Program to operate wiper and washer devices in a car through CAN I/P or Switch

```
/*Program to operate wiper and washer devices in a car through CAN I/P or Switch */

// PIC18F458 Configuration Bit Settings
// 'C' source line configure statements
// CONFIG1H
#pragma config OSC = HS         // Oscillator Selection bits (HS oscillator)
#pragma config OSCS = OFF       // Oscillator System Clock Switch Enable bit (Oscillator system clock switch
option is disabled (main oscillator is source))

// CONFIG2L
#pragma config PWRT = OFF       // Power-up Timer Enable bit (PWRT disabled)
#pragma config BOR = OFF        // Brown-out Reset Enable bit (Brown-out Reset disabled)
#pragma config BORV = 25        // Brown-out Reset Voltage bits (VBOR set to 2.5V)

// CONFIG2H
#pragma config WDT = OFF        // Watchdog Timer Enable bit (WDT disabled (control is placed on the
SWDTEN bit))
#pragma config WDTPS = 128      // Watchdog Timer Postscale Select bits (1:128)
```

```
// CONFIG4L
#pragma config STVR = OFF       // Stack Full/Underflow Reset Enable bit (Stack Full/Underflow will not
cause Reset)
#pragma config LVP = OFF        // Low-Voltage ICSP Enable bit (Low-Voltage ICSP disabled)

// CONFIG5L
#pragma config CP0 = OFF        // Code Protection bit (Block 0 (000200-001FFFh) not code protected)
#pragma config CP1 = OFF        // Code Protection bit (Block 1 (002000-003FFFh) not code protected)
#pragma config CP2 = OFF        // Code Protection bit (Block 2 (004000-005FFFh) not code protected)
#pragma config CP3 = OFF        // Code Protection bit (Block 3 (006000-007FFFh) not code protected)

// CONFIG5H
#pragma config CPB = OFF        // Boot Block Code Protection bit (Boot Block (000000-0001FFh) not code
protected)
#pragma config CPD = OFF        // Data EEPROM Code Protection bit (Data EEPROM not code protected)

// CONFIG6L
#pragma config WRT0 = OFF       // Write Protection bit (Block 0 (000200-001FFFh) not write protected)
#pragma config WRT1 = OFF       // Write Protection bit (Block 1 (002000-003FFFh) not write protected)
```

```
#pragma config WRT2 = OFF       // Write Protection bit (Block 2 (004000-005FFFh) not write protected)
#pragma config WRT3 = OFF       // Write Protection bit (Block 3 (006000-007FFFh) not write protected)

// CONFIG6H
#pragma config WRTC = OFF       // Configuration Register Write Protection bit (Configuration registers
(300000-3000FFh) not write protected)
#pragma config WRTB = OFF       // Boot Block Write Protection bit (Boot Block (000000-0001FFh) not write
protected)
#pragma config WRTD = OFF       // Data EEPROM Write Protection bit (Data EEPROM not write protected)

// CONFIG7L
#pragma config EBTR0 = OFF      // Table Read Protection bit (Block 0 (000200-001FFFh) not protected from
Table Reads executed in other blocks)
#pragma config EBTR1 = OFF      // Table Read Protection bit (Block 1 (002000-003FFFh) not protected from
Table Reads executed in other blocks)
#pragma config EBTR2 = OFF      // Table Read Protection bit (Block 2 (004000-005FFFh) not protected from
Table Reads executed in other blocks)
#pragma config EBTR3 = OFF      // Table Read Protection bit (Block 3 (006000-007FFFh) not protected from
Table Reads executed in other blocks)
```

```
// CONFIG7H
#pragma config EBTRB = OFF      // Boot Block Table Read Protection bit (Boot Block (000000-0001FFh) not
protected from Table Reads executed in other blocks)

#include <xc.h>
#define _XTAL_FREQ 20000000    //define the crstal oscillator frequency
int buff0[11];                   //declared an array to store the data received from can bus.
void sys_init()                  //defined a function to enable the Global, peripheral interrupts and also enabled
internal pull up resistor
{
   GIE=1;                       //global interrupt enabled
   PEIE=1;                      //peripheral interrupt enabled
   RBPU=0;                      //enabled the internal pull up resistor
}
void can_init()        //CAN initialization function
{
   TRISBbits.RB3=1;       //RB3 - CAN_RX
   TRISBbits.RB2=0;       //RB2 - CAN_TX
   TRISBbits.RB0=1;       //RB0 for switch as input
   PIE3bits.RXB0IE=1;      //peripheral interrupt enable flag bit
```

```
    IPR3bits.RXB0IP=1;      //peripheral interrupt priority bit
}
void set_baud()             //this function is used set the baud rate.
