DAC

Digital to Analog Converter (DAC)

DACs are used in audio equipment like Music Players to convert the digital data into analog audio signals.
Similarly, there are video DACs, for converting digital video data into analog video signals to be displayed on a screen.

LPC1768 contains a 10-bit DAC peripheral based on Resistor String Architecture. It can produce buffered output and the maximum update rate is 1 MHz.

The output analog voltage of this 10-bit DAC can be calculated using the following formula.

$$V_{AOUT} = \frac{\text{DACVALUE} * (\text{VREFP} - \text{VREFN})}{1024} + \text{VREFN}$$

Where,

V_{AOUT} = Output Analog Voltage of DAC [Pin P0.26]
V_{REFP} = Positive Reference Voltage of DAC
V_{REFN} = Negative Reference Voltage of DAC
DACVALUE = 10-bit digital value, which must be converted to analog voltage.

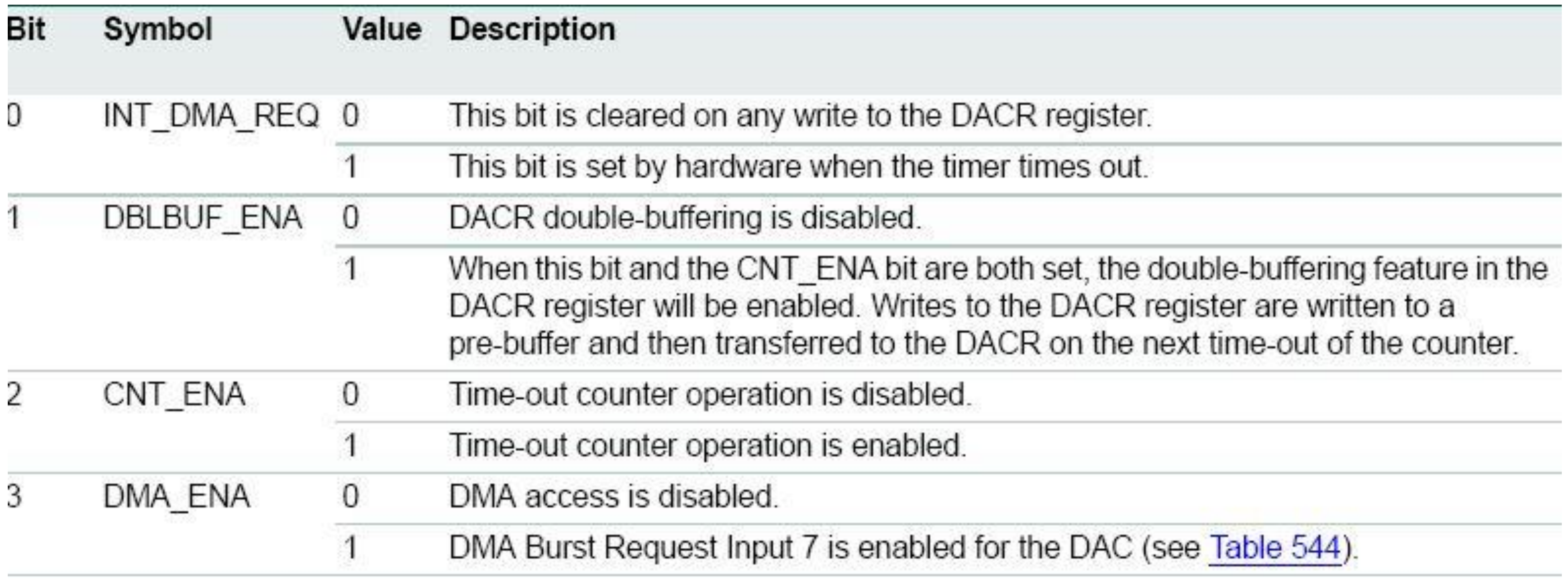
Digital to Analog Converter (DAC)

DACR – D/A Converter Register: It contains the digital value that must be converted to analog value.

Bits [15:6]	VALUE	These bits contain the digital value (DACVALUE) that is to be converted to analog value (V_{AOUT}) based on the previously mentioned formula.
Bit [16]	BIAS	When 0, settling time of DAC is 1 µS (update rate is 1 MHz), max current is 700 µA. When 1, settling time of DAC is 2.5 µS (update rate is 400 kHz), max current is 350 µA.

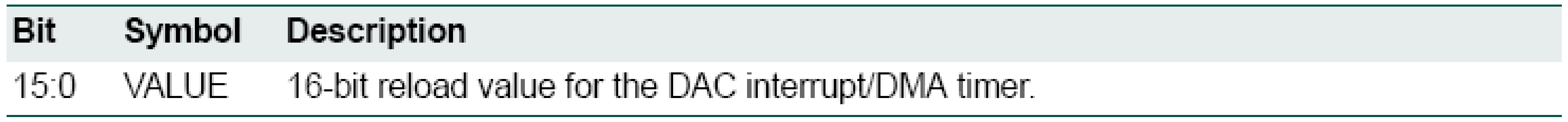
Digital to Analog Converter (DAC)

D/A Converter Control register (DACCTRL)



Bit	Symbol	Value	Description
0	INT_DMA_REQ	0	This bit is cleared on any write to the DACR register.
		1	This bit is set by hardware when the timer times out.
1	DBLBUF_ENA	0	DACR double-buffering is disabled.
		1	When this bit and the CNT_ENA bit are both set, the double-buffering feature in the DACR register will be enabled. Writes to the DACR register are written to a pre-buffer and then transferred to the DACR on the next time-out of the counter.
2	CNT_ENA	0	Time-out counter operation is disabled.
		1	Time-out counter operation is enabled.
3	DMA_ENA	0	DMA access is disabled.
		1	DMA Burst Request Input 7 is enabled for the DAC (see Table 544).

D/A Converter Counter Value register (DACCNTVAL)



Bit	Symbol	Description
15:0	VALUE	16-bit reload value for the DAC interrupt/DMA timer.

Digital to Analog Converter (DAC)

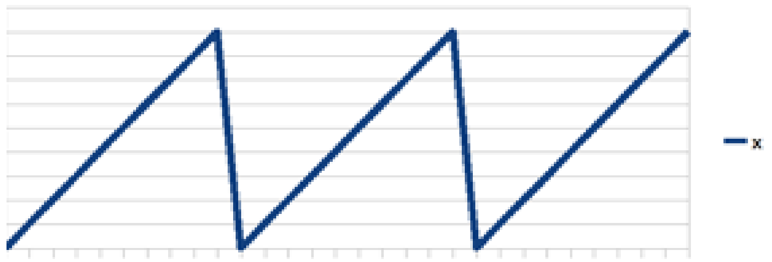
Double buffering

- Double-buffering is enabled only if both, the CNT_ENA and the DBLBUF_ENA bits are set in DACCTRL.
- In this case, any write to the DACR register will only load the pre-buffer
- The DACR will be loaded from the pre-buffer whenever the counter reaches zero
- At the same time the counter is reloaded with the COUNTVAL register value.
- Reading the DACR register will only return the contents of the DACR register itself, not the contents of the pre-buffer register.

Digital to Analog Converter (DAC)

Generate a sawtooth waveform with peak to peak amplitude 3.3v at P0.26.

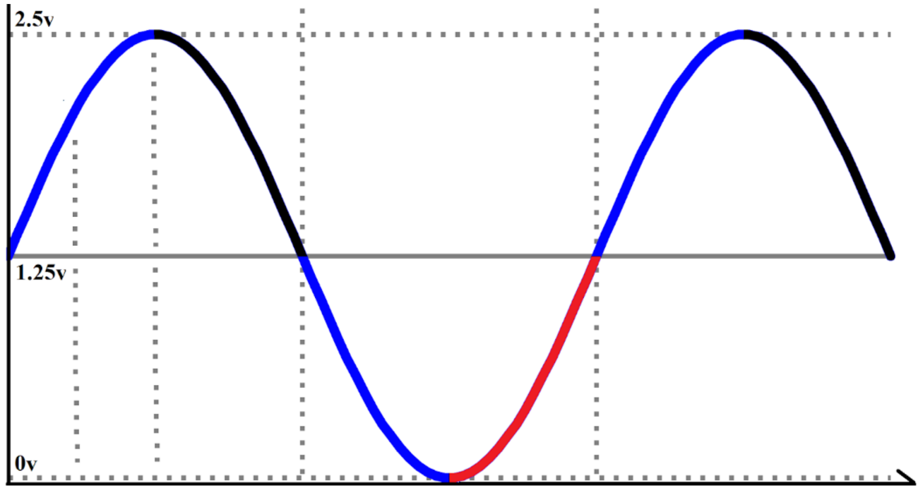
```
#include <lpc17xx.h>
 void DAC_Init(void);
 int main (void)
 {
                unsigned int m,i;
                SystemInit();
                SystemCoreClockUpdate();
                LPC_PINCON->PINSEL1 = 2<<20;// Analog Input P0.26
                LPC_DAC->DACCNTVAL = 0x0050; // DAC Counter for Double Buffering
                LPC_DAC->DACCTRL = (0x1<<1)|(0x1<<2); //Double buffering
                while ( 1 )
                {
                                LPC_DAC->DACR = (i << 6) ;
                                i++;
                                if ( i == 0x400 )                    //Maximum value is 0x3FF in 10 bit DAC
                                {
                                i = 0;
                                }
                }
 }
```



Digital to Analog Converter (DAC)

Generate a sinewave with peak to peak amplitude 2.5v at P0.26. (i.e. Vout = 1.25+1.25 sin θ)

