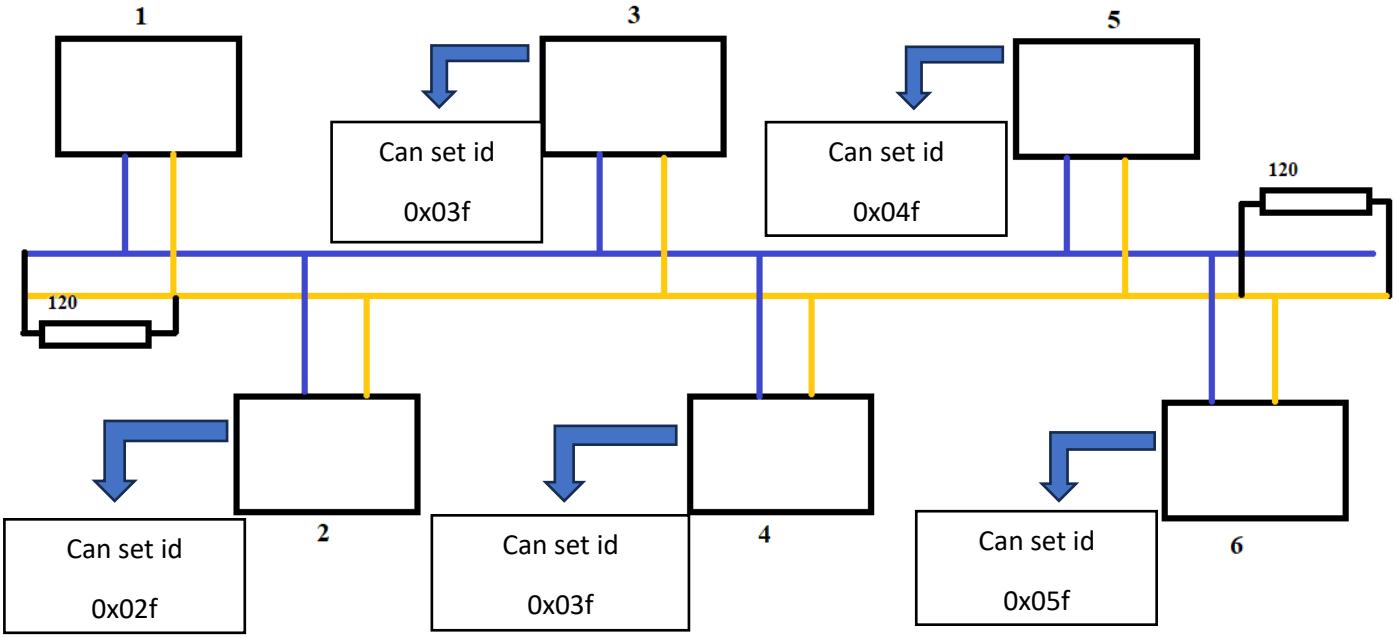
ข้อมูลเพื่อใช้ประกอบการฝึกอบรมพอสังเขป




สรุปหน้าที่การทำงานของแต่ละ NODE

-NODE 1

รับข้อมูล อุณหภูมิมอเตอร์ อุณหภูมิชุดขับเคลื่อนมอเตอร์ จาก NODE 4

รับข้อมูล สัญญานคันเร่ง ความเร็วรอบมอเตอร์ จาก NODE 5

รับข้อมูล แรงดันแบตเตอรี่ ค่ากระแสมอเตอร์ จาก NODE 6

ส่งข้อมูล การควบคุมความเร็วมอเตอร์ให้กับ NODE 5

ส่งข้อมูล การควบคุมทิศทางมอเตอร์ให้กับ NODE 6

NODE 2

NODE 3

NODE 4

ส่งข้อมูล อุณหภูมิมอเตอร์ อุณหภูมิชุดขับเคลื่อนมอเตอร์ ให้ NODE 1

NODE 5

ส่งข้อมูล สันญานคันเร่ง และ ความเร็วมอเตอร์ให้กับ NODE 1

รับข้อมูลโหมดการควบคุมจาก NODE 1

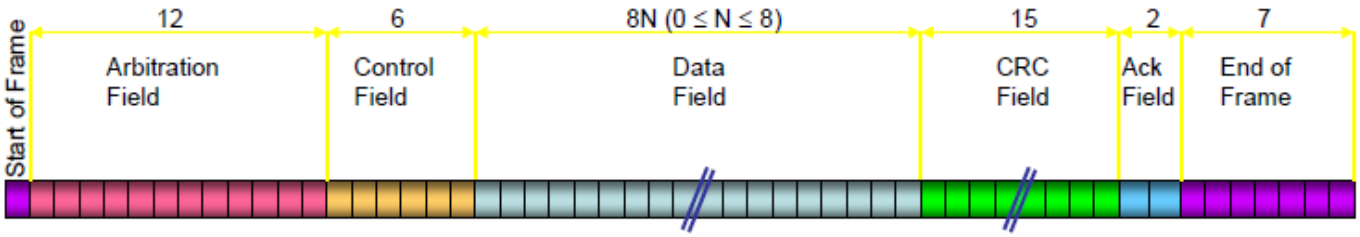
NODE 6

ส่งข้อมูลแรงดันแบตเตอรี่ และ ค่ากระแสมอเตอร์ให้กับ NODE 1

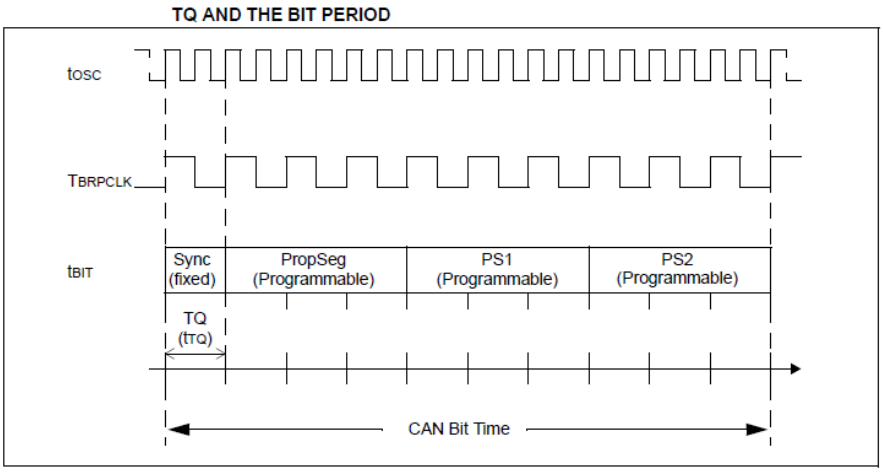
รับข้อมูลการควบคุมทิศทางมอเตอร์จาก NODE 1

