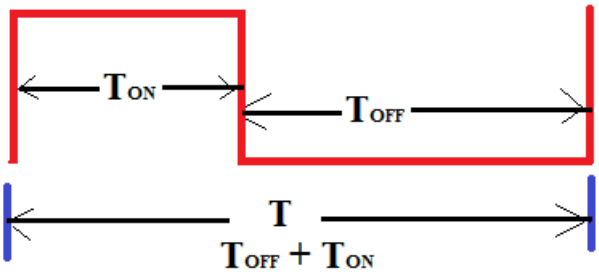
LPC 2148 PWM Unit and Programming

The PWM is based on the standard Timer block and inherits all of its features. When PWM unit is not required in the project, one can use the PWM unit as third timer unit.

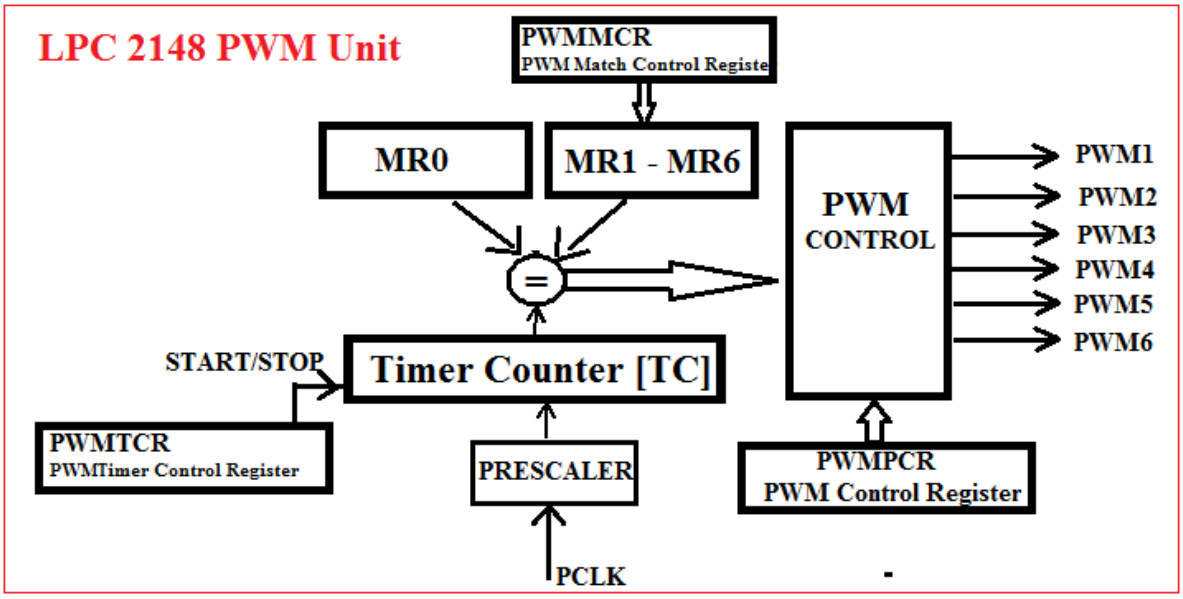
What is PWM: PWM output is basically a high and low waveform, whose ON time & Period (On+Off) can be varied, i.e we can generate the waveform with programmable Pulse rate (period of one cycle- On + Off time) & Pulse period(On time) and hence achieving different duty cycles (ratio of ON time to ON+OFF time).