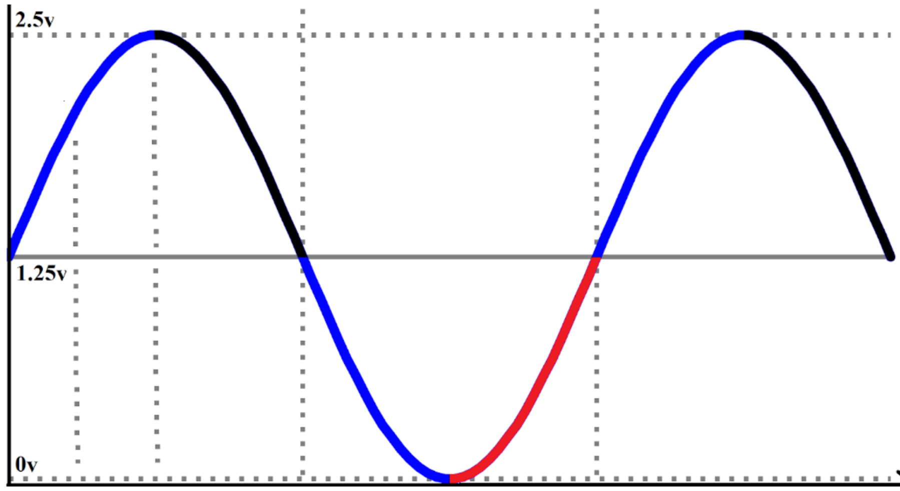
```
#include <lpc17xx.h>
 void DAC_Init(void);
sinetable[]= { 388, 582,723, 776, 723,582, 388,194,52,0,52,194};
 int main (void)
 {
                unsigned int m,i;
                SystemInit();
                SystemCoreClockUpdate();
                LPC_PINCON->PINSEL1 = 2<<20;// Analog Input P0.26
                while ( 1 )
                {
                                for(i=0; i < 12; i++) // Assuming samples separated by 30 degrees
                               {
                                LPC_DAC->DACR = (sinetable[i % 12]<< 6) ;
                                delay(); // Call timer delay based on period. If period is 10 ms. Delay is  (10ms/12)
                }
}
```



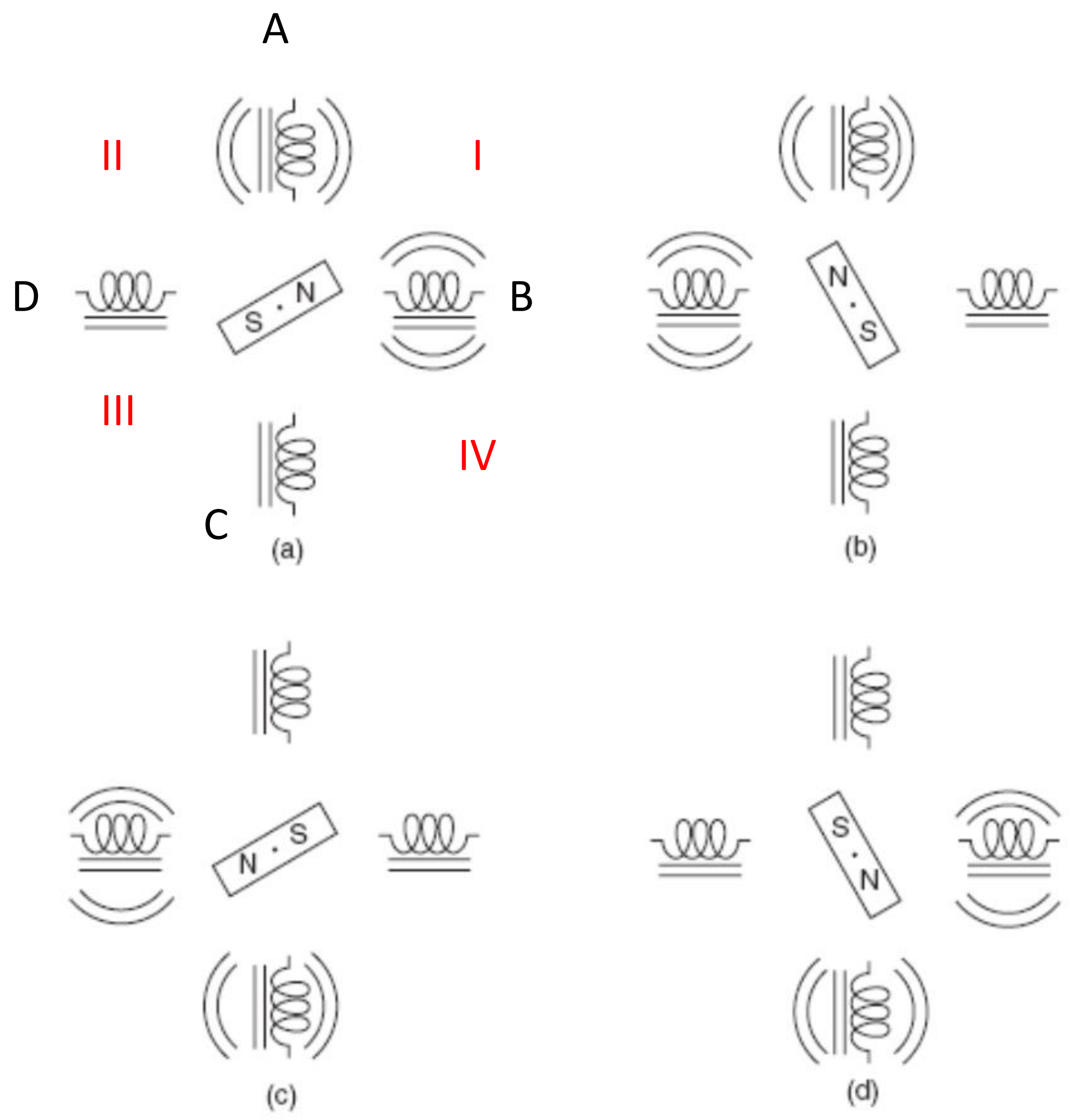
Digital to Analog Converter (DAC)

Generate a sinewave with peak to peak amplitude 2.5v at P0.26. (i.e. Vout = 1.25+1.25 sin θ)

V0 = 1.25+1.25 sin 0 =1.25; Dig value = 1024xV/3.3 =388
V30 =1.25+1.25sin 30 =1.875; Dig value =582
V60 =1.25+1.25sin 60 =2.33; Dig value =723
V90 =1.25+1.25sin 90 =2.5; Dig value =776
V120 =1.25+1.25sin 120 =2.33; Dig value =723
V150 =1.25+1.25sin 150 =1.875; Dig value =582
V180 =1.25+1.25sin 180 =1.25; Dig value =388
V210 =1.25+1.25sin 210 =0.626; Dig value =194
V240 =1.25+1.25sin 240 =0.167; Dig value =52
V270 =1.25+1.25sin 270 =0; Dig value =0
V310 =1.25+1.25sin 310 =0.167; Dig value =52
V330 =1.25+1.25sin 330 =0.626; Dig value =194



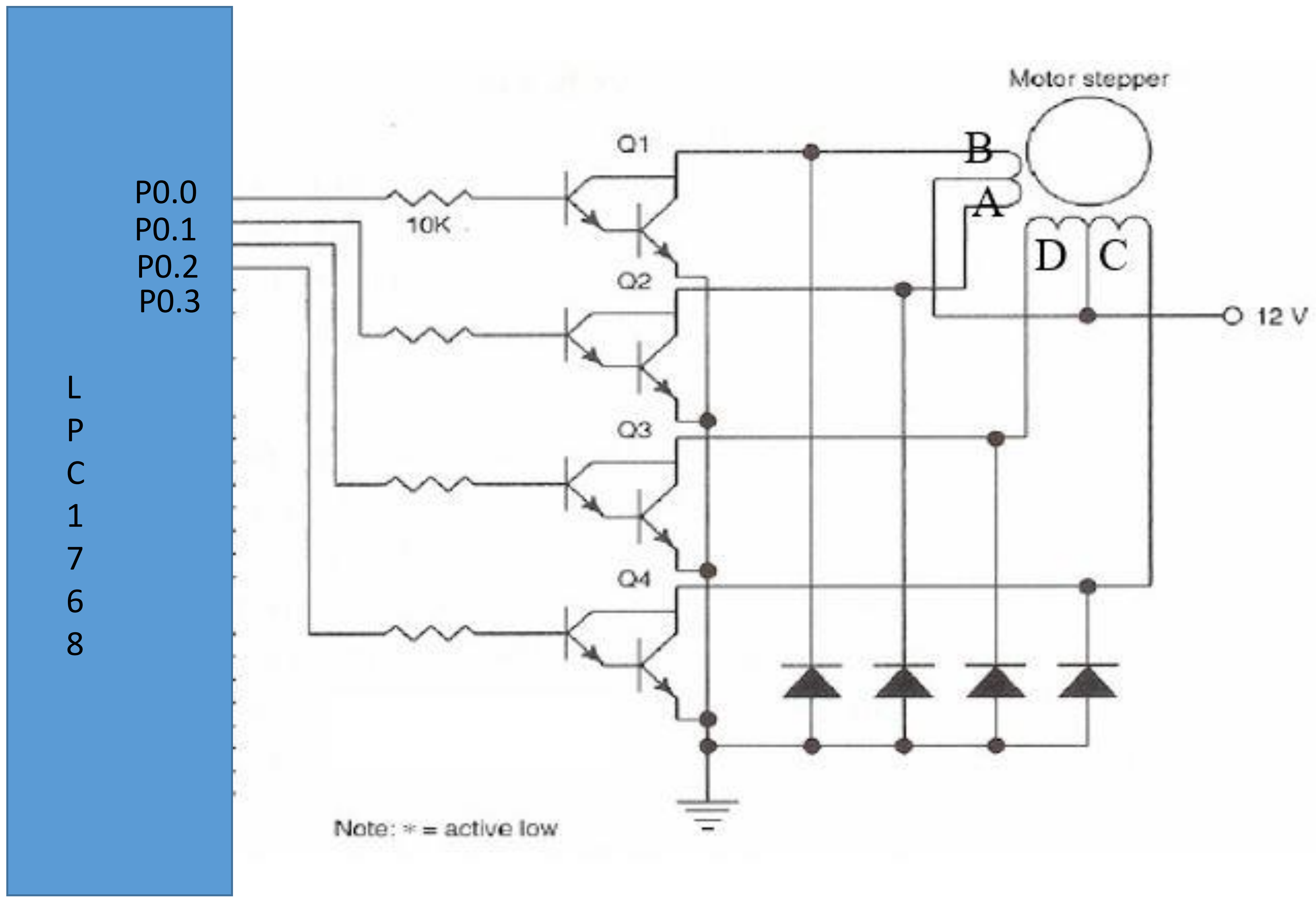
Stepper Motor Interfacing



CDAB	
0 0 1 1	3
1 0 0 1	9
1 1 0 0	C
0 1 1 0	6

For Clockwise (Forward) : 3,9,C,6
For Anticlockwise (Reverse): 3,6.C,9

Stepper Motor Interfacing



Stepper Motor Interfacing

Rotate the stepper motor 2.5 revolutions in the clockwise direction at 60 Revolutions Per Minute. Assume step angle = 6 degrees

```
#include <lpc17xx.h>
 step_pos[]= { 3,9,C,6};
 int main (void)
 {
                unsigned int m,i;
                SystemInit();
                SystemCoreClockUpdate();
                LPC_GPIO0->FIODIR = 0x0F;// Output
                while ( 1 )
                {
                                for(i=0; i < 150; i++)
                                        {
                                         LPC_GPIO0->FIOPIN = step_pos[i % 4];
                                         delay(); //Use Timer delay for 16.67 ms
                }
}
```