**ใช้รูปแบบ Standard Data Frame ในการรับส่งข้อมูล**

- Versions 1.0 and 2.0A
- 11-bit Identifier Field



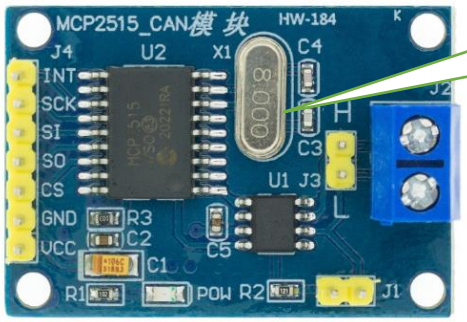
ใช้ความเร็วในการรับส่งข้อมูล(Baue rate) 62.5Kbit/sec




Bit time period

การคำณวนค่าความเร็วในการสื่อสาร

Tq = (2 x (PRE + 1))/Fosc )




Fosc

8Mhz

Tq = (2x(3+1))/8000000

= 1usec

#define CAN_USE_EXTENDED_ID false

#define CAN_BRG_SYNCH_JUMP_WIDTH 0 **//synchronized jump width (def: 1 x Tq)**

#define CAN_BRG_PRESCALAR 3

Prescale=3

#define CAN_BRG_SAM 0

#define CAN_BRG_PHASE_SEGMENT_1 5 **//phase segment 1 (def: 6 x Tq)**

#define CAN_BRG_PROPAGATION_TIME 2 **//propagation time select (def: 3 x Tq)**

#define CAN_BRG_WAKE_FILTER FALSE

```
#define CAN_BRG_PHASE_SEGMENT_2  5 //phase segment 2 time select (def: 6 x Tq)

#define CAN_USE_RX_DOUBLE_BUFFER TRUE

#define CAN_ENABLE_DRIVE_HIGH 0

#define CAN_ENABLE_CAN_CAPTURE 0
```

จาก source code เวลาใน 1 bit = 1+6+3+6 = 16 Tq = 16 usec

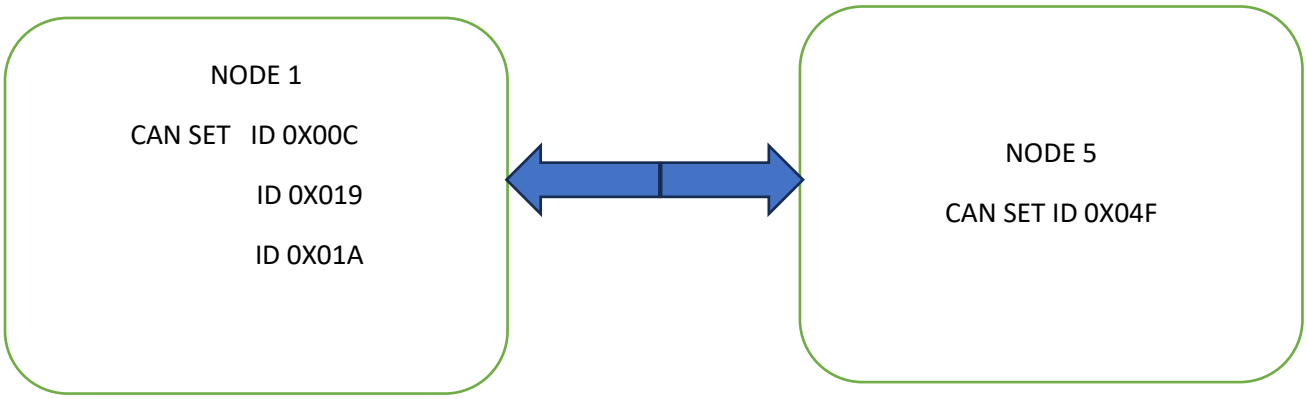
ดังนั้นจะได้ความเร็วในการสื่อสาร(Baue rate)

