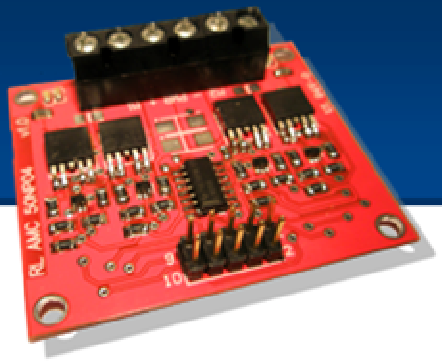
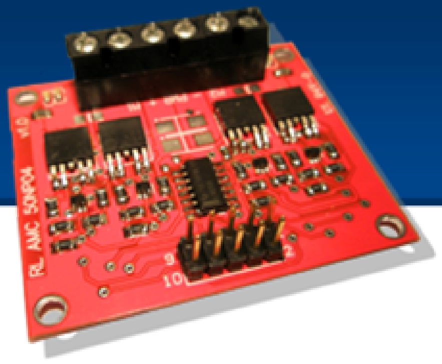
RL AMC 50NP04

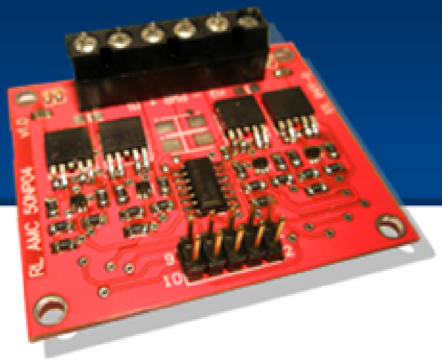
Reference source code



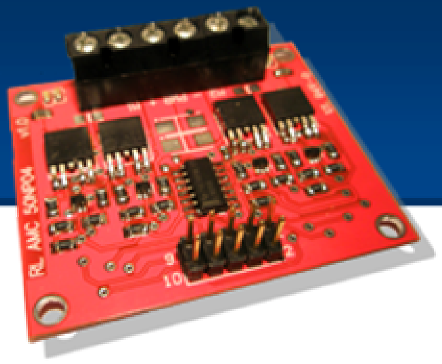
```c
//--------------- main.c ---------------


/////////////////////////////////////////////////////////
// Filename: main.c
// Authors:  Daniel Alvarez <dani001@gmail.com>
//           Alberto Calvo  <albertoct@gmail.com>
//
// Description:
//
// RL-AMC-50NP04 Board test software.
// This software allows to test the RL-AMC-50NP04 board and control two
// DC motors by providing peed change, braking and inverting functions
// through the LPC2138-01 board.
//
// You can find RL-AMC-50NP04 specifications at
// http://dani.foroselectronica.es/h-bridge-mosfet-board-rl-amc-50np04-71/
//
// You can find LPC2138-01 specifications at
// http://dani.foroselectronica.es/arm-development-board-lpc2138-01-25/
//
//
//          +----+----+----+-----------+
//          | M3 | M2 | M1 |  ACTION   |
//          +--------------------------+
//          | 1  | 0  | 0  |   BRAKE   |
//          +--------------------------+
//          | 1  | 0  | 1  |  FORWARD  |
//          +--------------------------+
//          | 1  | 1  | 0  |  REVERSE  |
//          +--------------------------+
//          | 1  | 1  | 1  |   BRAKE   |
//          +--------------------------+
//          | 0  | X  | X  |   COAST   |
//          +--------------------------+
//
//          +--------------------+
//          |      MOTOR 1       |
//          +-----+-------+------+
//          | M11 | P0.3  | J5.4 |
//          +--------------------+
//          | M12 | P0.17 | J7.2 |
//          +-----+-------+------+
//          | M13 | P0.7  | J5.8 |
//          +-----+-------+------+
//
//          +--------------------+
//          |      MOTOR 2       |
//          +--------------------+
//          | M21 | P0.2  | J5.3 |
//          +--------------------+
//          | M22 | P0.20 | J7.5 |
//          +-----+-------+------+
//          | M23 | P0.21 | J7.6 |
//          +--------------------+
//
//
// Copyright 2008.
// All the software and boards have been developed by
// Daniel Alvarez and Alberto Calvo
//
/////////////////////////////////////////////////////////////


#include <LPC213x.H>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "motor.h"
#include "serial.h"
```



