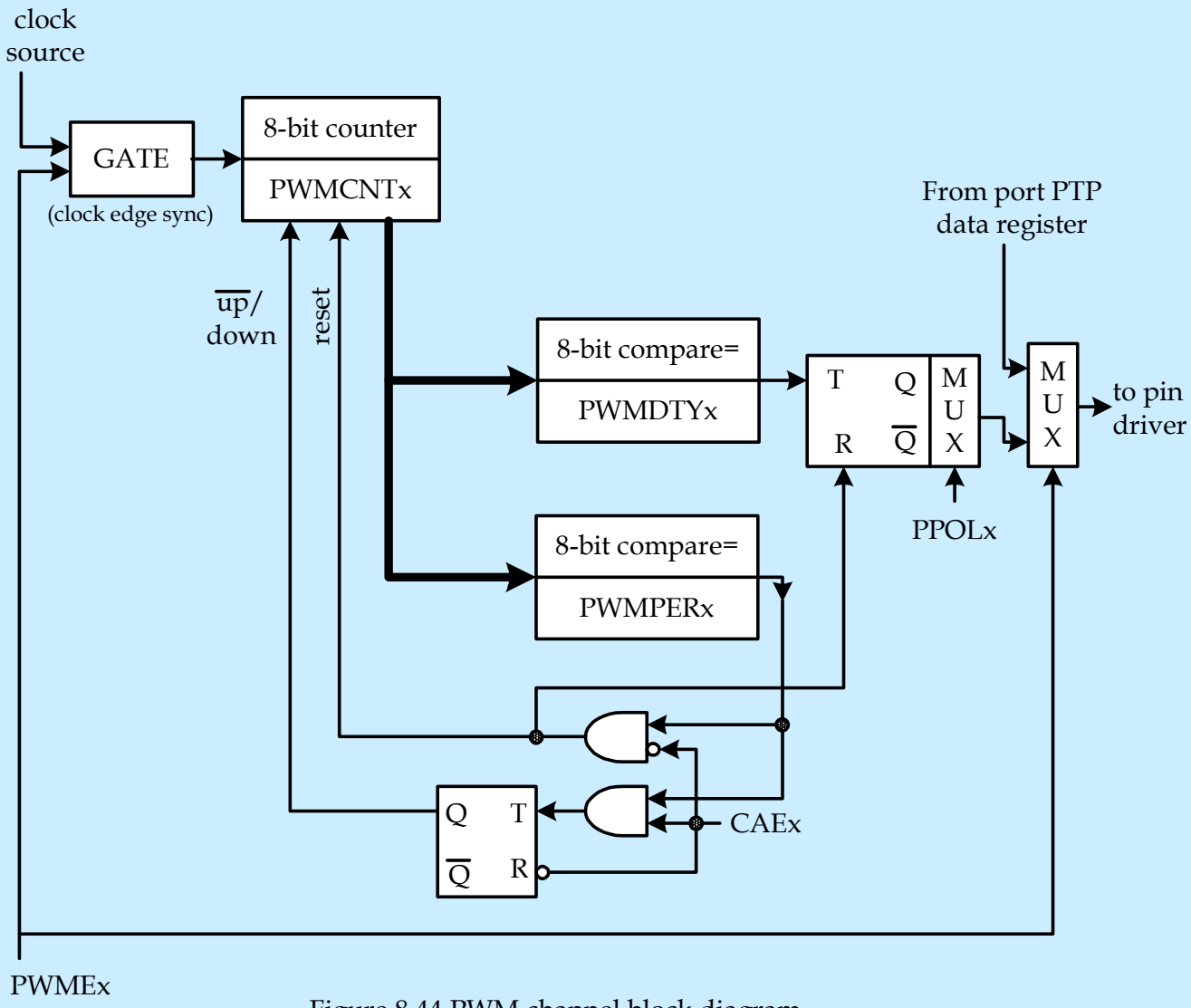


Figure 8.44 PWM channel block diagram

PWM Waveform Alignment

- PWM output waveform can be left-aligned or center-aligned.
- The choice of alignment is controlled by the PWMCAE register.