{
    CANCON=0x80;            //for setting baud rate change the mode of operation in to configure mode.
    while(CANSTAT!=0x80);
    BRGCON1 = 0XC1;
    BRGCON2 = 0XAE;
    BRGCON3 = 0X45;
    CANCON = 0x0E;          //after setting baud rate set the mode of operation to normal mode.
}
void mask_filter()          //defined this function to set mask and filter to receive all the data
{
    CANCON=0x0E;
    RXM0SIDH=0X00;
    RXM0SIDL=0X00;
    RXF0SIDH=0X00;
    RXF0SIDL=0X00;
    RXB0CON=0x00;
}
```

```
void can_read()          //defined a function store the RX received
{
   CANCON=0x0E;          //Configured to normal mode(RX)
   buff0[0]=RXB0D0;
   buff0[1]=RXB0D1;
   buff0[2]=RXB0D2;
   buff0[3]=RXB0D3;           //rain sensor status input for wiper
   buff0[4]=RXB0D4;           //dust sensor
   buff0[5]=RXB0D5;           //fluid tank status
   buff0[6]=RXB0D6;
   buff0[7]=RXB0D7;
   buff0[8]=RXB0DLC;
   buff0[9]=RXB0SIDL;
   buff0[10]=RXB0SIDH;
   RXB0CONbits.RXFUL=0;
}
void write_wiper()       //CAN O/P for Wiper with ID 0x11
{
   CANCON=0x08;
   TXB0SIDH=0x02;
```

```
    TXB0SIDL=0x20;
    TXB0D0=0x11;
    TXB0D1=0x11;
    TXB0D2=0x11;
    TXB0D3=0x11;
    TXB0D4=0x11;
    TXB0D5=0x11;
    TXB0D6=0x11;
    TXB0D7=0x11;
    TXB0CON=0x08;
    CANCON=0x08;
}
void write_washer()             //CAN O/P for Washer with ID 0x22
{
    CANCON=0x08;
    TXB0SIDH=0x04;
    TXB0SIDL=0x40;
    TXB0D0=0x22;
    TXB0D1=0x22;
    TXB0D2=0x22;
    TXB0D3=0x22;
    TXB0D4=0x22;
    TXB0D5=0x22;
    TXB0D6=0x22;
    TXB0D7=0x22;
    TXB0CON=0x08;
    CANCON=0x08;
}

void __interrupt() ISR1()
{
    if(PIR3bits.RXB0IF)         //if interrupt occurred
    {
       PIR3bits.RXB0IF=0;       //disable interrupt
        can_read();             //RX function is called
    }
}

void handshake()            //in order to give acknowledgement to can bus transmitting a frame of data with 0x03
{
    CANCON=0x08;            //set operation mode to normal mode
    TXB0SIDH=0x00;          //MSG ID 0x03
    TXB0SIDL=0x60;
    TXB0D0=0x03;
    TXB0D1=0x03;
    TXB0D2=0x03;
    TXB0D3=0x03;
    TXB0D4=0x03;
    TXB0D5=0x03;
    TXB0D6=0x03;
    TXB0D7=0x03;
    TXB0CON=0x08;       //enabling TX buff control register with 0x08
    CANCON=0x08;        //after TX, setting operation to normal mode
}

void main()        //start main()
{
    TRISC=0x00;     //making port c as output port
    PORTC=0x00;
    sys_init();     //calling system initialization function
    can_init();     //calling CAN initialization function
    set_baud();     //calling set baud rate function to set baud rate
    mask_filter();  //call mask filter function set mask and filter
    int flag=0;
    int flag1=0;
    int flag2=0;
    int flag3=0;
    int flag4=0;
    while(1)
    {
      handshake();          //call handshake() to indicate proper communication in bus
      __delay_ms(500);
      if(buff0[9]==0x00 && buff0[10]==0xE0)      //if data frame is for 0x700 ID then below will be executed
       {
          if((buff0[3]==0x00) && ((buff0[4]>=0x0B) && (buff0[4]<=0xFF)))     //if there is no rain and if there
is dust on windscreen the flag is set to 1
          {
             flag=1;
             flag1=0;
             flag2=0;
```

```
            flag3=0;
            flag4=0;
        }
        else if((flag==0)&& (buff0[5]<0x0B))            //if there is dust but the tank is empty
        {
            flag=0;
            flag1=0;
            flag2=0;
            flag3=0;
            flag4=1;
        }
        else if(buff0[3]>=0x01 && buff0[3]<=0x0A)            //in case of low rain wiper is activated then led will
glow with 2sec delay.