= 1/16 usec

= 1/0.000016

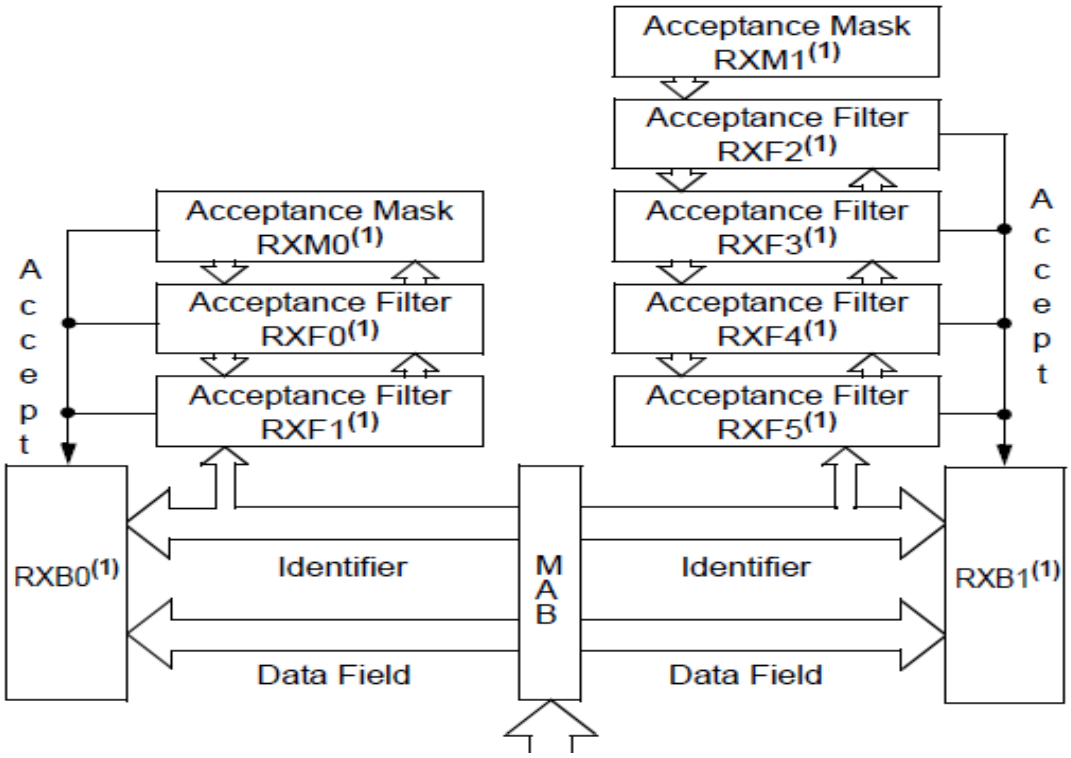
= 62500 bit/sec

ตัวอย่างการรับสังข้อมูลระหว่าง NODE 1 กับ NODE 5




จากรูปข้างบนกล่าวได้ว่ากำหนดให้ NODE 1 รับข้อมูลรับข้อมูลจาก 3 ID คือ 0X00C,0X019 และ 0X01A ซึ่งมาจากกา

การกำหนดค่า หนด ให้กับส่วน ของ register mask และ register filter




Source code การกำหนดค่าให้กับ register mask และ register filter ให้กับ NODE 1 เพื่อให้รับข้อมูลจาก ID

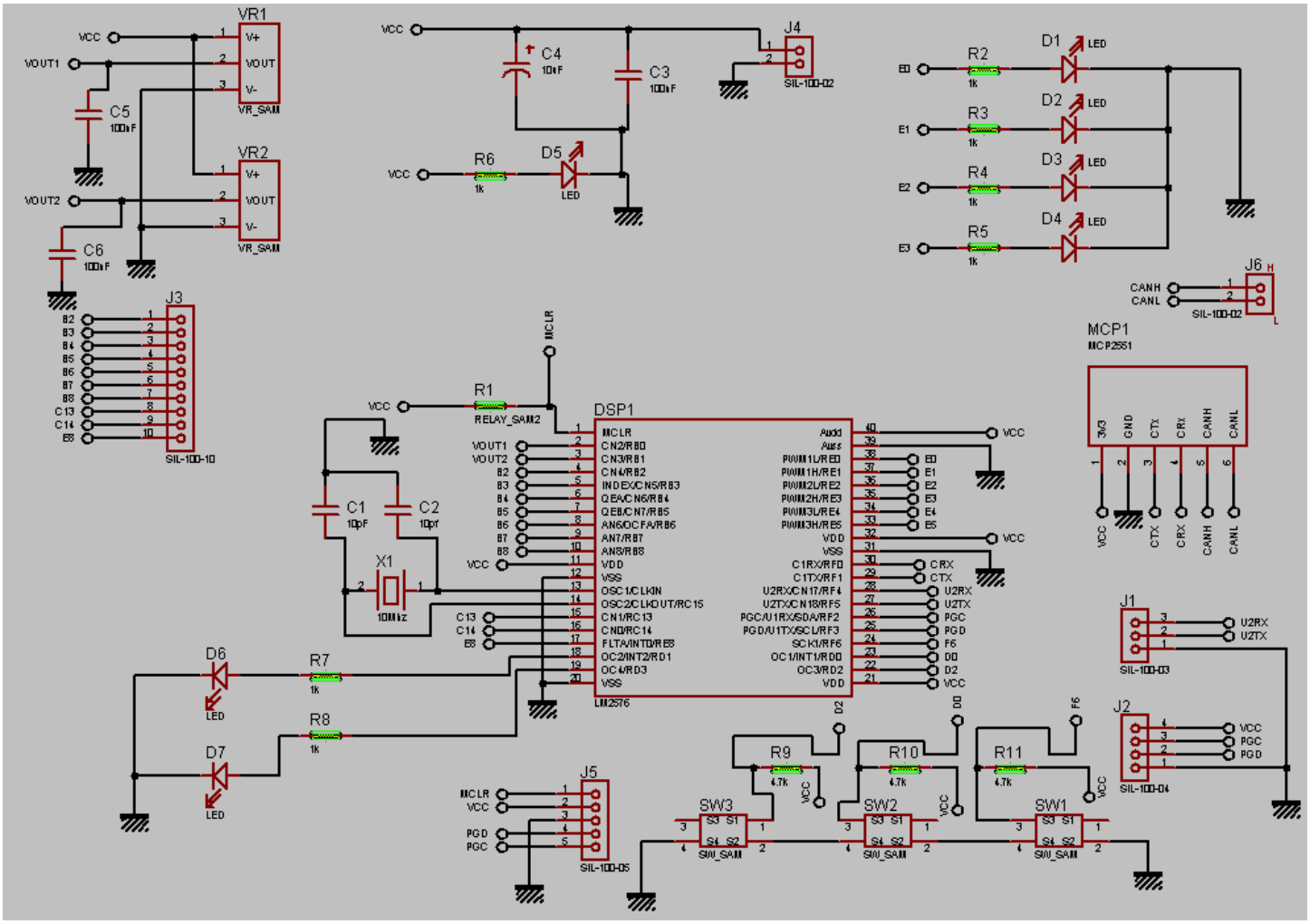
0X00C,0X019 และ 0X01A

```
can_set_id(&C1RXM0, 0B11111111110, 0);  //set mask 0
 can_set_id(&C1RXF0, 0B00000011010, 0);  //set filter 0 of mask 0
 can_set_id(&C1RXF1, 0B00000001100, 0);  //set filter 1 of mask 0
 can_set_id(&C1RXM1, 0B11111111111, 0);  //set mask 1
 can_set_id(&C1RXF2, 0B00000011001, 0);  //set filter 0 of mask 1
 can_set_id(&C1RXF3, 0B00000011001, 0);  //set filter 1 of mask 1
 can_set_id(&C1RXF4, 0B00000011001, 0);  //set filter 2 of mask 1
 can_set_id(&C1RXF5, 0B00000011001, 0);  //set filter 3 of mask 1
```