Time for 1 revolution @60 rpm = 1 second
Time for rotating one step =1 second/60 steps = 16.67 ms
(i.e 360/6 = 60 steps in 1 revolution)
Number of steps needed = 2.5 x 60 = 150

Pulse Width Modulation (PWM)

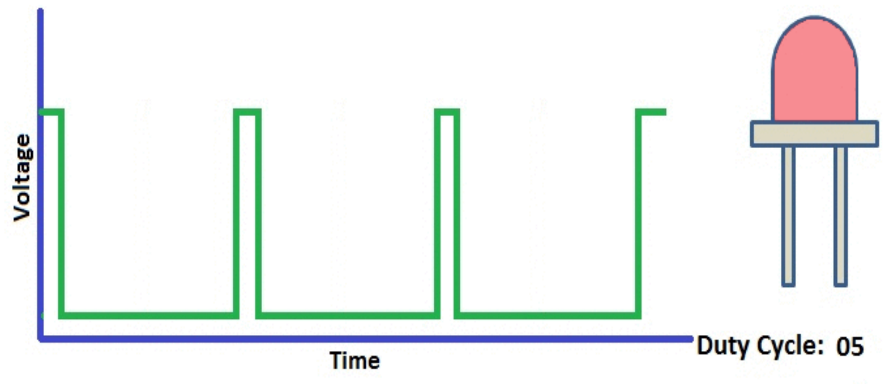
PWM is one of the commonly used techniques to control the amount of power delivered through a pin. The PWM technique allows you to control the brightness of an LED, speed of a Motor, position of a Servo etc.

The width of the Pulse i.e. the duration for with the pulse stays ON in a period is varied. Hence, it is called Pulse Width Modulation.

The Period of a PWM Cycle is the sum of duration for which the Pulse is HIGH and the duration for which the Pulse in LOW. This is usually represented by T_{ON} and T_{OFF}. So, Period of PWM = $T_{ON} + T_{OFF}$.

An important parameter of a PWM Signal is its Duty cycle. Duty cycle is the ratio of duration for which Pulse is HIGH to the total period of the PWM Signal.

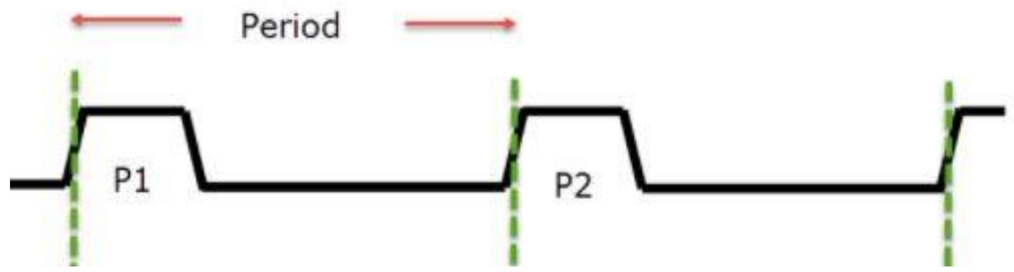
Duty cycle = $T_{ON} / (T_{ON} + T_{OFF})$
Duty cycle can also be represented as percentage.
Duty cycle % = $T_{ON} * 100 / (T_{ON} + T_{OFF})$



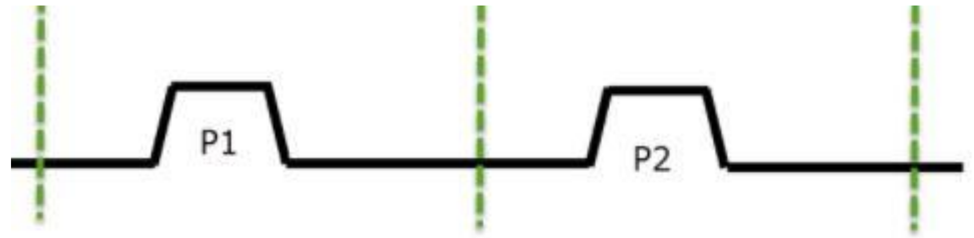
Pulse Width Modulation (PWM)

Two types of PWM
:
•Single Edge PWM
•Double Edge PWM

In **Single Edge PWM**, the Pulse starts at the beginning of the period



In **Double Edge PWM**, both the edges can be modulated and hence, the pulse is placed anywhere in the period. Double Edge PWM is generally used in multi-phase motor control applications.



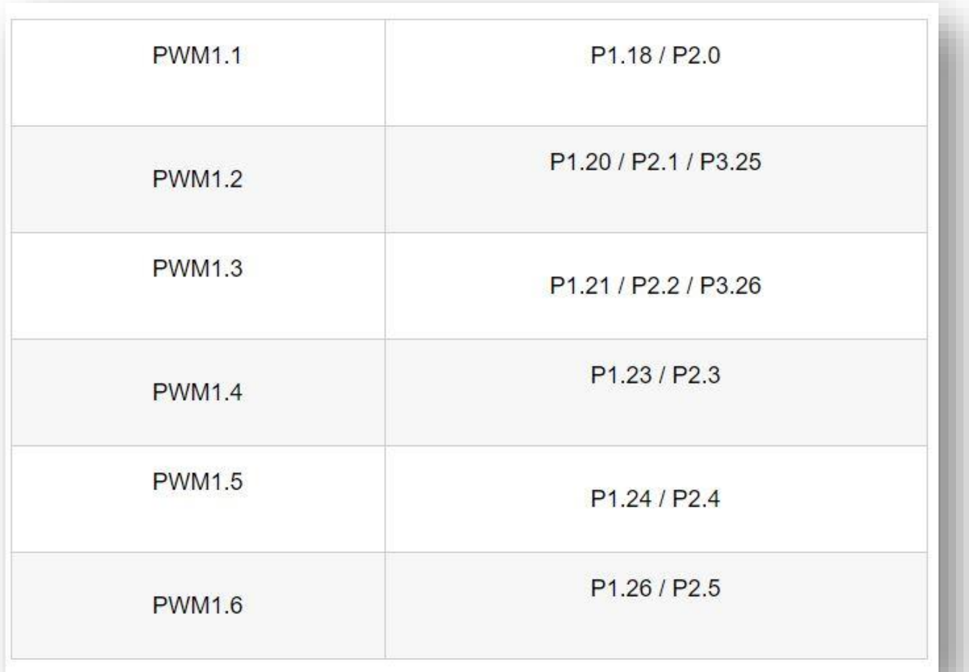
Pulse Width Modulation (PWM)

Duty cycle of the PWM determines the average power of a PWM Signal and it is calculated using the following formula.

V_{AVG} **= Duty Cycle *** V_H

Where V_H is the maximum voltage level of the PWM Signal.

In LPC1768, there are 6 PWM outputs called PWM1.1, PWM1.2, …PWM1.6



PWM1.1	P1.18 / P2.0
PWM1.2	P1.20 / P2.1 / P3.25
PWM1.3	P1.21 / P2.2 / P3.26
PWM1.4	P1.23 / P2.3
PWM1.5	P1.24 / P2.4
PWM1.6	P1.26 / P2.5

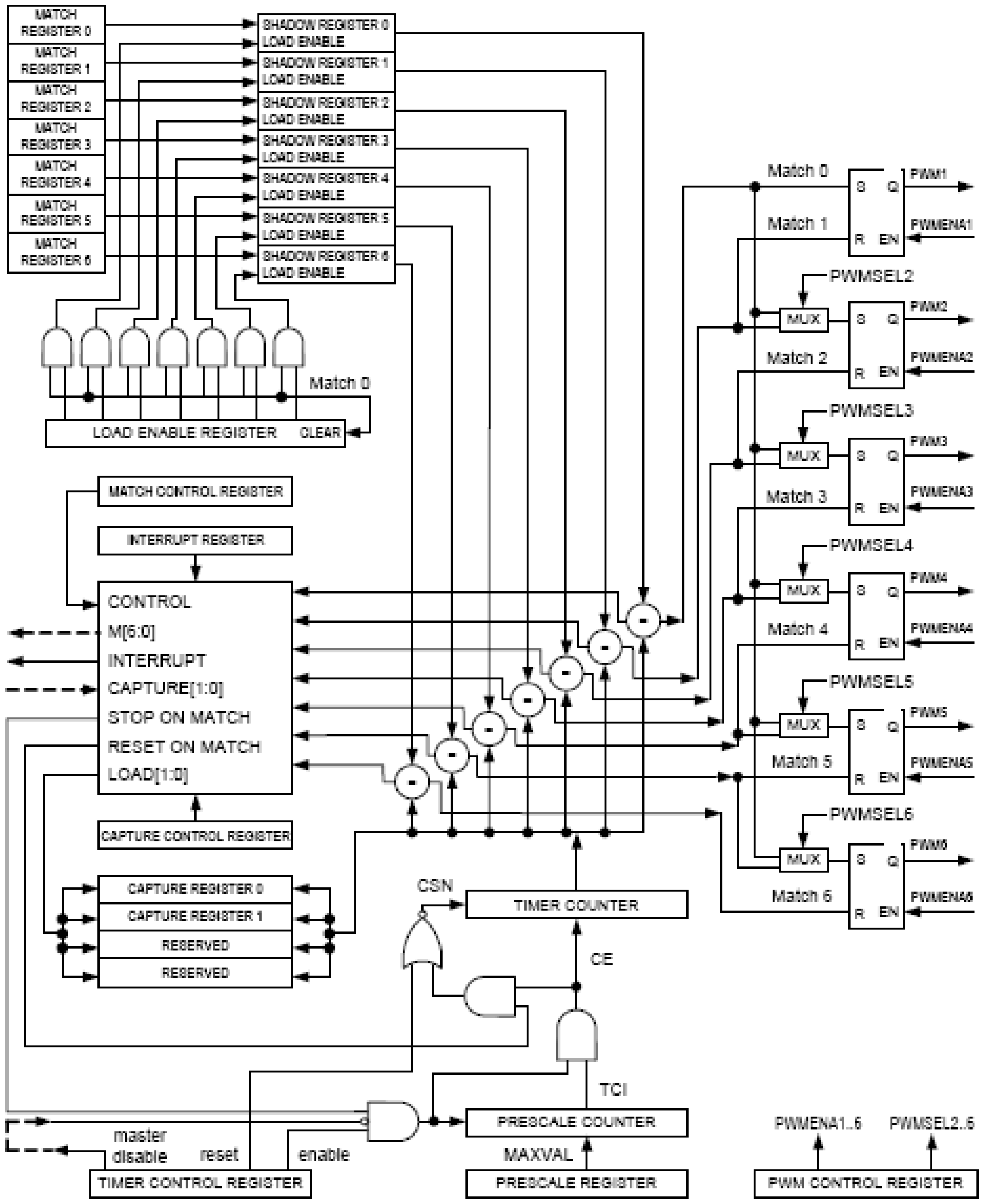
Pulse Width Modulation (PWM)

The PWM is based on the standard Timer block and inherits all of its features, although only the PWM function is pinned out.