         {
            flag=0;
            flag1=0;
            flag2=1;
            flag3=0;
            flag4=0;
         }
```

```
            else if(buff0[3]>=0x0B && buff0[3]<=0xF0)      //in case of moderate rain wiper is activated then led
will glow with 0.5delay
            {
               flag=0;
               flag1=0;
               flag2=0;
               flag3=1;
               flag4=0;
            }
            else if(buff0[3]>0xF0 && buff0[3]<=0xFF)              //heavy rain
            {
               flag1=1;
               flag=0;
               flag2=0;
               flag3=0;
               flag4=0;
            }
            else
            {
               flag1=0;
               flag=0;
               flag2=0;
               flag3=0;
               flag4=0;
            }
          }
         if(flag==1) //if flag is 1 dust_sensor() is called for activating washer
         {

            write_washer();   //incase of washer activated this function gives can o/p with 0x22 id
            PORTCbits.RC0=1;          //fluid tank is more tan 50% filled then it's status will be indicated by led1
            __delay_ms(500);
            PORTCbits.RC1=1;          //washer will be activated(indicated by led2) if fluid tank is not empty
            __delay_ms(500);
            PORTCbits.RC5=1;          //after D0 device split water on wind screen it must be wiped, the wiper
functionality will be indicated by led1
            __delay_ms(500);
            PORTCbits.RC0=0;          //led1 is off
            PORTCbits.RC1=0;          //led2 is off
            PORTCbits.RC5=0;          //led3 is off
```

```
     __delay_ms(500);

}
if(flag4==1)
{
     PORTCbits.RC7=1;            //fluid tank is less tan 50% filled then it's status will be indicated by led1
     __delay_ms(100);
     PORTCbits.RC7=0;            //fluid tank is less tan 50% filled then it's status will be indicated by led1
     __delay_ms(100);
}

if(flag2==1)     //if flag1 is 1 wiper() is called for activating washer
{

 if(PORTBbits.RB0==0)      //press switch1 in order to control the wipers
 {
 write_wiper();                           //in case of rain occurred a frame of data with 0x11 as ID
 RC5=1;                                  //led3 will glow as indicator for front wiper
 RC6=1;                                  //led4 will glow as indicator for front wiper
 __delay_ms(500);
 RC5=0;                              //led3 is off
 RC6=0;                              //led4 is off
 __delay_ms(500);

    }
}
if(flag3==1)
 {

    if(PORTBbits.RB0==0)      //press switch1 in order to control the wipers
     {
     write_wiper();                          //in case of rain occurred a frame of data with 0x11 as ID
     RC5=1;                            //led3 will glow as indicator for front wiper
     RC6=1;                            //led4 will glow as indicator for front wiper
     __delay_ms(300);
     RC5=0;                            //led3 is off
     RC6=0;                            //led4 is off
     __delay_ms(300);

     }
```

```
         }
        if(flag1==1)
        {
          if(PORTBbits.RB0==0)      //press switch1 in order to control the wipers
           {
           write_wiper();                              //in case of rain occurred a frame of data with 0x11 as ID
           RC5=1;                                    //led3 will glow as indicator for front wiper
           RC6=1;                                    //led4 will glow as indicator for front wiper
           __delay_ms(100);
           RC5=0;                                    //led3 is off
           RC6=0;                                    //led4 is off
           __delay_ms(100);

           }
        }

      }
   }
```