	7	6	5	4	3	2	1	0
	CAE7	CAE6	CAE5	CAE4	CAE3	CAE2	CAE1	CAE0
reset:	0	0	0	0	0	0	0	0

CAEx: Center aligned enable bit for channel x
0 = PWM channel x output is left aligned
1 = PWM channel x output is center alighed

Figure 8.46 PWM center align enable register (PWMCAE)

Left-Aligned Output

- The PWMCNTx counter is configured as a count-up counter.
  - PWMx frequency = Clock(A, B, SA, SB frequency) ÷ PWMPERx
  - Polarity = 0
  - PWMx duty cycle = [(PWMPERx – PWMDTYx) ÷ PWMPERx] × 100%
  - Polarity = 1
  - PWMx duty cycle = [PWMDTYx ÷ PWMPERx] × 100%

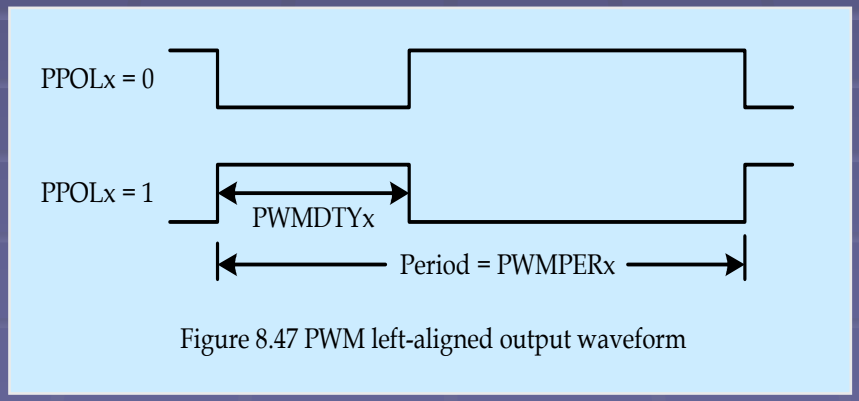


Figure 8.47 PWM left-aligned output waveform

Center-Aligned Mode

- PWM counter operates as an up/down counter and is set to count up whenever the counter is equal to $00.
- When the counter matches the duty register the output flip-flop changes state causing the PWM output to also change state.
- A match between the PWM counter and the period register changes the counter direction from an up-count to a down-count.
- When the PWM counter decrements and matches the duty register again, the output flip-flop changes state causing the PWM output to also change state.
- When the PWM counter decrements to 0, the counter direction changes from a down-count back to an up-count and the period and duty registers are reloaded from their buffers.

In Aligned Mode

PWMx frequency = Clock (A, B, SA, or SB) frequency ÷ (2 × PWMPERx)