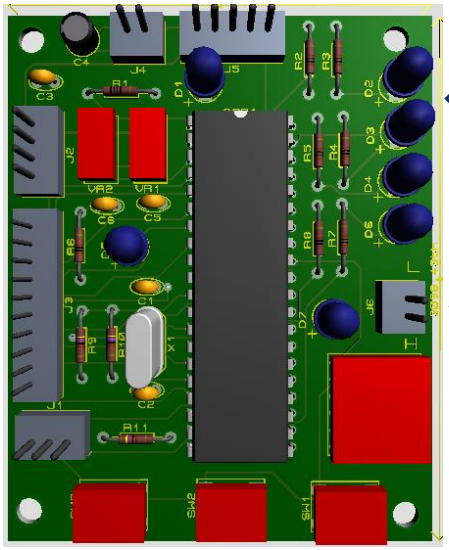
source code การกำหนดค่าให้กับ register mask และ register filter ให้กับ NODE 5 เพื่อให้รับข้อมูลเฉพาะ ID0X04F

```
 can_set_id(RX0MASK, 0xfff, 0);  //set mask 0 (RX BUFFER 0)
 can_set_id(RX0FILTER0, 0x04f, 0);  //set filter 0 of mask 0 (RX BUFFER 0)
 can_set_id(RX0FILTER1, 0x04f, 0);  //set filter 1 of mask 0 (RX BUFFER 0)
 can_set_id(RX1MASK, 0xfff, 0);  //set mask 1 (RX BUFFER 1)
 can_set_id(RX1FILTER2, 0x04f, 0);  //set filter 0 of mask 1 (RX BUFFER 1)
 can_set_id(RX1FILTER3, 0x04f, 0);  //set filter 1 of mask 1 (RX BUFFER 1)
 can_set_id(RX1FILTER4, 0x04f, 0);  //set filter 2 of mask 1 (RX BUFFER 1)
 can_set_id(RX1FILTER5, 0x04f, 0);  //set filter 3 of mask 1 (RX BUFFER 1)
```

รายละเอียดวงจร NODE 1




Touch Screen Display



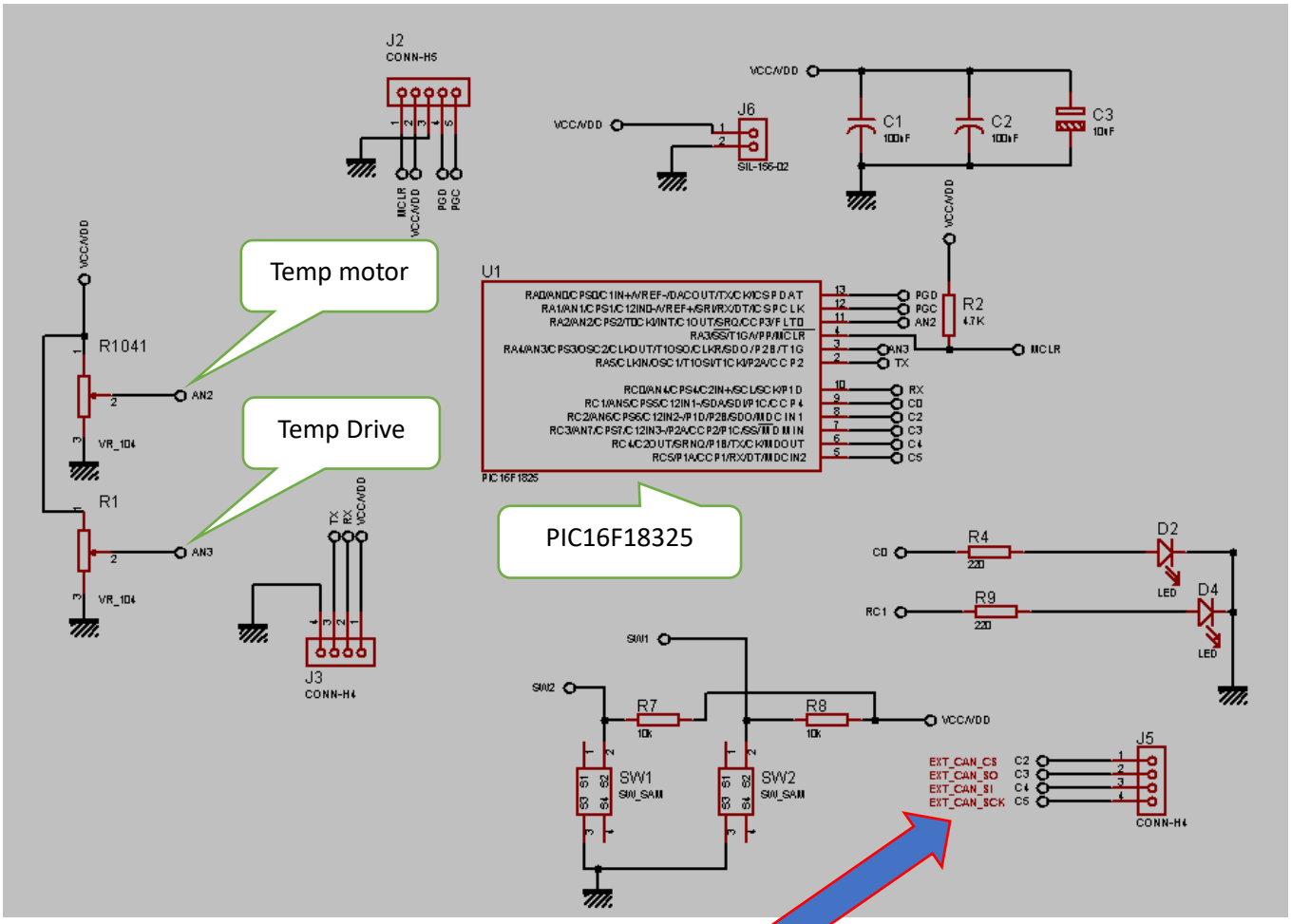
Rs232(2)