The Timer is designed to count cycles of the peripheral clock (PCLK) and optionally generate interrupts or perform other actions when specified timer values occur, based on seven match registers.

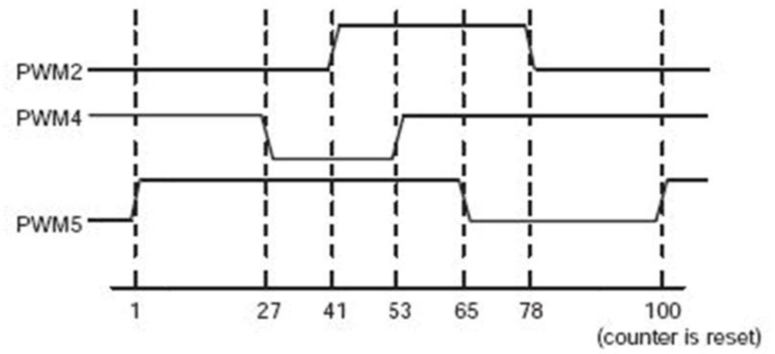
The PWM function is in addition to these features, and is based on match register events.

Pulse Width Modulation (PWM)



Two match registers can be used to provide a single edge controlled PWM output. One match register (PWMMR0) controls the PWM cycle rate, by resetting the count upon match. The other match register controls the PWM edge position.

Three match registers can be used to provide a PWM output with both edges controlled. Again, the PWMMR0 match register controls the PWM cycle rate. The other match registers control the two PWM edge positions.



The waveforms show one PWM cycle and demonstrate PWM outputs under the following conditions:

The timer is configured for PWM mode (counter resets to 1).

Match 0 is configured to reset the timer/counter when a match event occurs.

Control bits PWMSEL2 and PWMSEL4 are set.

The Match register values are as follows:

MR0 = 100 (PWM rate)

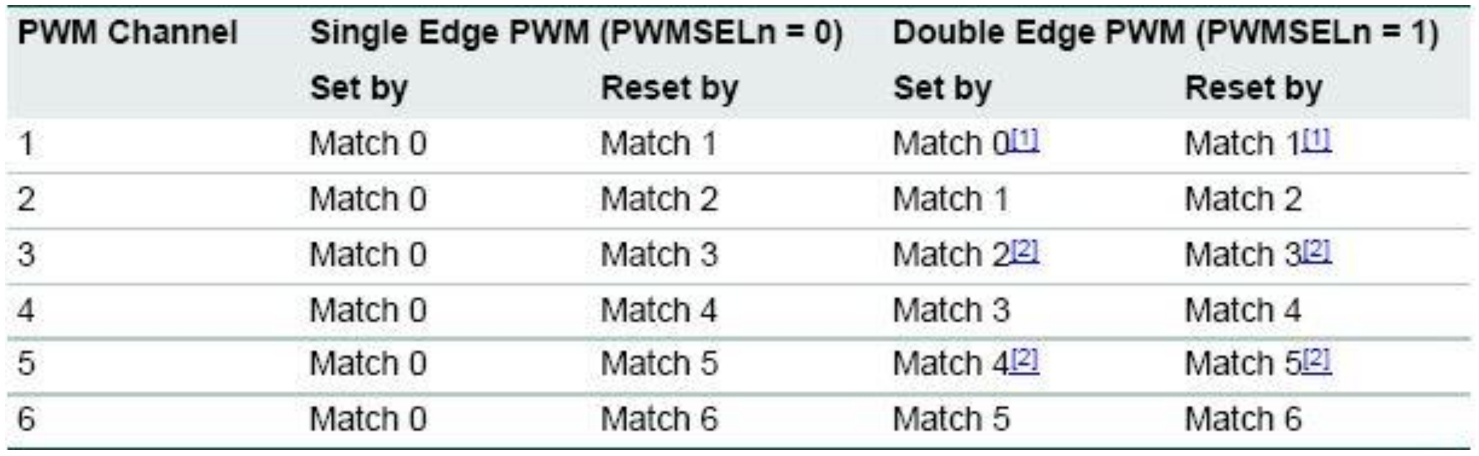
MR1 = 41, MR2 = 78 (PWM2 output)

MR3 = 53, MR4 = 27 (PWM4 output)

MR5 = 65 (PWM5 output)

Pulse Width Modulation (PWM)

Set and reset inputs for PWM Flip-Flops



PWM Channel	Single Edge PWM (PWMSELn = 0)		Double Edge PWM (PWMSELn = 1)	
	Set by	**Reset by**	**Set by**	**Reset by**
1	Match 0	Match 1	Match 0[1]	Match 1[1]
2	Match 0	Match 2	Match 1	Match 2
3	Match 0	Match 3	Match 2[2]	Match 3[2]
4	Match 0	Match 4	Match 3	Match 4
5	Match 0	Match 5	Match 4[2]	Match 5[2]
6	Match 0	Match 6	Match 5	Match 6

Pulse Width Modulation (PWM)

PWM Timer Control Register (PWM1TCR)

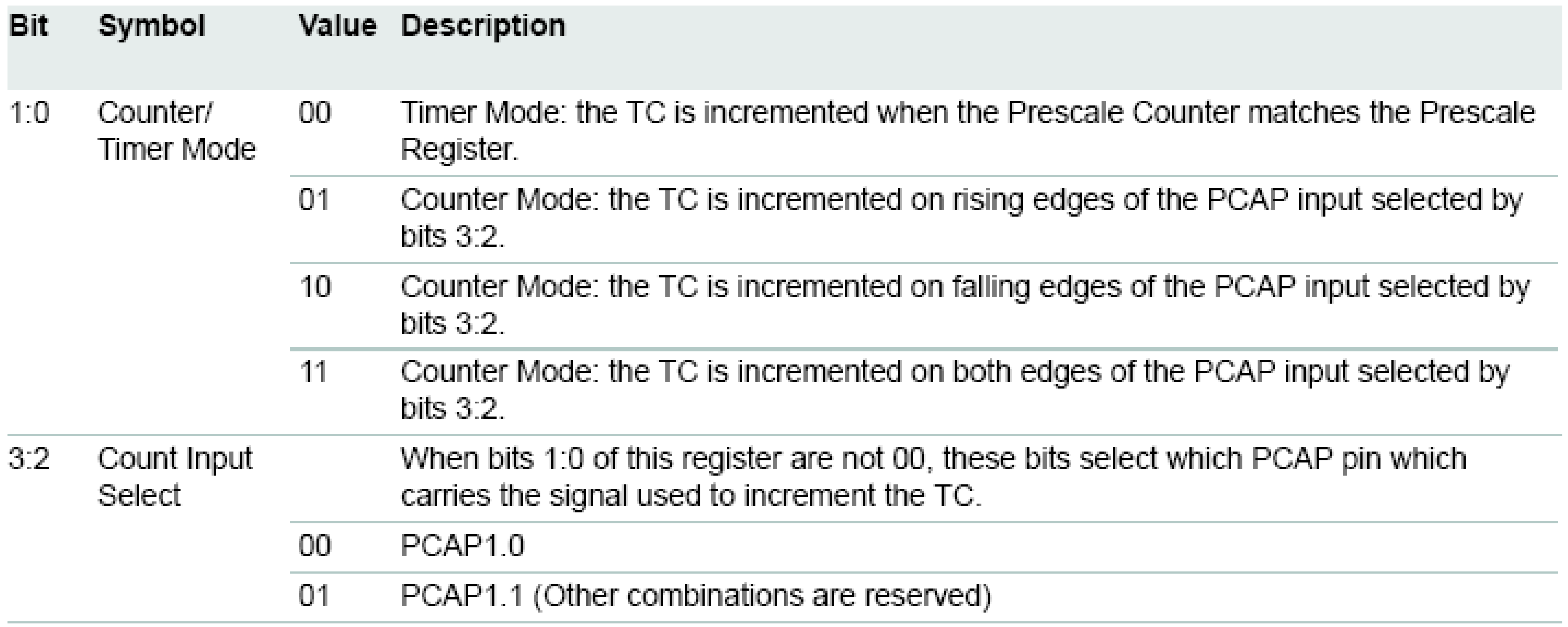
The PWM Timer Control Register (PWMTCR) is used to control the operation of the PWM Timer Counter.

Bit 0	Counter Enable	When 1, PWM Timer Counter and Prescale Counter are enabled.
Bit 1	Counter Reset	When 1, PWM Timer Counter and Prescale Counter are reset on next positive edge of PCLK.
Bit 3	PWM Enable	When 1, PWM Mode is enabled. TC will reset to 1.