When polarity = 0,
PWMx duty cycle = [(PWMPERx – PWMDTYx) ÷ PWMPERx] × 100%

When polarity = 1,
PWMx duty cycle = [PWMDTYx ÷ PWMPERx] × 100%

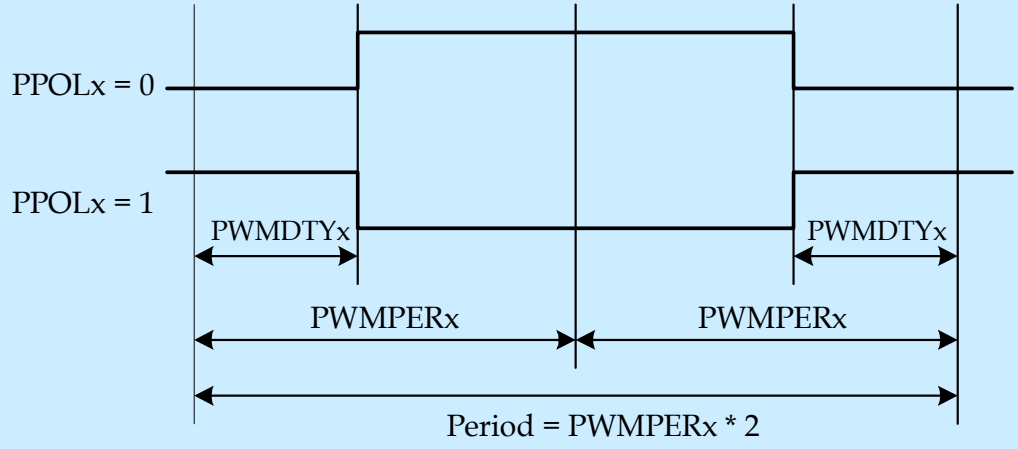


Figure 8.48 PWM center aligned output waveform

PWM 16-bit Mode (1 of 2)

- Two adjacent PWM channels can be concatenated into a 16-bit PWM channel.
- The concatenation of PWM channels are controlled by the PWMCTL register.
- The 16-bit PWM system is illustrated in Figure 8.49.
- When channel k and k+1 are concatenated, channel k is the high-order channel, whereas channel k+1 is the lower channel. (k is even number). A 16-bit channel outputs from the lower-order channel pin and is also enabled by the lower-order channel.
- Both left-aligned and center-aligned mode apply to the 16-bit mode.

PWM 16-bit Mode (2 of 2)

	7	6	5	4	3	2	1	0
	CON67	CON45	CON23	CON01	PSWAI	PFRZ	0	0
reset:	0	0	0	0	0	0	0	0

CONjk: concatenate channels j and k (j = 0, 2, 4, or 6; k = j+1)
- 0 = channel j and k are separate 8-bit PWMs
- 1 = Channels j and k are concatenated to create one 16-bit PWM channel. Channel j becomes the high order byte and channel k becomes the low order byte. Channel k output pin is used as the output for this 16-bit PWM. Channel k clock select bit determines the clock source, channel k polarity bit determines the polarity, channel k enable bit enables the output and channel k center aligned enable bit determines the output mode.

PSWAI: PWM stops in wait mode
- 0 = allow the clock to the prescaler to continue while in wait mode
- 1 = stop the input clock to the prescaler whenever the MCU is in wait mode

PFRZ: PWM counters stop in freeze mode
- 0 = allow PWM to continue while in freeze mode
- 1 = disable PWM input clock to the prescaler whenever the part is in freeze mode.

Figure 8.40 PWM control register (PWMCTL)