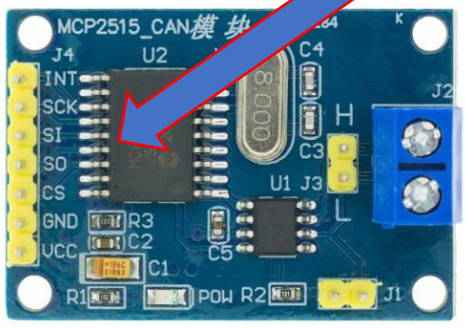
CAN BUS


DSPIC30F4011

NEXTION DISPLAY

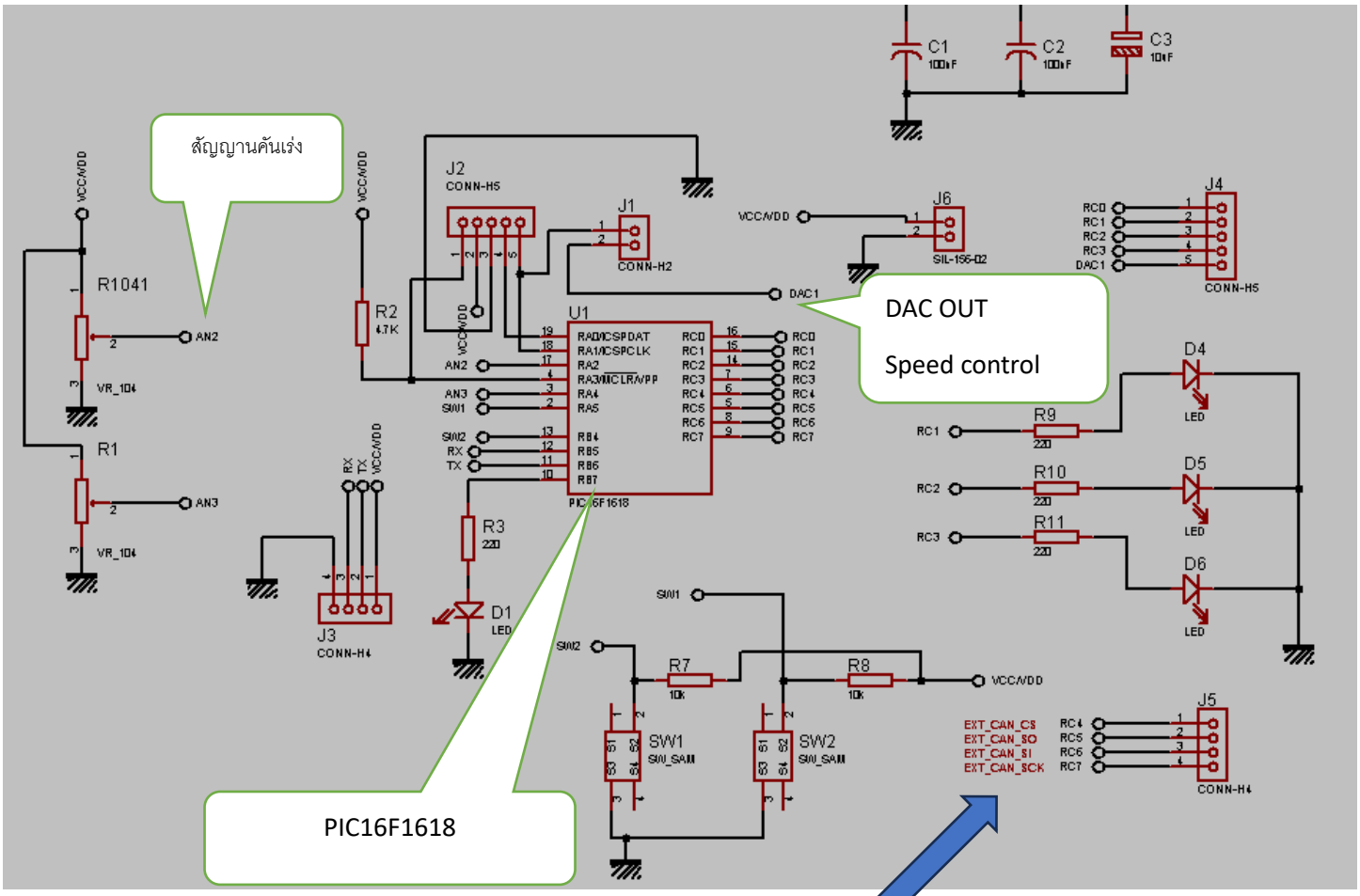
รายละเอียดวงจร NODE 2,3,4,6

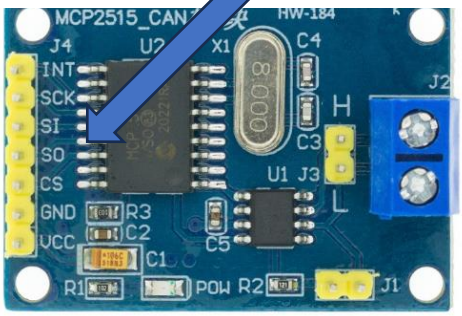



CAN BUS

รายละเอียดวงจร NODE 5




CAN BUS

Source code NODE 1 ประกอบด้วย 4 file คือ

1 Node_c.c

```
#include "30F4011.h"
#device ICSP=1
#device PASS_STRINGS = IN_RAM
#include<string.h>
#include<stdio.h>
#include<stdlib.h>
#priority RDA2
#FUSES HS
#FUSES NOBROWNOUT              //No brownout reset
#use delay(crystal=10000000)
//#use delay(CLOCK=20MHZ)
#use rs232(UART1, baud=9600,PARITY=n,BITS =8,STOP=1, stream=PORT1)
#use rs232(xmit=PIN_f5, rcv=PIN_f4, baud=9600,PARITY=n,BITS =8,STOP=1, stream=PORT2)
//#define CAN_USE_EXTENDED_ID   false
#define CAN_USE_EXTENDED_ID        false
 #define CAN_BRG_SYNCH_JUMP_WIDTH  0
 #define CAN_BRG_PRESCALAR  4
 #define CAN_BRG_SAM 0
 #define CAN_BRG_PHASE_SEGMENT_1  5
 #define CAN_BRG_PROPAGATION_TIME 2
 #define CAN_BRG_WAKE_FILTER FALSE
 #define CAN_BRG_PHASE_SEGMENT_2 5
 #define CAN_USE_RX_DOUBLE_BUFFER TRUE
 #define CAN_ENABLE_DRIVE_HIGH 0
  #define CAN_ENABLE_CAN_CAPTURE 0
   #include "can-dsPIC30.c"
   struct rx_stat rxstat;
   int32 rx_id;
  unsigned int8 in_data[8];
   int8 rx_len;
   int8 out_data[8];
   int32 tx_id=27;
   int1 tx_rtr=0;
   int1 tx_ext=0;
   int8 tx_len=8;
   int8 tx_pri=1;
   char str[16];
   char str2[15];
    char str3[15];
    char ccc[10]={'0','1','2','3','4','5','6','7','8','9'};
   char send_display[16];
   int8 i;
  // int8 out_data[8];
 unsigned  int8 buffer[8];
   char a,b,c;
   int8 num0,num1,num2;
     int8 ms=0;
    unsigned int16 *ptr;
   unsigned int8 speed1=0;
     int8 speed2;
```

```
//////////////////////////////VAR NEXTION ////////////
signed long result;
char *ptrx;
   char data_ser[30];
   signed int8 index_byte=0;
   int1 flag_rx=0;
   int1 flg_disable_rda=0;
   int1 flag_rda=0;
   int1 flag_page_display=0;
   int1 flag_obd=0;
   int1 flag_motor=0;
   int8 k=0;
  VOID SENDSTRING(CHAR *TXT,int8 num_txt);
  VOID SENDSTRING_display(CHAR *TXT,int8 num_txt);
   VOID SENDSTRING_obd(CHAR *TXT,int8 num_txt);
     VOID SENDSTRING_motor(CHAR *TXT,int8 num_txt);
  void process_display();
   void process_obd();
    void process_motor(void);
   #include "instrument.c"
   #include "check.c"
   #include "motor.c"
   void process_command();
    void process_command2();
    void page_display(void);