$$\text{DUTY CYCLE} = \frac{T_{ON}}{T_{OFF} + T_{ON}} = \frac{T_{ON}}{T}$$

Duty cycle = [Ton / (Ton + Toff)] x 100 in%

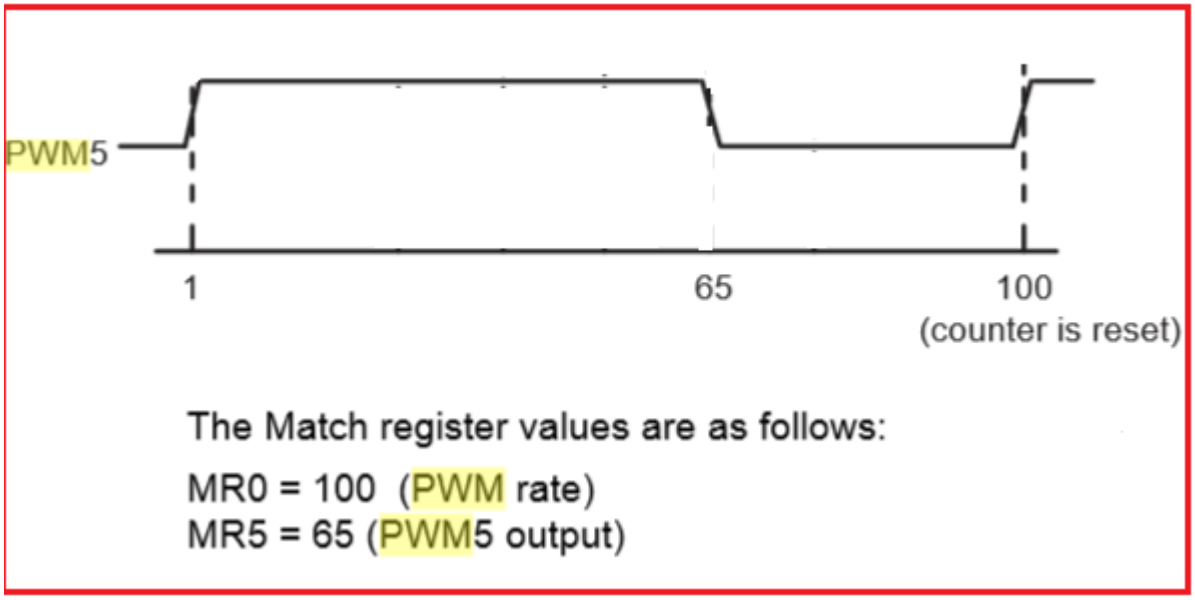
LPC 2148 PWM Unit: The PWM is based on the standard Timer block and inherits all of its features. When PWM unit is not required in the project, one can use the PWM unit as third timer unit.



LPC 2148 PWM unit provides 6 PWM channels, PWM 1 to 6. Each of these channels can be connected to motor driver circuits, to control their speed. PWM waveforms are very commonly used to drive DC motors, Servo motors and for other applications.

The registers of PWM unit are similar to timer. It provides 32 bit timer counter for counting PCLK cycles, to derive the required Pulse rate and Pulse period. One Match Register is required to define the number of PCLK cycles required to make one T, period (Ton + Toff), to simplify the hardware, all the 6 PWM channels have the same Pulse rate i.e T. MR0 is used to define this value. To define, different pulse periods for six different PWM channels, PWM1 TO PWM6, six Match Registers are provided, MR1 to MR6. The number, which is loaded into these registers, define number of PCLK cycles will constitute Ton time. Pulse period and width can be any number of timer counts and all PWM outputs will occur at the same repetition rate.

The waveform below show a single PWM cycle, for PWM5 Channel for the indicated values of MR0 and MR5. Total period of PWM is 100 PCLK cycles, Ton period is 65 PCLK cycles. At the beginning of the PWM cycle, output becomes '1'(HIGH), after the Ton time, it goes low, logic '0'.



PWM Registers :

PWMTC - PWM timer counter , 32 bits, similar to timer Timer Count(TC) register, counts up using PCLK, from 0000 0000h to FFFF FFFFH. Its value can be reset, before reaching the maximum value, based on the Match Register contents.

PRE-SCALER (PWMPR, PWMPC) – works similar to timer pre-scaler.

PWMTCR - PWM Timer Control Register, similar to timer TCR , used to Start/Stop the timer, Reset the counter, Enable/Disable PWM. If PWM is disabled, PWM unit acts like other timer. When PWM is not required, programmers use PWM as third timer.