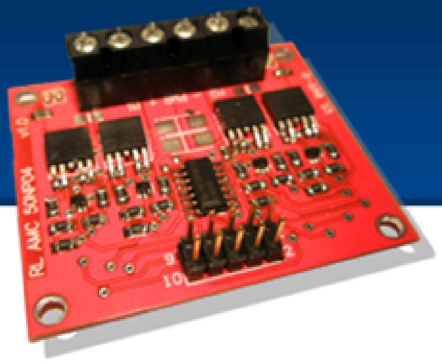
```
char buffer[512];

void PrintHelp(void)
{
    sprintf(buffer,"Motor Driver Test Software\r\n");
    usb_putbuffer(buffer,strlen(buffer));

    sprintf(buffer,"Instructions:\r\n");
    usb_putbuffer(buffer,strlen(buffer));
    sprintf(buffer,"A/Z: +\r\nS/X: -\r\nD/C: Invert\r\nF/V: Stop\r\nG/B: Brake\r\n");
    usb_putbuffer(buffer,strlen(buffer));
    sprintf(buffer,"H: Print this help\r\n");
    usb_putbuffer(buffer,strlen(buffer));

}

int main()
{
    char c;
    int speed1=0, speed2=0;

    init_serial();
    motor_init();

    set_speed_motor_left(0);
    set_speed_motor_right(0);

    PrintHelp();

    while(1)
    {
        c=usb_getkey();
        usb_putchar(c);
        usb_putchar('\r');
        usb_putchar('\n');
        switch(c)
        {
            case 'A':
            case 'a':
                speed1+=48;
                if(speed1 > 2048)
                    speed1 = 2048;
                set_speed_motor_left(speed1);
            break;
            case 'S':
            case 's':
                speed1-=48;
                if(speed1 < -2048)
                    speed1 = -2048;
                set_speed_motor_left(speed1);
            break;
            case 'D':
            case 'd':
                speed1=-speed1;
                set_speed_motor_left(speed1);
                 break;
            case 'F':
            case 'f':
                 speed1=0;
                 set_speed_motor_left(0);
                 break;
            case 'G':
            case 'g':
                 brake_motor_left();
                 break;
            case 'Z':
            case 'z':
                speed2+=48;
```



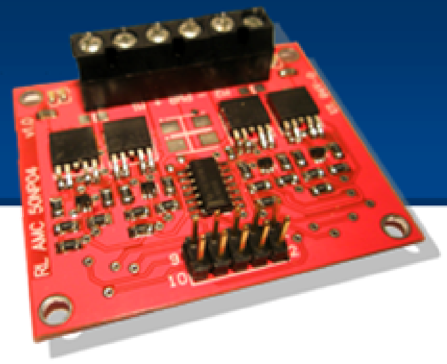
```
                if(speed2 > 2048)
                    speed2 = 2048;
                set_speed_motor_right(speed2);
            break;
            case 'X':
            case 'x':
                speed2-=48;
                if(speed2 < -2048)
                    speed2 = -2048;
                set_speed_motor_right(speed2);
            break;
            case 'C':
            case 'c':
                speed2=-speed2;
                set_speed_motor_right(speed2);
                break;
            case 'V':
            case 'v':
                speed2=0;
                set_speed_motor_right(0);
            break;
            case 'B':
            case 'b':
                 brake_motor_right();
                 break;
            case 'H':
            case 'h':
                 PrintHelp();
                 break;
        }
    }

}
```