  #include "monitor.c"
#INT_TIMER2
void  timer2_isr(void)
{
 ms++;//output_toggle(pin_e3);
}
#INT_RDA2
void  rda2_isr(void)
{
//////////////////////////////////////////get data from nextion display//////////////////////

          flag_rda=1;

          data_ser[index_byte] =fgetc(port2);//output_toggle(pin_e8);
           if (( index_byte>3) && ((unsigned char)data_ser[index_byte]==0xff) && ((unsigned char)data_ser[index_byte-
1]==0xff)&&((unsigned char)data_ser[index_byte-2]==0xff) )
          {
           flag_rx=1;//data serial ok
           disable_interrupts(INT_RDA2);//flg_disable_rda=1;
           // fprintf(port1,data_ser);
          // process_command();
          strcpy (str2, data_ser);
          }
     index_byte++;
    if(index_byte>15)
    {
     index_byte=0;enable_interrupts(INT_RDA2);flg_disable_rda=0;

     flag_rx=0;memset(data_ser,0,30);

    }

}
```

```
void main()
{
    can_init();
   // setup_timer2(TMR_INTERNAL | TMR_DIV_BY_1, 5000);//over flow 2msec
     disable_interrupts(INT_TIMER2);
    // enable_interrupts(INT_RDA2);
     enable_interrupts(INTR_GLOBAL);
      DELAY_ms(1000);
     buffer[0] = 10;      buffer[1] = 20;
     buffer[2] = 30;      buffer[3] = 40;
     buffer[4] = 50;      buffer[5] = 60;
     buffer[6] = 70;      buffer[7] = 80;
     i=0;
  for (i=0;i<8;i++) {
   output_low(pin_e2);output_low(pin_e1);output_low(pin_e0);
   delay_ms(500);
   output_high(pin_e2);output_high(pin_e1);output_high(pin_e0);
   delay_ms(500);
   }
   memset(data_ser,0,30);enable_interrupts(INT_RDA2);
output_low(pin_e2);output_low(pin_e8);output_low(pin_e4);
   while(true)
   {
       if(flag_rx)
      {flag_rda=0;
       process_command();// page_display();
       flag_rx=0;index_byte=0; memset(data_ser,0,30); enable_interrupts(INT_RDA2);
      }
       output_toggle(pin_e3);delay_ms(500);

   }

}
  void process_command()
  {
     if ( strstr(data_ser,"get_id15" ))
    {
    SENDSTRING("Instrument page",3);
    delay_ms(3000);
   flag_page_display=1;page_display();
   //delay_ms(500);

    }
    else if(strstr(data_ser,"get_id16" ))
    {
    SENDSTRING("Engin page",3);
     delay_ms(3000);
   flag_motor=1;page_motor();
    }
    else if(strstr(data_ser,"get_id27" ))
    {
    SENDSTRING("obdpage",3);
     delay_ms(3000);
   flag_obd=1;page_obd();
    }
    else{flag_rx=0;index_byte=0; memset(data_ser,0,30);flag_rda=0; enable_interrupts(INT_RDA2);}
  }
```