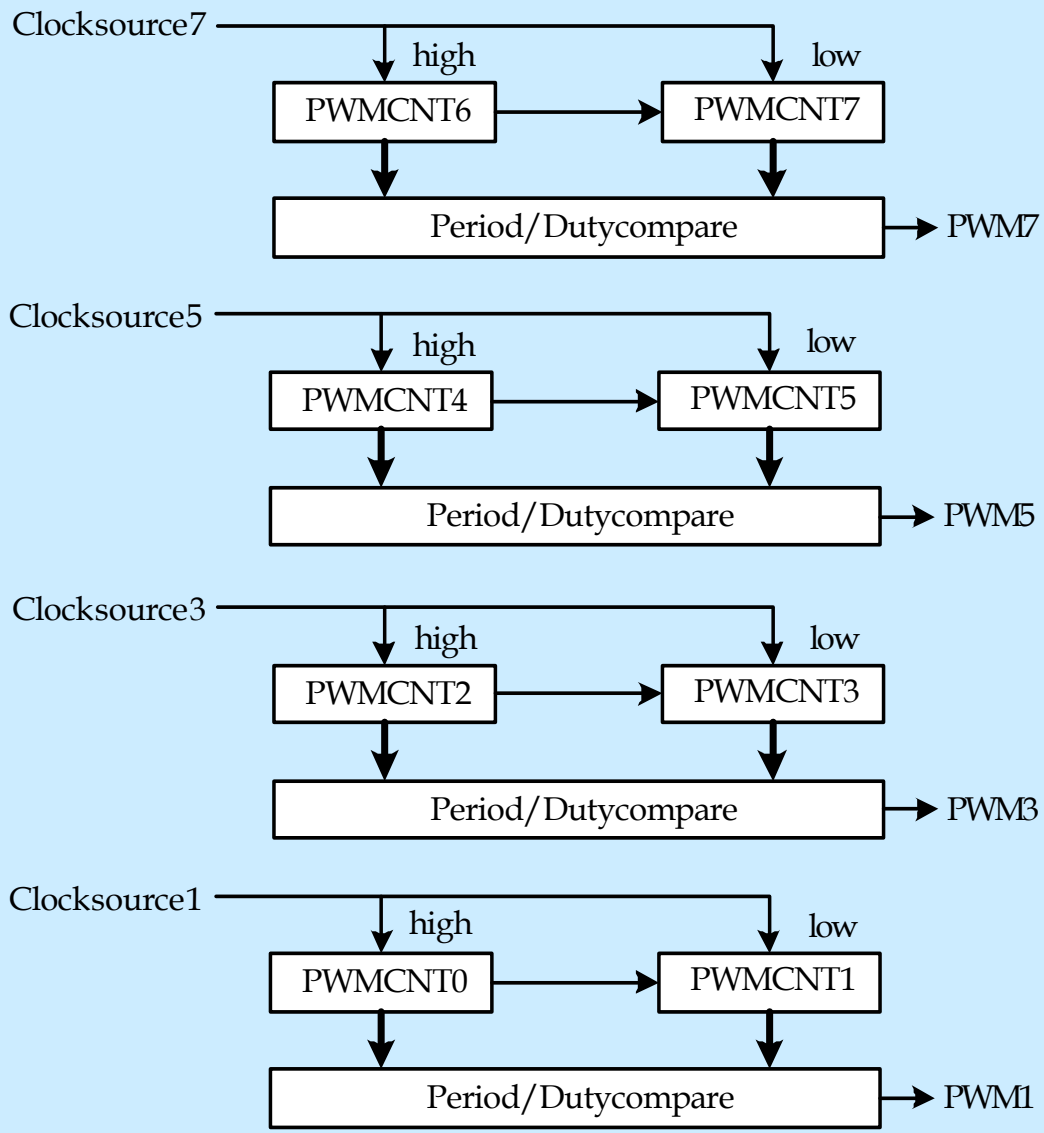


Figure8.49PWM16-bitmode

- Example 8.21 Write an instruction sequence to generate a 100KHz waveform with 50% duty cycle from the PWM0 pin (PP0). Assume that the E clock frequency is 24 MHz.

- Solution: Use the following setting:

  - Select clock A as the clock source to PWM0 and set its prescaler to 2.
  - Select left-aligned mode.
  - Load the value 120 into the PWMPER0 register (= 24000000 ÷100000 ÷2)
  - Load the value 60 into the PWMDTY0 register (= 120 × 50%)

```
#include "c:\miniide\hcs12.inc"
        …
        movb    #0,PWMCLK       ; select clock A as the clock source for PWM0
        movb    #1,PWMPRCLK     ; set clock A prescaler to 2
        movb    #1,PWMPOL       ; channel 0 output high at the start of the period
        movb    #0,PWMCAE       ; select left-aligned mode
        movb    #$0C,PWMCTL     ; 8-bit mode, stop PWM in wait and freeze mode
        movb    #120,PWMPER0    ; set period value
        movb    #60,PWMDTY0     ; set duty value
        movb    #0,PWMCNT0      ; reset the PWM0 counter
        bset    PWMEN,PWME0     ; enable PWM channel 0
```

- Example 8.22 Write an instruction sequence to generate a square wave with 20 ms period and 60% duty cycle from PWM0 and use center-aligned mode.