Bit	Symbol	Description	Reset value
0	Counter Enable	When one, the PWM Timer Counter and PWM Prescale Counter are enabled for counting. When zero, the counters are disabled.	0
1	Counter Reset	When one, the PWM Timer Counter and the PWM Prescale Counter are synchronously reset on the next positive edge of PCLK. The counters remain reset until TCR[1] is returned to zero.	0
2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
3	PWM Enable	When one, PWM mode is enabled. PWM mode causes shadow registers to operate in connection with the Match registers. A program write to a Match register will not have an effect on the Match result until the corresponding bit in PWMLER has been set, followed by the occurrence of a PWM Match 0 event. Note that the PWM Match register that determines the PWM rate (PWM Match 0) must be set up prior to the PWM being enabled. Otherwise a Match event will not occur to cause shadow register contents to become effective. **Remark:** The PWM Enable bit needs to be always set to 1 for PWM operation. Otherwise, the PWM will behave as standard timer .	0
7:4	-	Reserved, user software should not write ones to	NA

MR0 - Match Register 0 value is used to define the pulse rate. It defines the pulse rate, i.e period T (ON+OFF).
MR1 to MR6 – 32 bit Match registers, used to fix the ON time of PWM waveforms, the values of these registers are continuously compared to the count in the timer register, and generates the corresponding PWM waveforms on channels PWM1 to PWM6 , so by changing the values in the MR1-MR6 we can produce different duty cycles.

PWMMCR – PWM Match Control Register, is useful when PWM hardware is used as timer, to configure the action to be initiated, when PWMTC matches with seven Match registers, MR0 –MR7. Bit allocation of PWMMCR is similar to timer MCR, but it is for 7 Match registers.

PWMPCR – PWM control register, used to selectively enable / disable PWM channels PWM1 – PWM6. Also this register is used to configure the PWM channels as
a) single edge
b) double edge.

In single edge the output is always high, we are only controlling the falling edge (that happens whenever count matches to the match registers). In double edge , we are controlling both the rising and falling edges of the pulse (i.e why we require two match registers for one channel, first match register controls the rising edge, other one controls the falling edge). Here, we are restricting our study for only single edge PWM generation. (by default, PWM channels work as single edge type channels).

PWMPCR - D15 D14 D13……D9.. D6 D5 …..D2 D1 D0
(Bits D9 to D14 are used to enable the channels PWM1 to PWM6)
Bit D9 – 1 (Enable PWM 1) 0 (Disable PWM1)
Bit D10 -1 (Enable PWM 2) 0 (Disable PWM2)
Bit D11 -1 (Enable PWM 3) 0 (Disable PWM3)
Bit D12 -1 (Enable PWM 4) 0 (Disable PWM4)
Bit D13 -1 (Enable PWM 5) 0 (Disable PWM5)
Bit D14 -1 (Enable PWM 6) 0 (Disable PWM6)

PWMLER – PWM Latch Enable Register, is used to control the update of the PWM Match registers when they are used for PWM generation. When software writes to Match register, the value is held in the shadow register. When PWM Match0 event occurs, the contents of shadow registers will be transferred to the actual Match registers, if the corresponding bit in the Latch Enable Register is 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 the bits of 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.

Bits:	D7	D6	D5	D4	D3	D2	D1	D0
		MR6	MR5	MR4	MR3	MR2	MR1	MR0

- '1' – Writing 1 to this bit allows the last value written to the Match register to become effective, when the timer is next by a PWM Match event.

Note:

1. **All single edge controlled PWM outputs go high at the beginning of a PWM cycle, unless their match value is equal to zero.**
2. **Each PWM output will go low when its match value is reached. If no match occurs (i.e the match value is greater than the PWM rate), the PWM output remains continuously high.**
3. **PWM Latch enable register, 8bit register - bit D0-D6 are for MR0- MR6, usually what we write to match registers will go to shadow registers, while timer is running when the match event occurs, then it will be copied to match registers, only if the corresponding bits in the PWMLER is enabled.**

Example: Generate the PWM waveform of 25% duty cycle at the PWM channel 3

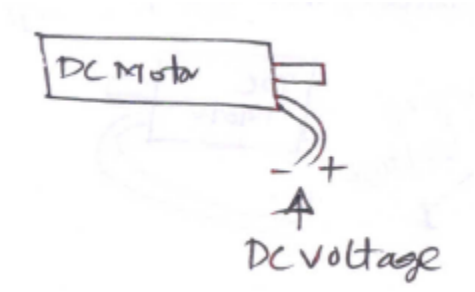
```
#include <LPC214x.h>
void PWM_Init(void)
 {
      //P0.1 pin has second alternate function as PWM3 channel, so using
      PINSEL0 register
      PINSEL0 |=  0x00000008; // Select  P0.1 as PWM output ,  bits D2 & D3
      are for P0.1
      //Configure PWM channel 3  as single edge type and enable the channel
      PWMPCR = 0x00000800; //bit D3 to select single edge(make it  0), bit
      D11 is for
                                    //enabling PWM3 channel(make it 1)
      //load the value to MR0 to fix the pulse rate
      PWMMR0 = 2000; // any other value, I could have taken
      // enable PWM unit of LPC2148 and start the timer
      PWMTCR = 0x0000 0009;   // bit D3 = 1 (enable  PWM), bit D0=1 (start
      the timer)
 }

int main()
{
PWM_Init();
while(1)
 {
      PWMMR3  =   500; // value which decides pulse ON time, ¼ of 2000
      PWMLER  = 0X08;

 }
}
```