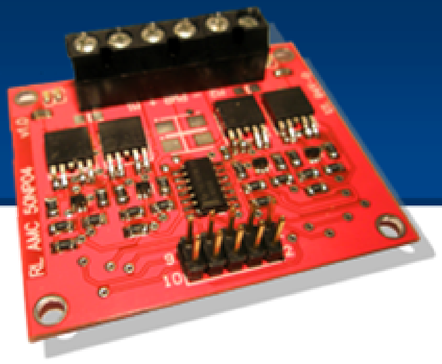
```
//--------------- motor.c ---------------

#include <LPC213x.H>
#include "motor.h"
#include "pwm.h"
#include <stdio.h>
#include <string.h>

#define MOTOR_LEFT_P2       (1<<17)         //  p0.17   - M12 - J7.2
#define MOTOR_RIGHT_P2      (1<<20)         //  p0.20   - M22 - J7.5
#define MOTOR_LEFT_P1       (1<<3)          //  p0.3    - M11 - J5.4
#define MOTOR_RIGHT_P1      (1<<2)          //  p0.2    - M21 - J5.3



void motor_init()
{
    /* P0.20 and P0.17 defined as Outputs */
    IODIR0 |= (MOTOR_LEFT_P2 | MOTOR_LEFT_P1 | MOTOR_RIGHT_P2 | MOTOR_RIGHT_P1);
    pwm_init();

    set_pwm2_duty(0);
    set_pwm5_duty(0);

    IOCLR0 |= MOTOR_LEFT_P2;
    IOCLR0 |= MOTOR_RIGHT_P2;
    IOCLR0 |= MOTOR_LEFT_P1;
    IOCLR0 |= MOTOR_RIGHT_P1;

    pwm_start();
}

void set_speed_motor_right(int speed)
{

    set_pwm5_duty(0);

    if(speed>0)
    {
        IOSET0 |= MOTOR_RIGHT_P1;
        IOCLR0 |= MOTOR_RIGHT_P2;
    }
    else
    {
        speed=-speed;
        IOSET0 |= MOTOR_RIGHT_P2;
        IOCLR0 |= MOTOR_RIGHT_P1;
    }

    set_pwm5_duty(speed);
}

void set_speed_motor_left(int speed)
{

    set_pwm2_duty(0);

    if(speed>0)
    {
        IOSET0 |= MOTOR_LEFT_P1;
        IOCLR0 |= MOTOR_LEFT_P2;
    }
    else
    {
        speed=-speed;
        IOSET0 |= MOTOR_LEFT_P2;
        IOCLR0 |= MOTOR_LEFT_P1;
    }

    set_pwm2_duty(speed);
```



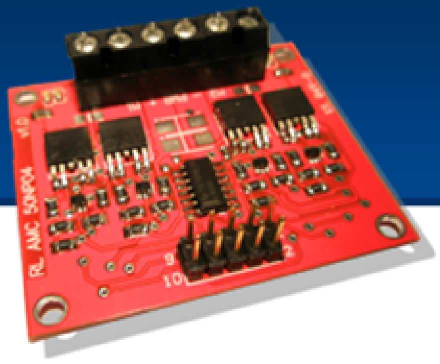
```
}

void set_motor_right(unsigned int direction, unsigned int speed)
{
    if(direction)
        set_speed_motor_right(speed);
    else
        set_speed_motor_right(-speed);
}


void set_motor_left(unsigned int direction, unsigned int speed)
{
{
    if(direction)
        set_speed_motor_left(speed);
    else
        set_speed_motor_left(-speed);
}

}

void brake_motor_right(void)
{
    set_pwm5_duty(0);
    IOCLR0 |= MOTOR_RIGHT_P2;
    IOCLR0 |= MOTOR_RIGHT_P1;
    set_pwm5_duty(2048);     // PWM high
}

void brake_motor_left(void)
{
    set_pwm2_duty(0);
    IOCLR0 |= MOTOR_LEFT_P2;
    IOCLR0 |= MOTOR_LEFT_P1;
    set_pwm2_duty(2048);     // PWM high

}
```