- Solution:

  - Select clock A as the clock source and set its prescaler to 2.
  - Load the value 120 into PWMPER0 register.
  - PWMPER0 = (20 × 24,000,000 ÷ 1,000,000) ÷ 2 ÷ 2 = 120
  - PWMDTY0 = PWMPER0 × 60% = 72.

```
movb   #0,PWMCLK       ; select clock A as the clock source
movb   #1,PWMPOL       ; set PWM0 output to start with high level
movb   #1,PWMPRCLK     ; set the PWM0 prescaler to clock A to 2
movb   #1,PWMCAE       ; select PWM0 center-aligned mode
movb   #$0C,PWMCTL     ; select 8-bit mode, stop PWM in wait mode
movb   #120,PWMPER0    ; set period value
movb   #72,PWMDTY0     ; set duty value
bset   PWME,PWME0      ; enable PWM channel 0
```

- Example 8.23 Write an instruction sequence to generate a 50 Hz digital waveform with 80% duty cycle using the 16-bit mode from the PWM1 output pin.

- Solution: Using the following setting:

  - Select clock A as the clock source and set its prescaler to 16.
  - Select left aligned mode and select polarity 1.
  - Load the value 30000 into the PWMPER0:PWMPER1 register.
  - Load the value 24000 into the PWMDTY0:PWMDTY1 register.

```
movb   #0,PWMCLK        ; select clock A as the clock source
movb   #2,PWMPOL        ; set PWM0:PWM1 output to start with high level
movb   #4,PWMPRCLK      ; set prescaler to 16
movb   #$1C,PWMCTL      ; concatenate PWM0:PWM1, stop PWM in wait mode
movb   #0,PWMCAE        ; select left align mode
movw   #30000,PWMPER0   ; set period to 30000
movw   #24000,PWMDTY0   ; set duty to 24000
bset   PWME,PWME1       ; enable PWM0:PWM1
```

- Example 8.24 Use PWM to dim the light bulb. Assume that we use the PWM0 output to control the brightness of a light bulb. Write a C program to dim the light to 10% brightness gradually in five seconds.
- The E clock frequency is 24 MHz.

- Solution:

  - Set duty cycle to 100% at the beginning.
  - Dim the brightness by 10% in the first second and then 20% per second in the following four seconds.
  - Load 100 into the PWMPER0 register at the beginning.
  - Decrement PWMPER0 by 1 every 100 ms during the first second and decrement PWMPER0 by 2 every 100 ms in the following four seconds.

```
void main ()
{
    int dim_cnt;
    PWMCLK      = 0;      /* select clock A as the clock source */
    PWMPOL      = 1;      /* make waveform to start with high level */
    PWMCTL      = 0x0C;   /* select 8-bit mode */
    PWMPRCLK    = 2;      /* set clock A prescaler to 4 */
    PWMCAE      = 0;      /* select left-aligned mode */
    PWMPER0     = 100;    /* set period of PWM0 to 0.1 ms */
    PWMDTY0     = 100;    /* set duty cycle to 100% */
    PWME        |= 0x01;  /* enable PWM0 channel */
/* reduce duty cycle 1 % per 100 ms in the first second */
    for (dim_cnt = 0; dim_cnt < 10 ; dim_cnt ++) {
                delayby100ms(1);
        PWMDTY0--;
    }
/* reduce duty cycle 2% per 100 ms in the next 4 seconds */
    for (dim_cnt = 0; dim_cnt < 40; dim_cnt ++) {
        delayby100ms(1);
        PWMDTY0 -= 2;
    }
    asm ("swi");
}
```

DC Motor Control

- The DC motor has a permanent magnetic field and its armature is a coil.
- When a voltage and a subsequent current flow are applied to the armature, the motor begins to spin.
- The voltage level applied across the armature determines the speed of rotation.
- Almost every application that uses a DC motor requires it to reverse its direction of rotation or vary its speed.
- Reversing the direction is done by changing the polarity of voltage applied to the motor.
- Changing the speed requires varying the voltage level of the input to the motor.

It deserves a separate discussion