Calculation of duty cycle in the above program: if PCLK is 15MHz, then T= 0.067µsec, Pulse rate, T = ON+OFF = 2000 * T

Puse width, Ton = 500 * T

Duty cycle = Ton/T = 500*T/2000*T = ¼= 0.25 x 100 = 25 %

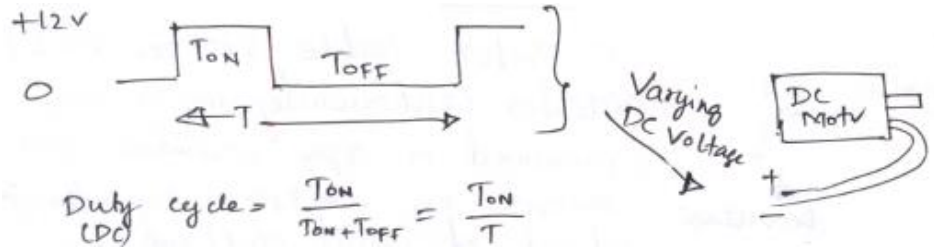
DC Motor Control Using LPC 2148 PWM Unit



DC motor, unlike steeper motor rotates continuously not in steps (measured in rpm – revolutions per minute ex: 100 rpm). By connecting the terminals of motor to power supply (+12V/+24V/+48V) the motor rotates at full speed. By changing polarity of input voltage the direction of rotation can be changed. By changing the input dc voltage (within the range specified by the manufacture) the speed of the motor can be changed (ex: Fan used in CPU cooling uses dc motor).

DC Motor speed control:

Achieved using PWM (pulse width modulation by changing width of the pulse, we can increase / decrease the power to the motor for a given load.

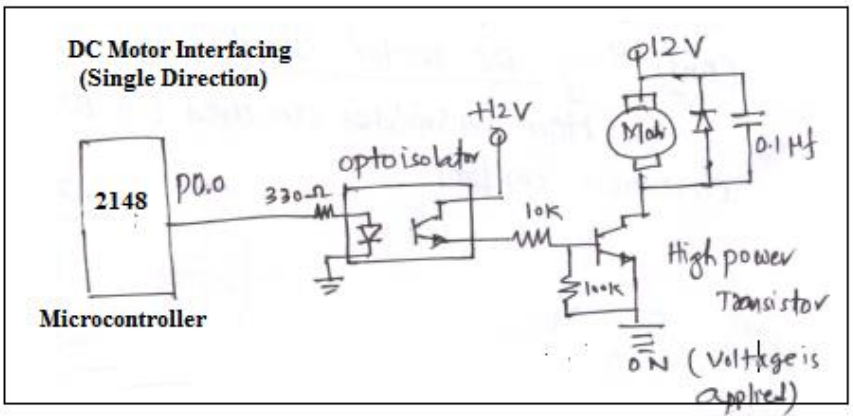


Effective dc voltage applied to DC motor $= \frac{T_{ON}}{T} \times suplly\ voltage\ (say\ 12V)$

(if $T_{ON} = T, Full\ power\ i.e. 12V, If\ T_{ON} = \frac{T}{2},\ 50\%\ DC\ \left(\frac{1}{2}\ power\right)$

By controlling T_{ON}time, effective DC voltage applied to motor is changed, hence the speed.