Pulse Width Modulation (PWM)

PWM Count Control Register (PWM1CTCR)

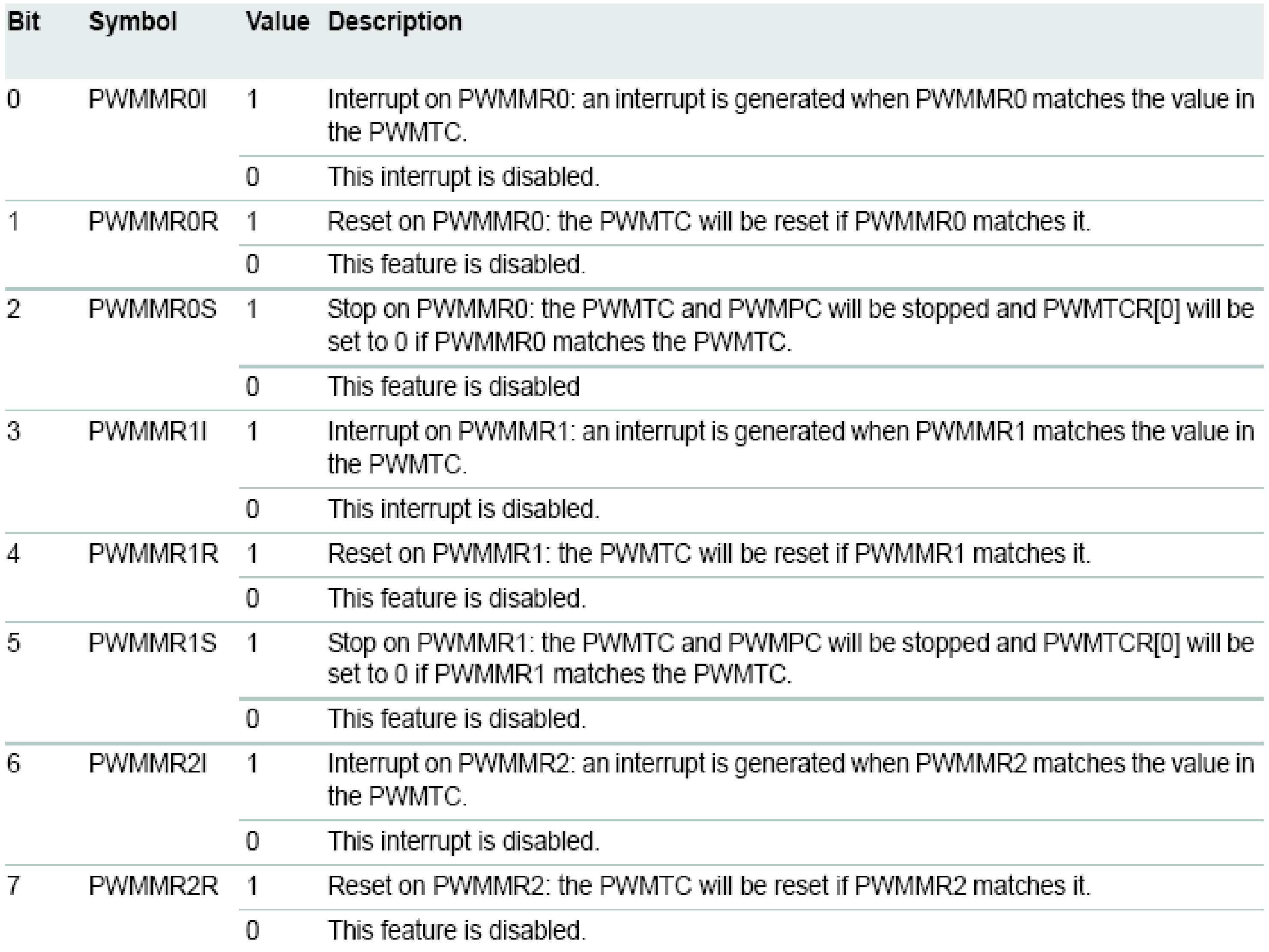
The Count Control Register (CTCR) is used to select between Timer and Counter mode



Bit	Symbol	Value	Description
1:0	Counter/ Timer Mode	00	Timer Mode: the TC is incremented when the Prescale Counter matches the Prescale Register.
		01	Counter Mode: the TC is incremented on rising edges of the PCAP input selected by bits 3:2.
		10	Counter Mode: the TC is incremented on falling edges of the PCAP input selected by bits 3:2.
		11	Counter Mode: the TC is incremented on both edges of the PCAP input selected by bits 3:2.
3:2	Count Input Select		When bits 1:0 of this register are not 00, these bits select which PCAP pin which carries the signal used to increment the TC.
		00	PCAP1.0
		01	PCAP1.1 (Other combinations are reserved)

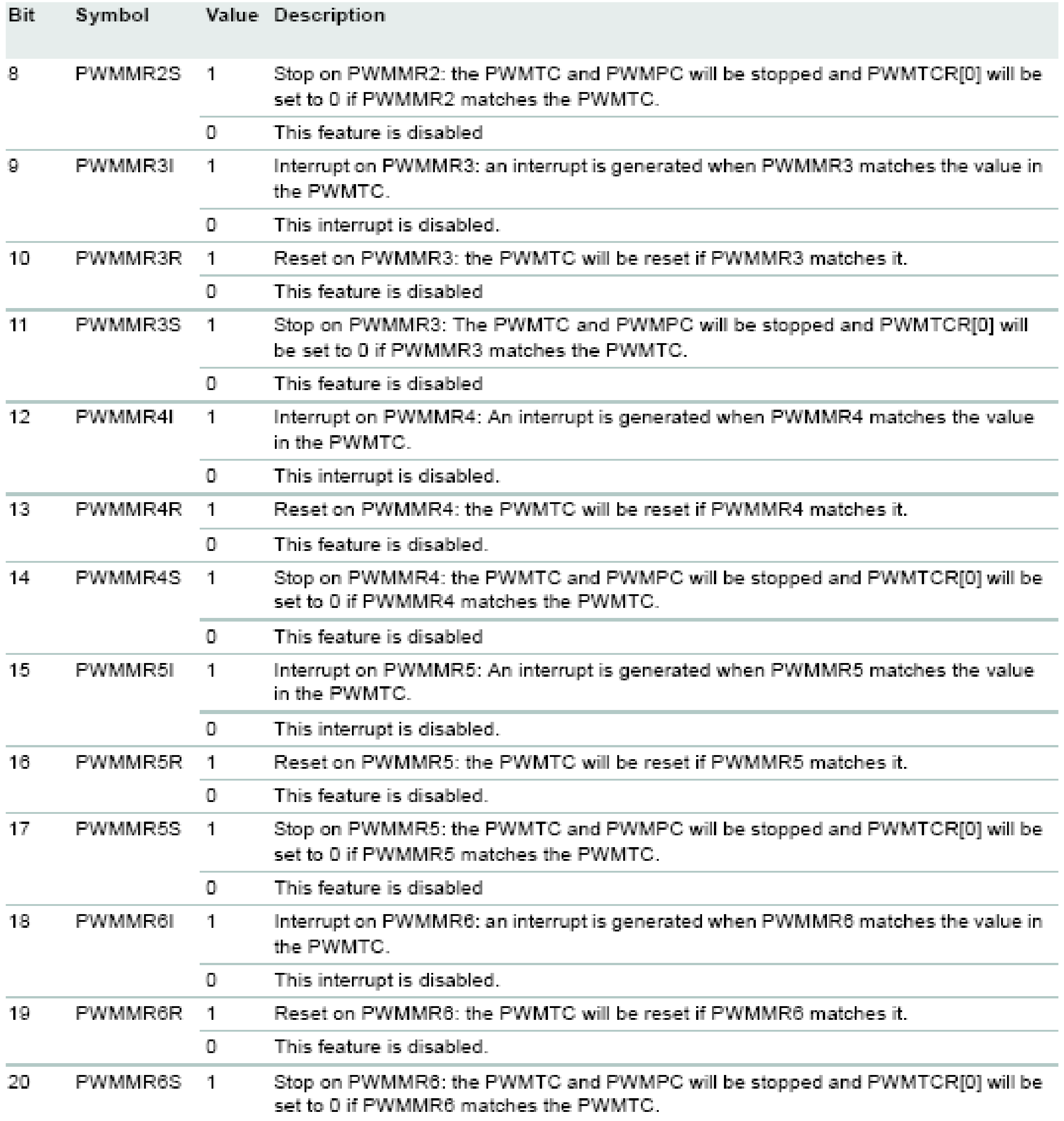
Pulse Width Modulation (PWM)

PWM Match Control Register (PWM1MCR)



Bit	Symbol	Value	Description
0	PWMMR0I	1	Interrupt on PWMMR0: an interrupt is generated when PWMMR0 matches the value in the PWMTC.
		0	This interrupt is disabled.
1	PWMMR0R	1	Reset on PWMMR0: the PWMTC will be reset if PWMMR0 matches it.
		0	This feature is disabled.
2	PWMMR0S	1	Stop on PWMMR0: the PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR0 matches the PWMTC.
		0	This feature is disabled
3	PWMMR1I	1	Interrupt on PWMMR1: an interrupt is generated when PWMMR1 matches the value in the PWMTC.
		0	This interrupt is disabled.
4	PWMMR1R	1	Reset on PWMMR1: the PWMTC will be reset if PWMMR1 matches it.
		0	This feature is disabled.
5	PWMMR1S	1	Stop on PWMMR1: the PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR1 matches the PWMTC.
		0	This feature is disabled.
6	PWMMR2I	1	Interrupt on PWMMR2: an interrupt is generated when PWMMR2 matches the value in the PWMTC.
		0	This interrupt is disabled.
7	PWMMR2R	1	Reset on PWMMR2: the PWMTC will be reset if PWMMR2 matches it.
		0	This feature is disabled.

Pulse Width Modulation (PWM)