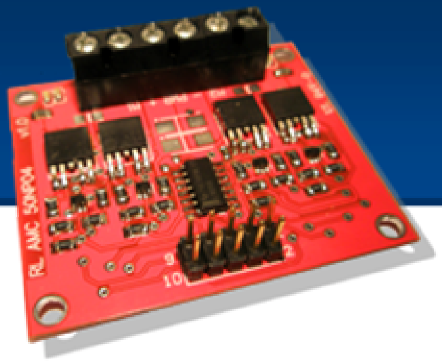
```
//--------------- pwm.c ---------------


#include <LPC213x.H>            /* LPC21xx definitions
#include "pwm.h"

void pwm_init()
{
    // Config PWM channels 2 & 5
    PINSEL0 |= (1<<15);           /* PWM2                                              */
    PINSEL1 |= (1<<10);           /* PWM5                                              */
    PWMPR    = 0x00000000;        /* Load Prescaler                                    */


    PWMPCR = 0x00002400;          /* PWM channels 2 and 5 single edged, output enabled */
    PWMMCR = 0x00000002;          /* On match with timer reset the counter             */
    PWMMR0 = 0x00000800;          /* set cycle rate                                    */
    PWMMR2 = 0x00000000;
    WMMR5 = 0x00000000;
    WMLER = 0x00000025;           /* enable shadow latch for match 0, 2 & 5            */
    PWMTCR = 0x00000002;          /* Reset counter and prescaler                       */
}


void pwm_start()
{
    PWMTCR = 0x00000002;          /* Reset counter and prescaler            */
    PWMTCR = 0x00000009;          /* enable counter and PWM, release counter from reset */
}

void pwm_stop()
{
    PWMTCR = 0x00000000;               /* Disable counter and prescaler         */
}


void set_pwm2_duty(unsigned int value)  /* value must be between 0 and 2048      */
{
    PWMMR2=value;
    PWMLER |= 0x0000004;                /* enable shadow latch for match 0 & 3   */
}

void set_pwm5_duty(unsigned int value)  /* value must be between 0 and 2048      */
{
    PWMMR5=value;
    PWMLER |= 0x0000020;                /* enable shadow latch for match 0 & 3   */

}
```



```
//--------------- serial.c ---------------

//////////////////////////////////////////////////////////
// Filename: Serial.c
// Authors:  Daniel Alvarez / Alberto Calvo
// Date:     Sat Aug 18 12:10:03 EST 2007
//
// Description: This file contains the implementation of
//                the low level serial port routines
//
//
//////////////////////////////////////////////////////////

#include <LPC213x.H>                              /* LPC21xx definitions   */

//#define USB

#define SEL_RXD0    (1<<2)
#define SEL_TXD0    (1<<0)

void init_serial (void)  {                        /* Initialize Serial Interface        */

  PINSEL0 = 0x00050000 | SEL_RXD0 | SEL_TXD0;    /* Enable RxD1 and TxD1               */
  U0LCR = 0x83;                                   /* 8 bits, no Parity, 1 Stop bit      */
  U0DLL = 0x08;                                   /* 115200 Baud Rate @ 15MHz VPB Clock */
  U0LCR = 0x03;                                   /* DLAB = 0                           */
  U1LCR = 0x83;                                   /* 8 bits, no Parity, 1 Stop bit      */
  U1DLL = 0x08;                                   /* 115200 Baud Rate @ 15MHz VPB Clock */
  U1LCR = 0x03;                                   /* DLAB = 0                           */

}


int serial_putchar (int ch)  {                    /* Write character to Serial Port     */

  while (!(U0LSR & 0x20));
  return (U0THR = ch);

}

int serial_putbuffer(char *buffer, int length)
{
 int i;
 for(i=0;i<length;i++)
    serial_putchar(buffer[i]);
 return length;
}


int serial_getkey (void)  {                      /* Read character from Serial Port */

  while( !(U0LSR & 0x01) );
    return U0RBR;

}

int serial_getbuffer(char *buffer, int length)
{
 int i=0;
 while(i<length)
    buffer[i++]=serial_getkey();
 return i;
}



int usb_putchar (int ch)  {                    /* Write character to Serial Port */

  while (!(U1LSR & 0x20));
  return (U1THR = ch);
```



```
}

int usb_putbuffer(char *buffer, int length)
{
 int i;
 for(i=0;i<length;i++)
    usb_putchar(buffer[i]);
 return length;
}


int usb_getkey (void)  {                    /* Read character from Serial Port */


  while( !(U1LSR & 0x01) );

    return (U1RBR);



}

int usb_getbuffer(char *buffer, int length)
{
 int i=0;
 while(i<length)
    buffer[i++]=usb_getkey();
 return i;
}
```



```
//--------------- serial.h ---------------

///////////////////////////////////////////////////////////
// Filename: Serial.h
// Authors:  Daniel Alvarez / Alberto Calvo
// Date:     Sat Aug 18 12:10:03 EST 2007
//
// Description: Header file for low level serial routines
//
//
//
//////////////////////////////////////////////////////////

#ifndef _SERIAL_H_
#define _SERIAL_H_

void init_serial (void);
int serial_putchar (int ch);
int serial_putbuffer(char *buffer, int length);
int serial_getkey (void);
int serial_getbuffer(char *buffer, int length);

int usb_putchar (int ch);
int usb_putbuffer(char *buffer, int length);
int usb_getkey (void);
int usb_getbuffer(char *buffer, int length);

#endif

//--------------- motor.h ---------------

#ifndef _MOTOR_H
#define _MOTOR_H

void motor_init(void);


void set_motor_right(unsigned int direction, unsigned int speed);
void set_motor_left(unsigned int direction, unsigned int speed);
void set_speed_motor_left(int speed);
void set_speed_motor_right(int speed);
void brake_motor_left(void);
void brake_motor_right(void);

#endif

//--------------- pwm.h ---------------

#ifndef _PWM_H
#define _PWM_H

void pwm_init(void);
void pwm_start(void);
void pwm_stop(void);
void set_pwm2_duty(unsigned int value); /* value must be between 0 and 2048 */
void set_pwm5_duty(unsigned int value); /* value must be between 0 and 2048 */


#endif
```