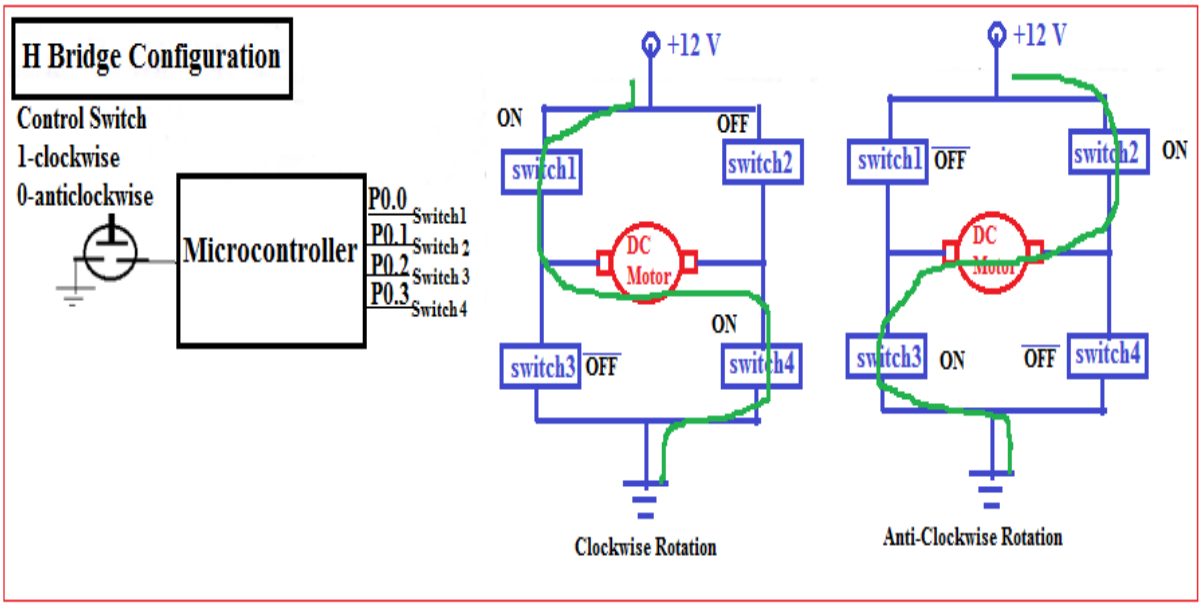
How can we drive the motor with microcontroller control. As Microcontroller can't supply the required voltage and current required by DC motor, we use driver circuit, like transistor (acting like a switch) to drive the motor as shown in the figure. Here when we send '1' to P0.0, transistor becomes ON, motor conducts. Sending logic '0', transistor becomes switched off, motor stops rotating.



H-Bridge Circuit: Controlling DC motor direction :

To make the DC motor run in both the directions, motor is connected in the H-Bridge configuration, as shown in the below diagram. Here, four transistor switches are used, connected with the motor and power-supply in the 'H' shape.

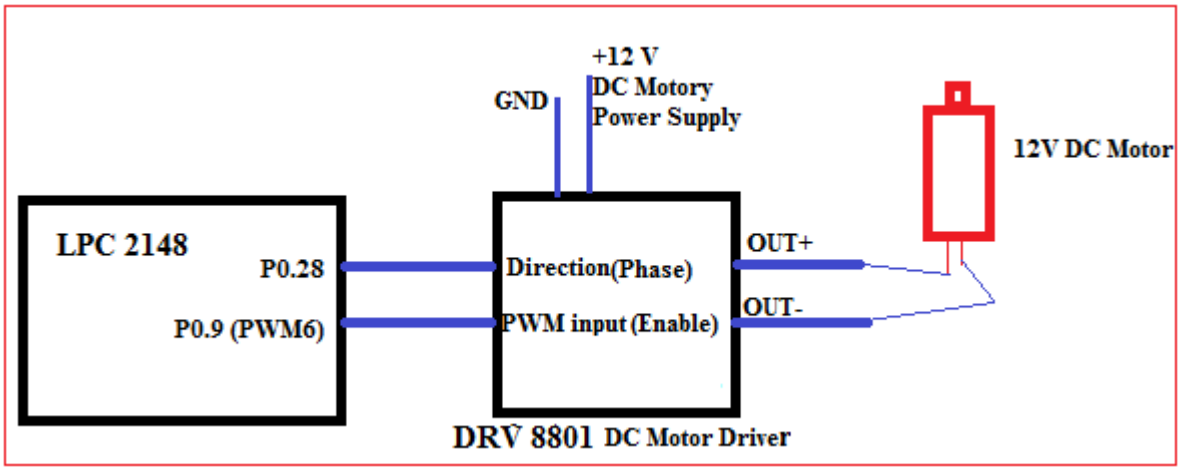
To run the motor in the clockwise direction, switch on – SW1,SW4, switch off – SW2,SW3
To run the motor in the anticlockwise direction: SW2,SW3-ON and SW1,SW4 - OFF



Many H Bridge circuits are commercially available, in the single chip, like L298, DRV 8801 etc. They are popularly referred as DC Motor Driver ICs. In general, all of them provide two inputs, which are connected to Microcontroller port pins.