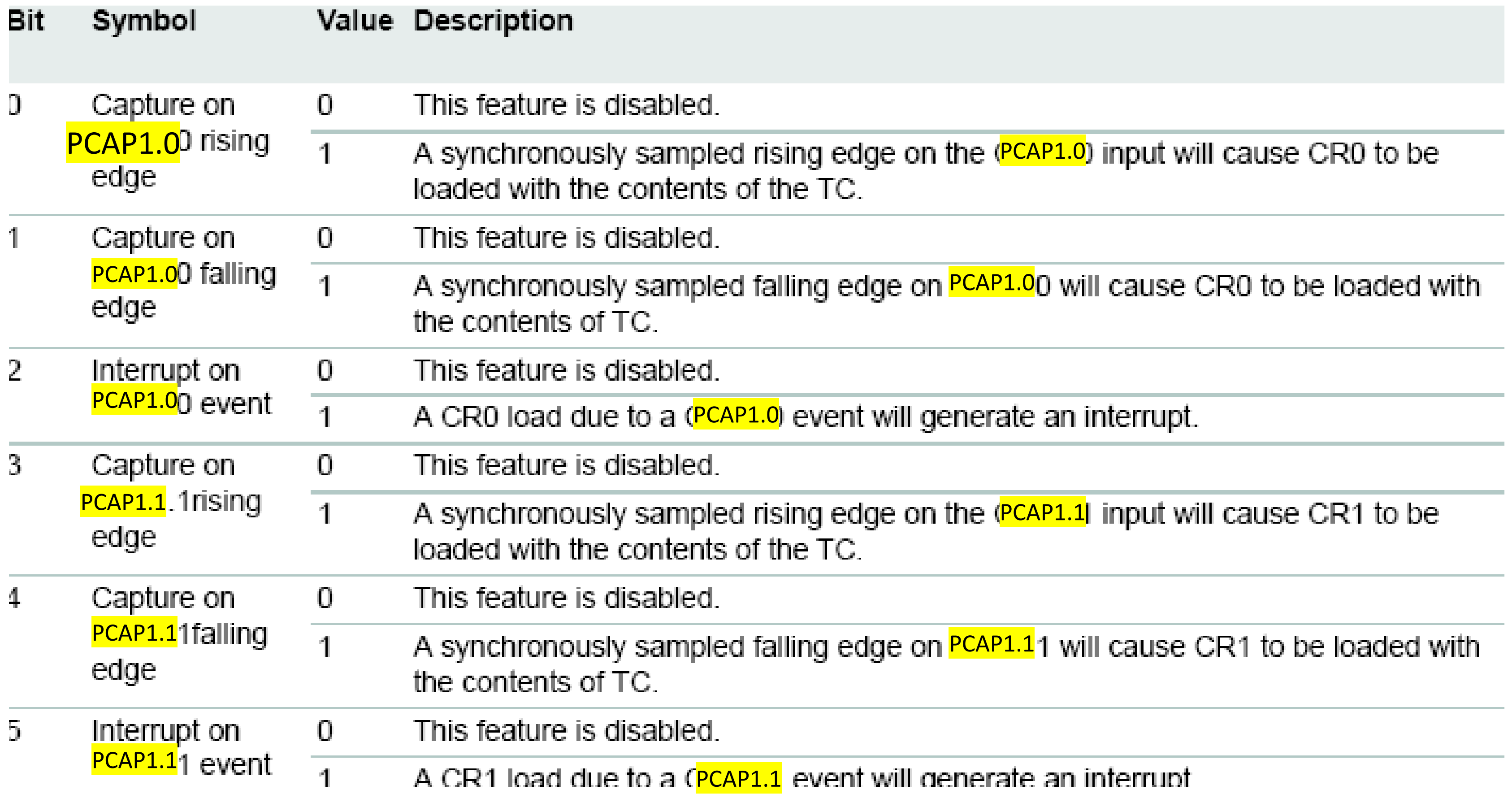


Bit	Symbol	Value	Description
8	PWMMR2S	1	Stop on PWMMR2: the PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR2 matches the PWMTC.
		0	This feature is disabled
9	PWMMR3I	1	Interrupt on PWMMR3: an interrupt is generated when PWMMR3 matches the value in the PWMTC.
		0	This interrupt is disabled.
10	PWMMR3R	1	Reset on PWMMR3: the PWMTC will be reset if PWMMR3 matches it.
		0	This feature is disabled
11	PWMMR3S	1	Stop on PWMMR3: The PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR3 matches the PWMTC.
		0	This feature is disabled
12	PWMMR4I	1	Interrupt on PWMMR4: An interrupt is generated when PWMMR4 matches the value in the PWMTC.
		0	This interrupt is disabled.
13	PWMMR4R	1	Reset on PWMMR4: the PWMTC will be reset if PWMMR4 matches it.
		0	This feature is disabled.
14	PWMMR4S	1	Stop on PWMMR4: the PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR4 matches the PWMTC.
		0	This feature is disabled
15	PWMMR5I	1	Interrupt on PWMMR5: An interrupt is generated when PWMMR5 matches the value in the PWMTC.
		0	This interrupt is disabled.
16	PWMMR5R	1	Reset on PWMMR5: the PWMTC will be reset if PWMMR5 matches it.
		0	This feature is disabled.
17	PWMMR5S	1	Stop on PWMMR5: the PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR5 matches the PWMTC.
		0	This feature is disabled
18	PWMMR6I	1	Interrupt on PWMMR6: an interrupt is generated when PWMMR6 matches the value in the PWMTC.
		0	This interrupt is disabled.
19	PWMMR6R	1	Reset on PWMMR6: the PWMTC will be reset if PWMMR6 matches it.
		0	This feature is disabled.
20	PWMMR6S	1	Stop on PWMMR6: the PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR6 matches the PWMTC.

Pulse Width Modulation (PWM)

PWM Capture Control Register (PWM1CCR)

The Capture Control Register is used to control whether one of the two Capture Registers is loaded with the value in the Timer Counter when a capture event occurs, and whether an interrupt is generated by the capture event. Setting both the rising and falling bits at the same time is a valid configuration, resulting in a capture event for both edges.

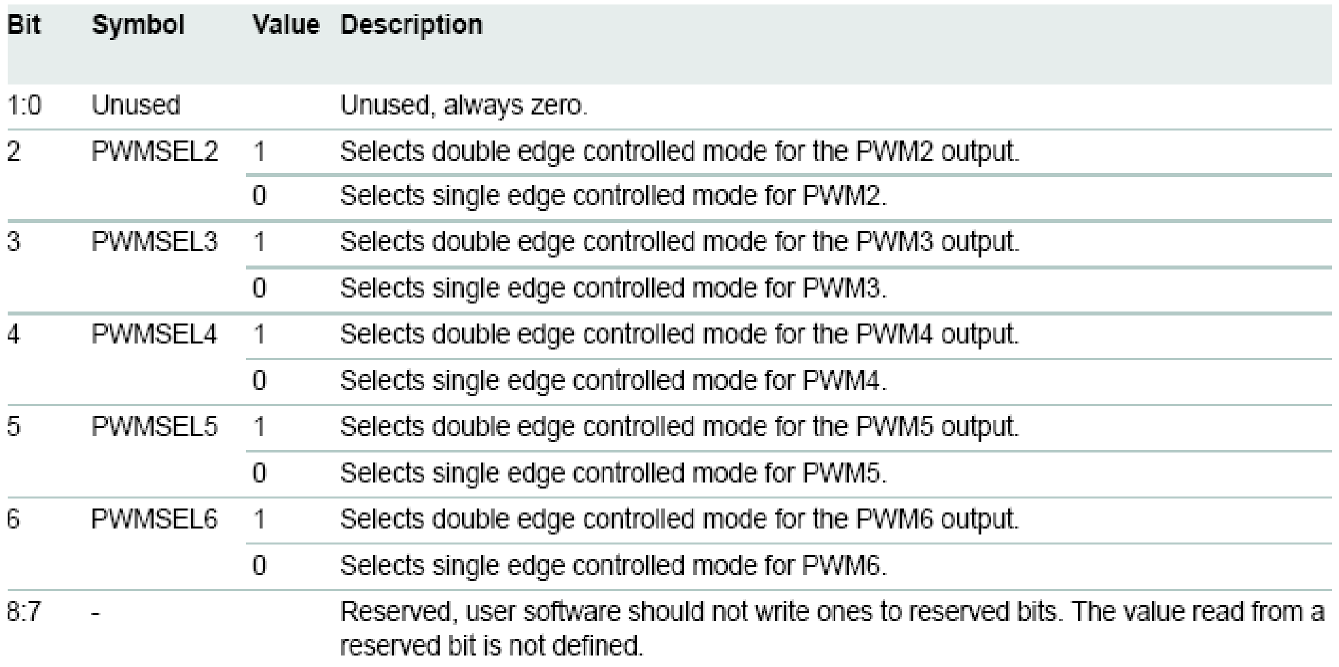


Bit	Symbol	Value	Description
0	Capture on PCAP1.00 rising edge	0	This feature is disabled.
		1	A synchronously sampled rising edge on the (PCAP1.0) input will cause CR0 to be loaded with the contents of the TC.
1	Capture on PCAP1.00 falling edge	0	This feature is disabled.
		1	A synchronously sampled falling edge on PCAP1.00 will cause CR0 to be loaded with the contents of TC.
2	Interrupt on PCAP1.00 event	0	This feature is disabled.
		1	A CR0 load due to a (PCAP1.0) event will generate an interrupt.
3	Capture on PCAP1.1.1rising edge	0	This feature is disabled.
		1	A synchronously sampled rising edge on the (PCAP1.1) input will cause CR1 to be loaded with the contents of the TC.
4	Capture on PCAP1.11falling edge	0	This feature is disabled.
		1	A synchronously sampled falling edge on PCAP1.11 will cause CR1 to be loaded with the contents of TC.
5	Interrupt on PCAP1.11 event	0	This feature is disabled.
		1	A CR1 load due to a (PCAP1.1 event will generate an interrupt

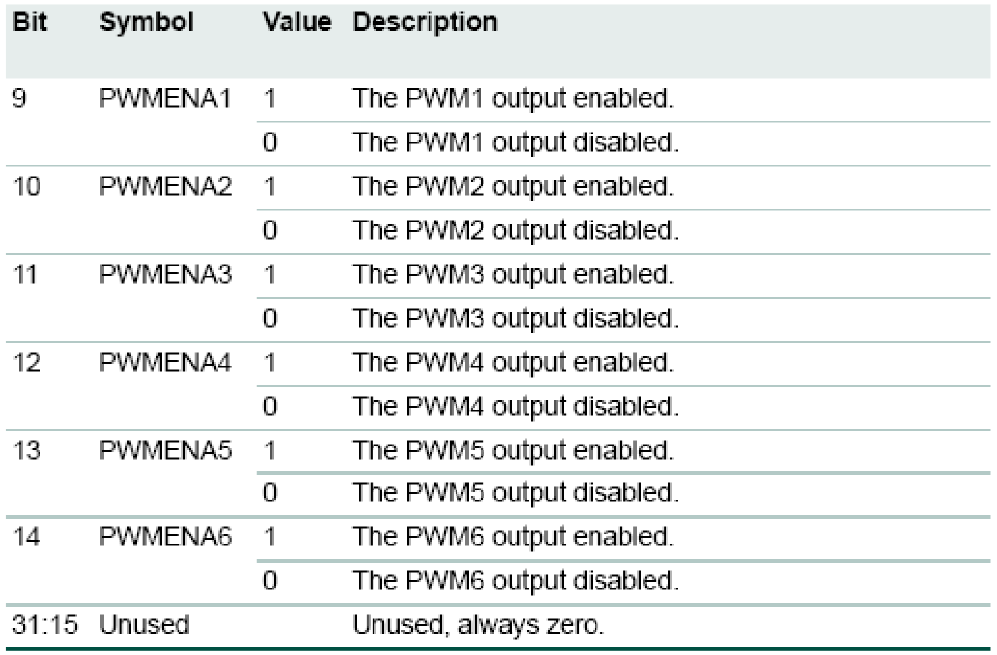
Pulse Width Modulation (PWM)

PWM Control Register (PWM1PCR

The PWM Control Register is used to enable and select the type of each PWM channel.