## 2 Motor.c

```
void page_motor(void)
{
can_set_mode(CAN_OP_CONFIG);   //must be in config mode before params can be set
can_set_baud();

C1CTRL.cancks=CAN_CANCKS;

ptr = &C1RX0CON;
//C1RX0CON=0;
*ptr = 0;
C1RX0CON.dben=CAN_USE_RX_DOUBLE_BUFFER;

C1CTRL.cancap=CAN_ENABLE_CAN_CAPTURE;

can_set_id(&C1RXM0, 0B11111111110, 0);  //set mask 0
can_set_id(&C1RXF0, 0B00000011010, 0);  //set filter 0 of mask 0
can_set_id(&C1RXF1, 0B00000001100, 0);  //set filter 1 of mask 0

can_set_id(&C1RXM1, 0B11111111111, 0);  //set mask 1
can_set_id(&C1RXF2, 0B00000011001, 0);  //set filter 0 of mask 1
can_set_id(&C1RXF3, 0B00000011001, 0);  //set filter 1 of mask 1
can_set_id(&C1RXF4, 0B00000011001, 0);  //set filter 2 of mask 1
can_set_id(&C1RXF5, 0B00000011001, 0);  //set filter 3 of mask 1

can_set_mode(CAN_OP_NORMAL);
  while(flag_motor){


    delay_ms(2);
   if(flag_rx)
   {flag_rda=0;
    process_motor();
    flag_rx=0;index_byte=0; memset(data_ser,0,30); enable_interrupts(INT_RDA2);

   }
  //  output_toggle(pin_e3);delay_ms(100);

  if ( can_kbhit() )   //if data is waiting in buffer...
   {
////////////////////////////////////////////////////recive massage ////////////////////////////////////
     if(can_getd(rx_id, &in_data[0], rx_len, rxstat)) { //...then get data from buffer

        sprintf(str, "0X%05LX",rx_id);
       if(!flag_rda)  SENDSTRING_motor(str,1);
       if(rx_id==0x19)
       {
       sprintf(str, "%03u",in_data[0]);//  get temp motor
       SENDSTRING_motor(str,4);// send display nextion page display t6
       sprintf(str, "%03u",in_data[1]);// get temp drive
       SENDSTRING_motor(str,5);// send display nextion page display t7
       }

       if(rx_id==0x1a)
       {
       sprintf(str, "%03u",in_data[0]);// get tps value
       SENDSTRING_motor(str,6);// send display nextion page display t8
       sprintf(str, "%03u",in_data[1]);// get rpm
       SENDSTRING_motor(str,7);//send display nextion page display t9
       }

       if(rx_id==0x0c)
       {
       sprintf(str, "%03u",in_data[0]);// get current
       SENDSTRING_motor(str,8);// send display nextion page display t10
```

```
            sprintf(str, "%03u",in_data[1]);// get vbat
            SENDSTRING_motor(str,9);//send display nextion page display t1
            }
          }
        }
        else {
          // printf("\r\nFAIL on GETD\r\n");
        }
     }

   }

  void process_motor()
  {

    if ( strstr(data_ser,"home" ))
    {

    SENDSTRING_motor("exit",3);delay_ms(3000);flag_motor=0;
    flag_rx=0;index_byte=0; memset(data_ser,0,30);flag_rda=0; SENDSTRING("Main
Page",1);enable_interrupts(INT_RDA2);
    }
    else if(strstr(data_ser,"stop" ))
    {

    buffer[0] = 0;buffer[1]=0;
    while( !can_tbe());
    can_putd(0x04f,buffer,2,tx_pri,tx_ext,tx_rtr);
    }
    else if(strstr(data_ser,"local" ))
    {

    buffer[0] = 0;buffer[1]=0;
    while( !can_tbe());
    can_putd(0x04f,buffer,2,tx_pri,tx_ext,tx_rtr);
    }
    else if(strstr(data_ser,"remote"))
    {

    buffer[0] = 1;buffer[1]=10;
    while( !can_tbe());
    can_putd(0x04f,buffer,2,tx_pri,tx_ext,tx_rtr);

    }
    else if(strstr(data_ser,"speed1"))
    {
     buffer[0] = 1;buffer[1]=90;
     while( !can_tbe());
     can_putd(0x04f,buffer,2,tx_pri,tx_ext,tx_rtr);
    }
    else if(strstr(data_ser,"speed2"))
    {
     buffer[0] = 1;buffer[1]=150;
     while( !can_tbe());
     can_putd(0x04f,buffer,2,tx_pri,tx_ext,tx_rtr);
    }
    else if(strstr(data_ser,"speed3"))
    {
     buffer[0] = 1;buffer[1]=220;
     while( !can_tbe());
     can_putd(0x04f,buffer,2,tx_pri,tx_ext,tx_rtr);
    }
    else if(strstr(data_ser,"D"))
    {
     buffer[0] = 255;//buffer[1]=220;
     while( !can_tbe());
```

```
          can_putd(0x05f,buffer,1,tx_pri,tx_ext,tx_rtr);
         }
         else if(strstr(data_ser,"R"))
         {
          buffer[0] = 0;//buffer[1]=220;
          while( !can_tbe());
          can_putd(0x05f,buffer,1,tx_pri,tx_ext,tx_rtr);
         }

         else{flag_rx=0;index_byte=0; memset(data_ser,0,30);flag_rda=0; enable_interrupts(INT_RDA2);}

       }
```

## 3 file instrument.c

```
void page_display(void)
{
  can_set_mode(CAN_OP_CONFIG);   //must be in config mode before params can be set
can_set_baud();
C1CTRL.cancks=CAN_CANCKS;
ptr = &C1RX0CON;
//C1RX0CON=0;
*ptr = 0;
C1RX0CON.dben=CAN_USE_RX_DOUBLE_BUFFER;
C1CTRL.cancap=CAN_ENABLE_CAN_CAPTURE;
can_set_id(&C1RXM0, 0B11111111110, 0);  //set mask 0
can_set_id(&C1RXF0, 0B00000011010, 0);  //set filter 0 of mask 0
can_set_id(&C1RXF1, 0B00000001100, 0);  //set filter 1 of mask 0

can_set_id(&C1RXM1, 0B11111111111, 0);  //set mask 1
can_set_id(&C1RXF2, 0B00000011001, 0);  //set filter 0 of mask 1
can_set_id(&C1RXF3, 0B00000011001, 0);  //set filter 1 of mask 1
can_set_id(&C1RXF4, 0B00000011001, 0);  //set filter 2 of mask 1
can_set_id(&C1RXF5, 0B00000011001, 0);  //set filter 3 of mask 1
can_set_mode(CAN_OP_NORMAL);
 while(flag_page_display){
 delay_ms(2);
   if(flag_rx)
   {flag_rda=0;
    process_display();
   flag_rx=0;index_byte=0; memset(data_ser,0,30); enable_interrupts(INT_RDA2);

   }
```

```
    if ( can_kbhit() )   //if data is waiting in buffer...
     {
////////////////////////////////////////////////recive massage ////////////////////////////////////
        if(can_getd(rx_id, &in_data[0], rx_len, rxstat)) { //...then get data from buffer
             sprintf(str, "0X%05LX",rx_id);
           if(!flag_rda)  SENDSTRING_display(str,1);
           if(rx_id==0x19)
           {
           sprintf(str, "%03u",in_data[0]);//  get temp motor
           SENDSTRING_display(str,4);// send display nextion page display t6
           sprintf(str, "%03u",in_data[1]);// get temp drive
           SENDSTRING_display(str,5);// send display nextion page display t7
           }
           if(rx_id==0x1a)
           {
           sprintf(str, "%03u",in_data[0]);// get tps value
           SENDSTRING_display(str,6);// send display nextion page display t8
           sprintf(str, "%03u",in_data[1]);// get rpm
           SENDSTRING_display(str,7);//send display nextion page display t9
           }
           if(rx_id==0x0c)
           {
           sprintf(str, "%03u",in_data[0]);// get current
           SENDSTRING_display(str,8);// send display nextion page display t10
           sprintf(str, "%03u",in_data[1]);// get vbat
           SENDSTRING_display(str,9);//send display nextion page display t1
           }
         }
        }
        else {
          // printf("\r\nFAIL on GETD\r\n");
        }
    }

  }

 void process_display(void)
 {
```

```
   if ( strstr(data_ser,"home" ))
   {
   SENDSTRING_display("exit",3);delay_ms(3000);flag_page_display=0;
   flag_rx=0;index_byte=0; memset(data_ser,0,30);flag_rda=0;SENDSTRING("Main Page",1);
enable_interrupts(INT_RDA2);

   }
   else{flag_rx=0;index_byte=0; memset(data_ser,0,30);flag_rda=0; enable_interrupts(INT_RDA2);}
 }
```