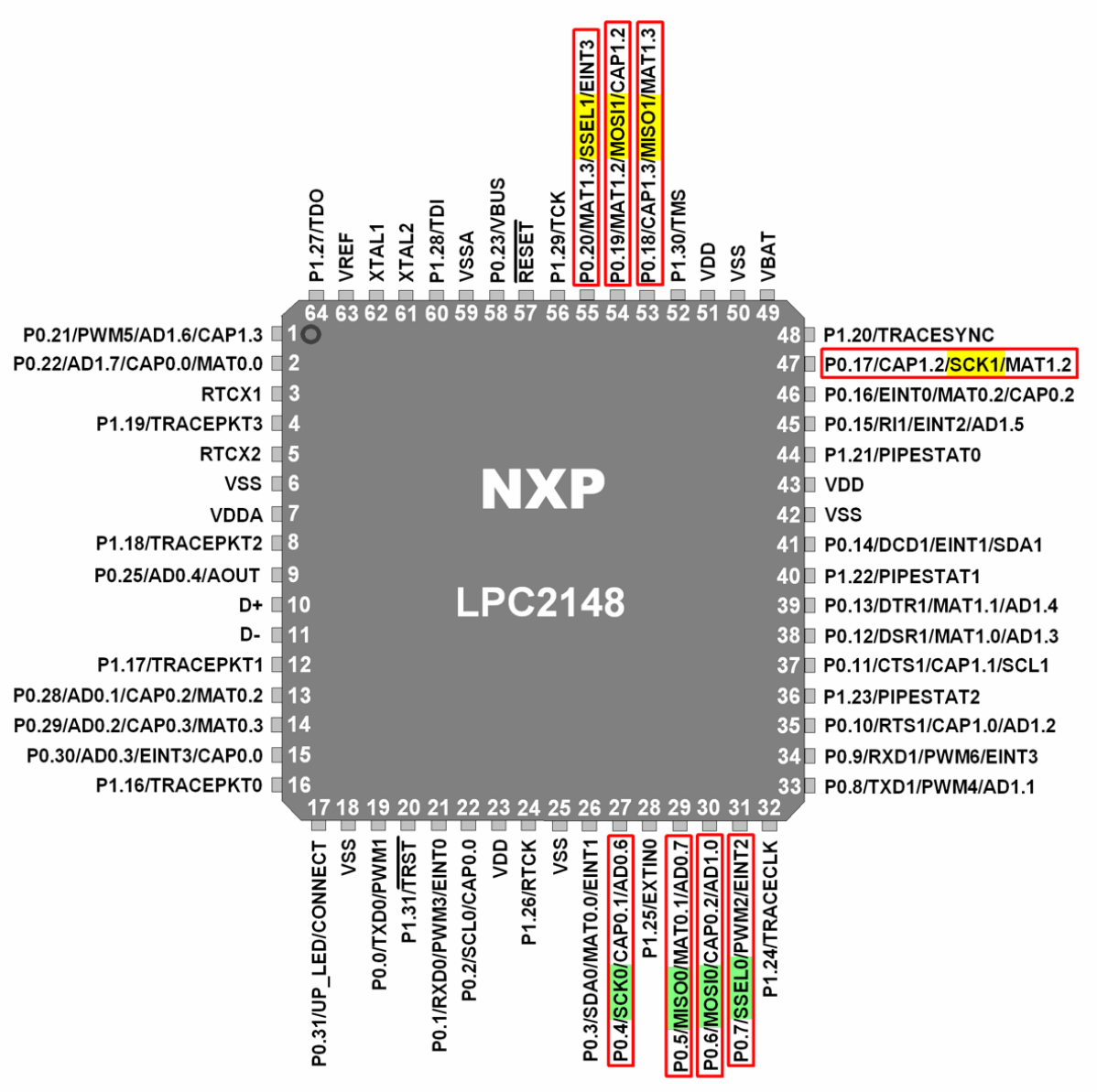
1. Direction control (0 – clock wise, 1 – anti-clock wise), any GPIO pin can be used
2. PWM input (to control the speed of the DC Motor), PWM pin can be used.