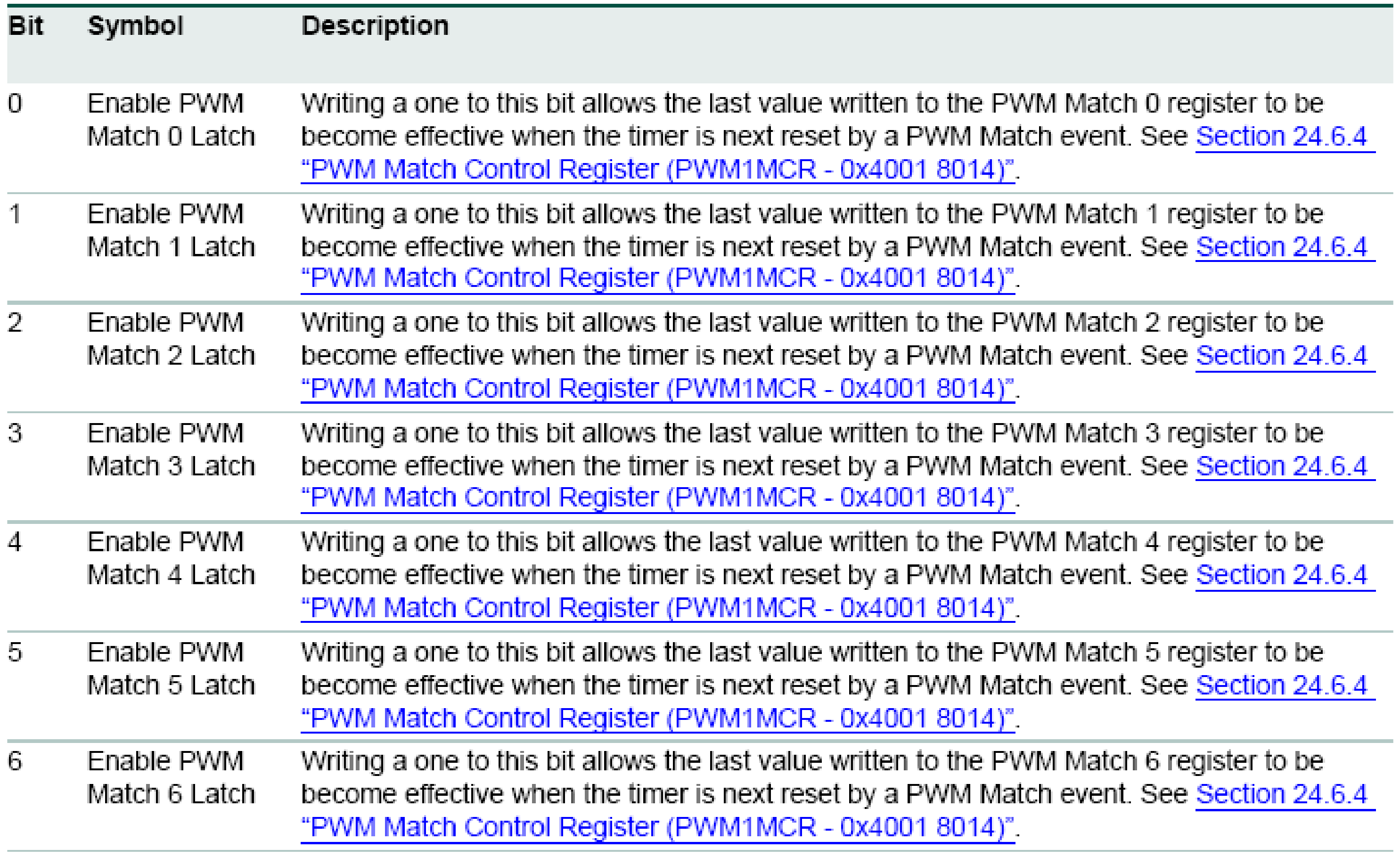
Bit	Symbol	Value	Description
1:0	Unused		Unused, always zero.
2	PWMSEL2	1	Selects double edge controlled mode for the PWM2 output.
		0	Selects single edge controlled mode for PWM2.
3	PWMSEL3	1	Selects double edge controlled mode for the PWM3 output.
		0	Selects single edge controlled mode for PWM3.
4	PWMSEL4	1	Selects double edge controlled mode for the PWM4 output.
		0	Selects single edge controlled mode for PWM4.
5	PWMSEL5	1	Selects double edge controlled mode for the PWM5 output.
		0	Selects single edge controlled mode for PWM5.
6	PWMSEL6	1	Selects double edge controlled mode for the PWM6 output.
		0	Selects single edge controlled mode for PWM6.
8:7	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.



Bit	Symbol	Value	Description
9	PWMENA1	1	The PWM1 output enabled.
		0	The PWM1 output disabled.
10	PWMENA2	1	The PWM2 output enabled.
		0	The PWM2 output disabled.
11	PWMENA3	1	The PWM3 output enabled.
		0	The PWM3 output disabled.
12	PWMENA4	1	The PWM4 output enabled.
		0	The PWM4 output disabled.
13	PWMENA5	1	The PWM5 output enabled.
		0	The PWM5 output disabled.
14	PWMENA6	1	The PWM6 output enabled.
		0	The PWM6 output disabled.
31:15	Unused		Unused, always zero.

Pulse Width Modulation (PWM)

PWM Latch Enable Register (PWM1LER)



Bit	Symbol	Description
0	Enable PWM Match 0 Latch	Writing a one to this bit allows the last value written to the PWM Match 0 register to be become effective when the timer is next reset by a PWM Match event. See Section 24.6.4 “PWM Match Control Register (PWM1MCR - 0x4001 8014)”.
1	Enable PWM Match 1 Latch	Writing a one to this bit allows the last value written to the PWM Match 1 register to be become effective when the timer is next reset by a PWM Match event. See Section 24.6.4 “PWM Match Control Register (PWM1MCR - 0x4001 8014)”.
2	Enable PWM Match 2 Latch	Writing a one to this bit allows the last value written to the PWM Match 2 register to be become effective when the timer is next reset by a PWM Match event. See Section 24.6.4 “PWM Match Control Register (PWM1MCR - 0x4001 8014)”.
3	Enable PWM Match 3 Latch	Writing a one to this bit allows the last value written to the PWM Match 3 register to be become effective when the timer is next reset by a PWM Match event. See Section 24.6.4 “PWM Match Control Register (PWM1MCR - 0x4001 8014)”.
4	Enable PWM Match 4 Latch	Writing a one to this bit allows the last value written to the PWM Match 4 register to be become effective when the timer is next reset by a PWM Match event. See Section 24.6.4 “PWM Match Control Register (PWM1MCR - 0x4001 8014)”.
5	Enable PWM Match 5 Latch	Writing a one to this bit allows the last value written to the PWM Match 5 register to be become effective when the timer is next reset by a PWM Match event. See Section 24.6.4 “PWM Match Control Register (PWM1MCR - 0x4001 8014)”.
6	Enable PWM Match 6 Latch	Writing a one to this bit allows the last value written to the PWM Match 6 register to be become effective when the timer is next reset by a PWM Match event. See Section 24.6.4 “PWM Match Control Register (PWM1MCR - 0x4001 8014)”.

Pulse Width Modulation (PWM)

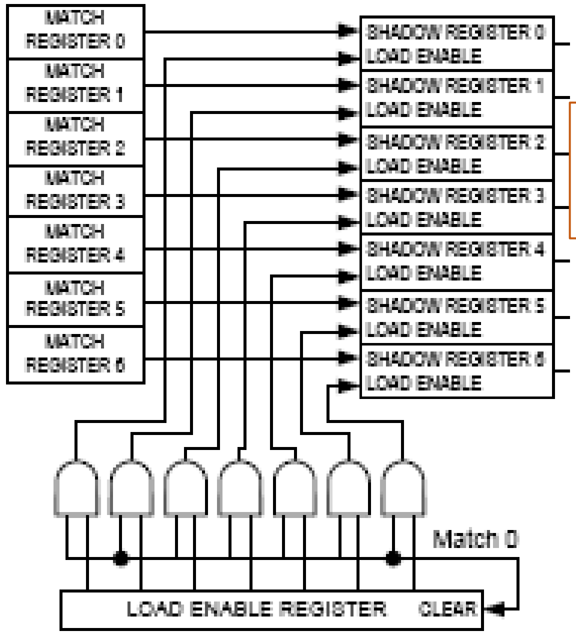
PWM Latch Enable Register (PWM1LER)

The PWM Latch Enable Registers are used to control the update of the PWM Match registers when they are used for PWM generation.

When a PWM Match 0 event occurs (normally also resetting the timer in PWM mode), the contents of shadow registers will be transferred to the shadow registers if the corresponding bit in the Latch Enable Register has been set.

At that point, the new values will take effect and determine the course of the next PWM cycle. Once the transfer of new values has taken place, all bits of the LER are automatically cleared.

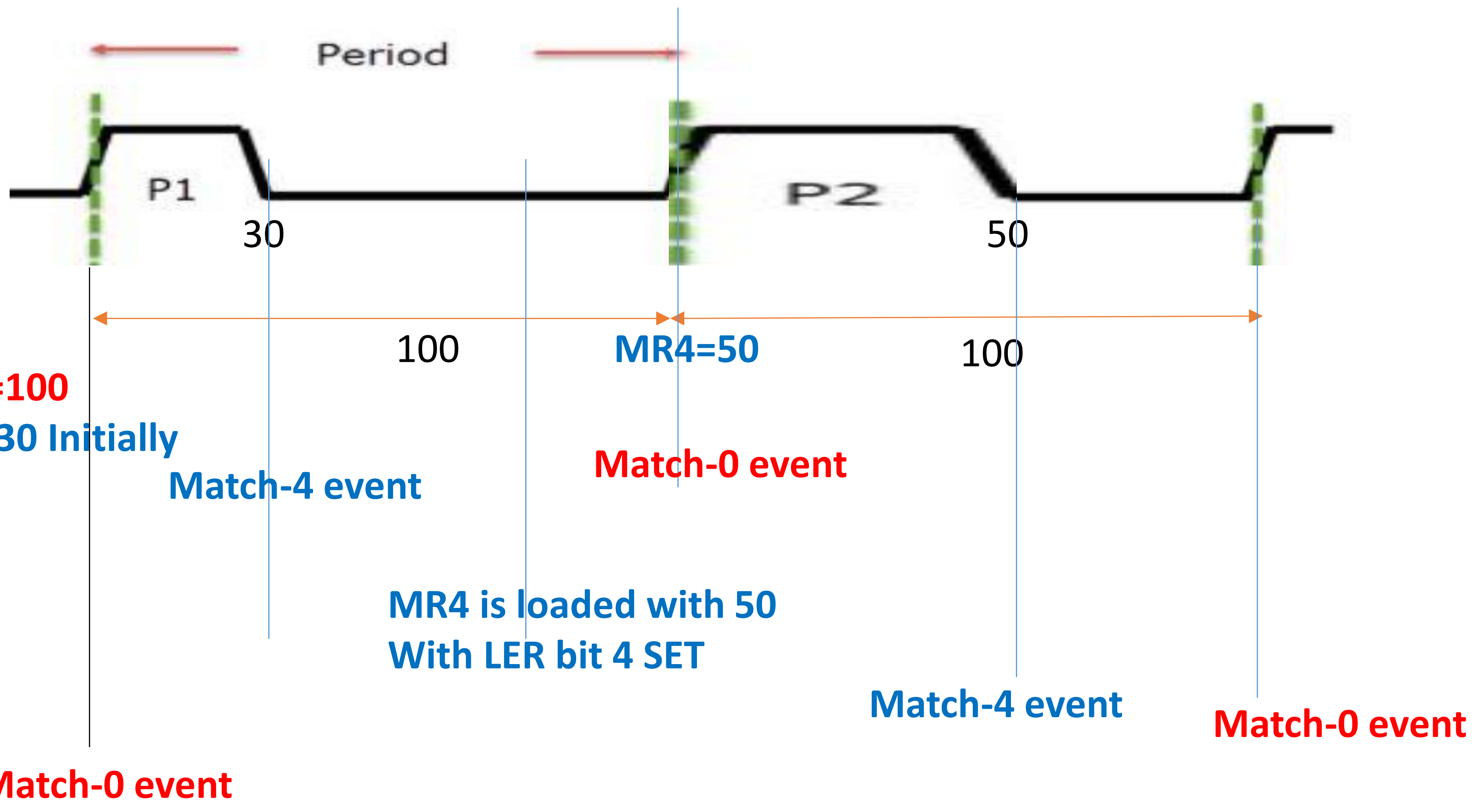
Until the corresponding bit in the PWMLER is set and a PWM Match 0 event occurs, any value written to the PWM Match registers has no effect on PWM operation.



To be compared with TC to generate Match0-Match6 outputs

Pulse Width Modulation (PWM)

Single edge PMM1.4



MR0 =100

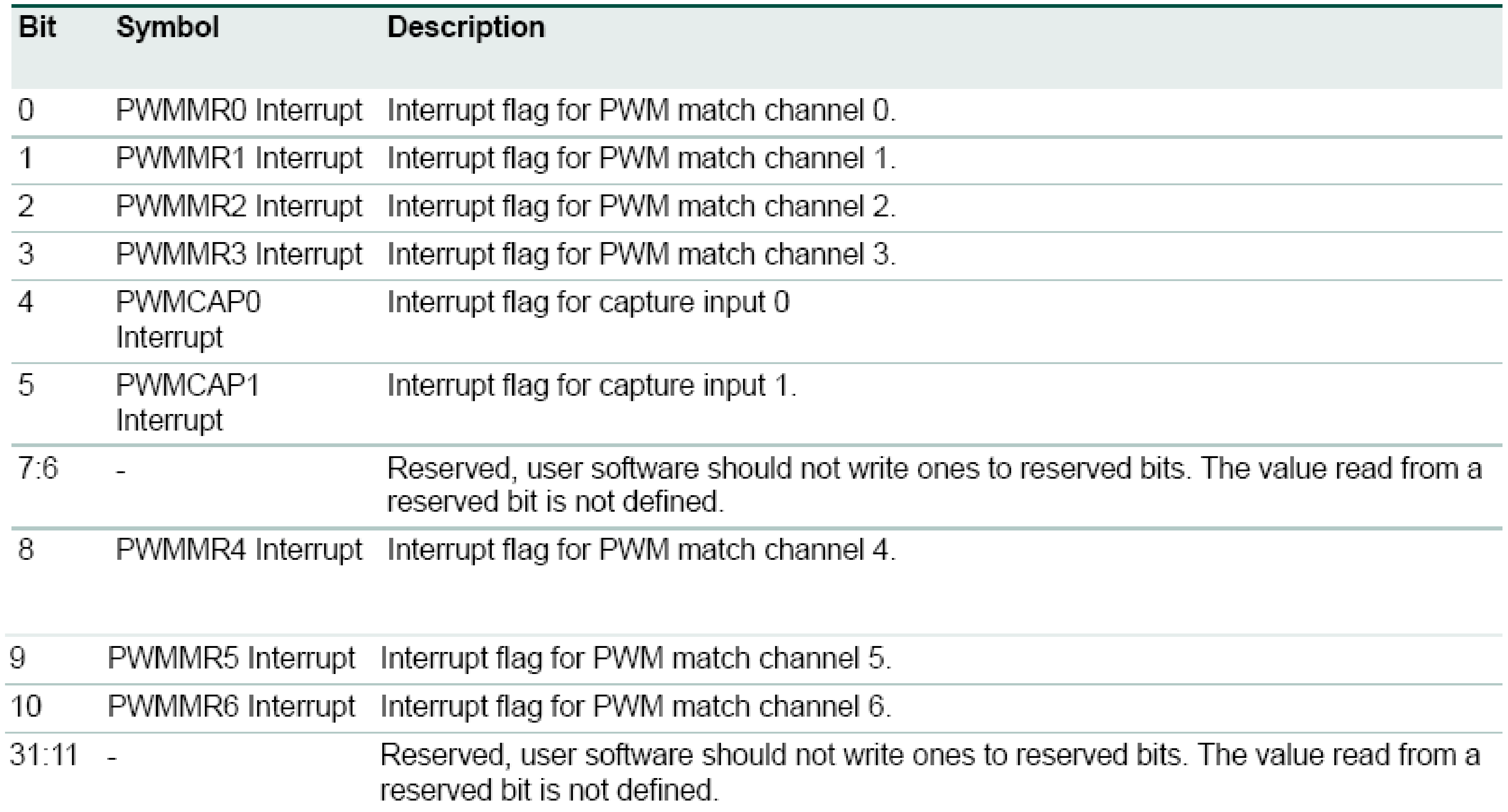
MR4=30 Initially

Match-0 event

Pulse Width Modulation (PWM)

PWM Interrupt Register (PWM1IR)

The PWM Interrupt Register consists of 9 INTR Flags (7 Match, 2 Capture) If an interrupt is generated then the corresponding bit in the PWMIR will be high. Otherwise, the bit will be low. Writing a logic 1 to the corresponding IR bit will reset the interrupt.

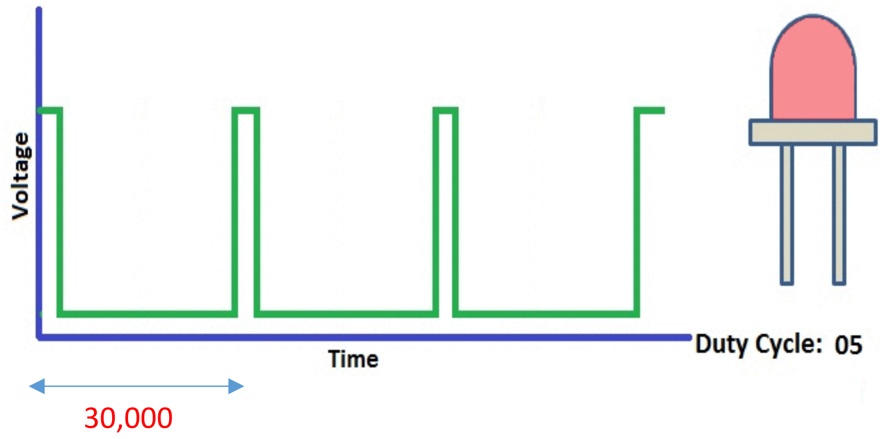


Bit	Symbol	Description
0	PWMMR0 Interrupt	Interrupt flag for PWM match channel 0.
1	PWMMR1 Interrupt	Interrupt flag for PWM match channel 1.
2	PWMMR2 Interrupt	Interrupt flag for PWM match channel 2.
3	PWMMR3 Interrupt	Interrupt flag for PWM match channel 3.
4	PWMCAP0 Interrupt	Interrupt flag for capture input 0
5	PWMCAP1 Interrupt	Interrupt flag for capture input 1.
7:6	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
8	PWMMR4 Interrupt	Interrupt flag for PWM match channel 4.
9	PWMMR5 Interrupt	Interrupt flag for PWM match channel 5.
10	PWMMR6 Interrupt	Interrupt flag for PWM match channel 6.
31:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

Pulse Width Modulation (PWM)

PWM program to change the intensity of LED connected to (PWM1.4) P1.23 continuously

```
#include <LPC17xx.H>
 void pwm_init(void);
 void PWM1_IRQHandler(void);
unsigned int a;
 unsigned long int i;
 unsigned char flag,flag1;
  int main(void)
 {
                SystemInit();
                SystemCoreClockUpdate();
                pwm_init();
                while(1);
                 }
  void pwm_init(void)
 {              LPC_PINCON->PINSEL3 = 2<<14;                          //pwm1.4 is selecte
                LPC_PWM1->PR  = 0x00000000;     //Count frequency : Fpclk
                LPC_PWM1->PCR = 0x00001000;     //select PWM1 single edge
                LPC_PWM1->MCR = 0x00000003;     //Reset and interrupt on PWMMR0
                LPC_PWM1->MR0 = 30000;          //setup match register 0 count
                LPC_PWM1->MR4 = 50;      //setup match register MR4
                LPC_PWM1->LER = 0x000000FF;     //enable shadow copy register
                LPC_PWM1->TCR = 0x00000002;     //RESET TC and PC
                LPC_PWM1->TCR = 0x00000009;     //enable PWM and counter
                NVIC_EnableIRQ(PWM1_IRQn);
                return;
 }
```



Pulse Width Modulation (PWM)

```
void PWM1_IRQHandler(void)
{
        LPC_PWM1->IR = 0x01; //clear the interrupts
        if(flag == 0x00)
  {
                LPC_PWM1->MR4 += 50;
                LPC_PWM1->LER = 0x000000FF;
                if(LPC_PWM1->MR4 > 29900)
                {
        flag = 0xff;
        LPC_PWM1->LER = 0x000000FF;
                }
        }
  else if(flag == 0xff)
  {
                LPC_PWM1->MR4 -= 50;
                LPC_PWM1->LER = 0x000000fF;
                if(LPC_PWM1->MR4 < 100)
                {
                        flag  = 0x00;
                        LPC_PWM1->LER = 0X000000FF;
                }
        }
}
```