Interface DC Motor to LPC 2148 and write an Embedded C program to generate PWM wave to control speed of DC motor.



Embedded C Program:

```
//DC Motor Speed Control
//P0.28 - used for direction control
//P0.9  - used for speed,generated by PWM6
//duty cycle - 0 to 100 controlled by PWM,

#include <lpc214x.h>
void runDCMotor(int direction, int dutycycle) ;
int main()
{
      IO0DIR |= 1U << 28;     //set P0.28 as output pin
      PINSEL0 |= 2 << 18;   //select P0.9 as PWM6 (option 2)
      runDCMotor(2,10); // run at 10% duty cycle
      while(1);  // do other jobs
}
void runDCMotor(int direction, int dutycycle)
{
          if (direction == 1)
                IO0SET = 1 << 28;   //set to 1, to choose anti-clockwise direction
          else
                IO0CLR = 1 << 28;  //set to 0, to choose clockwise direction

          PWMPCR = (1 << 14);  // enable PWM6 channel
          PWMMR0 = 1000;   // set PULSE rate to value suitable for DC Motor operation
          PWMMR6 = (1000U*dutycycle)/100;  // set PULSE period
          PWMTCR = 0x00000009;  // bit D3 = 1 (enable  PWM), bit D0=1 (start the timer)
          PWMLER = 0X70;  // load the new values to PWMMR0 and PWMMR6 registers
}
```


NXP
LPC2148
P1.27/TDO
VREF
XTAL1
XTAL2
P1.28/TDI
VSSA
P0.23/VBUS
RESET
P1.29/TCK
P0.20/MAT1.3/SSEL1/EINT3
P0.19/MAT1.2/MOSI1/CAP1.2
P0.18/CAP1.3/MISO1/MAT1.3
P1.30/TMS
VDD
VSS
VBAT
64 63 62 61 60 59 58 57 56 55 54 53 52 51 50 49
P0.21/PWM5/AD1.6/CAP1.3 1
P0.22/AD1.7/CAP0.0/MAT0.0 2
RTCX1 3
P1.19/TRACEPKT3 4
RTCX2 5
VSS 6
VDDA 7
P1.18/TRACEPKT2 8
P0.25/AD0.4/AOUT 9
D+ 10
D- 11
P1.17/TRACEPKT1 12
P0.28/AD0.1/CAP0.2/MAT0.2 13
P0.29/AD0.2/CAP0.3/MAT0.3 14
P0.30/AD0.3/EINT3/CAP0.0 15
P1.16/TRACEPKT0 16
48 P1.20/TRACESYNC
47 P0.17/CAP1.2/SCK1/MAT1.2
46 P0.16/EINT0/MAT0.2/CAP0.2
45 P0.15/RI1/EINT2/AD1.5
44 P1.21/PIPESTAT0
43 VDD
42 VSS
41 P0.14/DCD1/EINT1/SDA1
40 P1.22/PIPESTAT1
39 P0.13/DTR1/MAT1.1/AD1.4
38 P0.12/DSR1/MAT1.0/AD1.3
37 P0.11/CTS1/CAP1.1/SCL1
36 P1.23/PIPESTAT2
35 P0.10/RTS1/CAP1.0/AD1.2
34 P0.9/RXD1/PWM6/EINT3
33 P0.8/TXD1/PWM4/AD1.1
17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
P0.31/UP_LED/CONNECT
VSS
P0.0/TXD0/PWM1
P1.31/TRST
P0.1/RXD0/PWM3/EINT0
P0.2/SCL0/CAP0.0
VDD
P1.26/RTCK
VSS
P0.3/SDA0/MAT0.0/EINT1
P0.4/SCK0/CAP0.1/AD0.6
P1.25/EXTIN0
P0.5/MISO0/MAT0.1/AD0.7
P0.6/MOSI0/CAP0.2/AD1.0
P0.7/SSEL0/PWM2/EINT2
P1.24/TRACECLK