## 3 file monitor.c

```
 VOID SENDSTRING(CHAR *TXT,int8 num_txt)
{
  int8 num;
  num=num_txt;
    if(num==1){
          fprintf(PORT2,"home.t0.txt=");delay_ms(10);
          fprintf(PORT2,"\"");delay_ms(1);
          fprintf(PORT2,TXT);
          fprintf(PORT2,"\"");
          fprintf(PORT2,"\xFF\xFF\xFF");
    }
     if(num==2){
          fprintf(PORT2,"home.t1.txt=");delay_ms(10);
          fprintf(PORT2,"\"");delay_ms(1);
          fprintf(PORT2,TXT);
          fprintf(PORT2,"\"");
         fprintf(PORT2,"\xFF\xFF\xFF");
    }

     if(num==3){
          fprintf(PORT2,"home.t2.txt=");delay_ms(10);
          fprintf(PORT2,"\"");delay_ms(1);
          fprintf(PORT2,TXT);
          fprintf(PORT2,"\"");
          fprintf(PORT2,"\xFF\xFF\xFF");
    }

}
 VOID SENDSTRING_display(CHAR *TXT,int8 num_txt)
{
  int8 num;
  num=num_txt;
    if(num==1){
          fprintf(PORT2,"display1.t3.txt=");delay_ms(10);
          fprintf(PORT2,"\"");delay_ms(10);
          fprintf(PORT2,TXT);
          fprintf(PORT2,"\"");
          fprintf(PORT2,"\xFF\xFF\xFF");
    }
     if(num==2){
          fprintf(PORT2,"display1.t4.txt=");delay_ms(10);
          fprintf(PORT2,"\"");delay_ms(10);
          fprintf(PORT2,TXT);
          fprintf(PORT2,"\"");
         fprintf(PORT2,"\xFF\xFF\xFF");
    }
```

```
      if(num==3){
            fprintf(PORT2,"display1.t5.txt=");delay_ms(10);
            fprintf(PORT2,"\"");delay_ms(10);
            fprintf(PORT2,TXT);
            fprintf(PORT2,"\"");
            fprintf(PORT2,"\xFF\xFF\xFF");
      }
       ////////////////////////////////////////////////////////////////////////////

      if(num==4){
            fprintf(PORT2,"display1.t6.txt=");delay_ms(10);
            fprintf(PORT2,"\"");delay_ms(10);
            fprintf(PORT2,TXT);
            fprintf(PORT2,"\"");
            fprintf(PORT2,"\xFF\xFF\xFF");
      }
      if(num==5){
            fprintf(PORT2,"display1.t7.txt=");delay_ms(10);
            fprintf(PORT2,"\"");delay_ms(10);
            fprintf(PORT2,TXT);
            fprintf(PORT2,"\"");
            fprintf(PORT2,"\xFF\xFF\xFF");
      }
      if(num==6){
            fprintf(PORT2,"display1.t8.txt=");delay_ms(10);
            fprintf(PORT2,"\"");delay_ms(10);
            fprintf(PORT2,TXT);
            fprintf(PORT2,"\"");
            fprintf(PORT2,"\xFF\xFF\xFF");
      }
      if(num==7){
            fprintf(PORT2,"display1.t9.txt=");delay_ms(10);
            fprintf(PORT2,"\"");delay_ms(10);
            fprintf(PORT2,TXT);
            fprintf(PORT2,"\"");
            fprintf(PORT2,"\xFF\xFF\xFF");
      }
      if(num==8){
            fprintf(PORT2,"display1.t10.txt=");delay_ms(10);
            fprintf(PORT2,"\"");delay_ms(10);
            fprintf(PORT2,TXT);
            fprintf(PORT2,"\"");
            fprintf(PORT2,"\xFF\xFF\xFF");
      }
       if(num==9){
            fprintf(PORT2,"display1.t1.txt=");delay_ms(10);
            fprintf(PORT2,"\"");delay_ms(10);
            fprintf(PORT2,TXT);
            fprintf(PORT2,"\"");
            fprintf(PORT2,"\xFF\xFF\xFF");
      }
}

  VOID SENDSTRING_obd(CHAR *TXT,int8 num_txt)
{
   int8 num;
   num=num_txt;
     if(num==1){
            fprintf(PORT2,"obd.t0.txt=");delay_ms(10);
            fprintf(PORT2,"\"");delay_ms(10);
            fprintf(PORT2,TXT);
            fprintf(PORT2,"\"");
            fprintf(PORT2,"\xFF\xFF\xFF");
      }
       if(num==2){
```

```
            fprintf(PORT2,"obd.t1.txt=");delay_ms(10);
            fprintf(PORT2,"\"");delay_ms(10);
            fprintf(PORT2,TXT);
            fprintf(PORT2,"\"");
           fprintf(PORT2,"\xFF\xFF\xFF");
      }

       if(num==3){
            fprintf(PORT2,"obd.t2.txt=");delay_ms(10);
            fprintf(PORT2,"\"");delay_ms(10);
            fprintf(PORT2,TXT);
            fprintf(PORT2,"\"");
            fprintf(PORT2,"\xFF\xFF\xFF");
      }

}
  VOID SENDSTRING_motor(CHAR *TXT,int8 num_txt)
{
  int8 num;
  num=num_txt;
     if(num==1){
            fprintf(PORT2,"motor.t0.txt=");delay_ms(10);
            fprintf(PORT2,"\"");delay_ms(10);
            fprintf(PORT2,TXT);
            fprintf(PORT2,"\"");
            fprintf(PORT2,"\xFF\xFF\xFF");
      }
       if(num==2){
            fprintf(PORT2,"motor.t1.txt=");delay_ms(10);
            fprintf(PORT2,"\"");delay_ms(10);
            fprintf(PORT2,TXT);
            fprintf(PORT2,"\"");
           fprintf(PORT2,"\xFF\xFF\xFF");
      }

       if(num==3){
            fprintf(PORT2,"motor.t2.txt=");delay_ms(10);
            fprintf(PORT2,"\"");delay_ms(10);
            fprintf(PORT2,TXT);
            fprintf(PORT2,"\"");
            fprintf(PORT2,"\xFF\xFF\xFF");
      }
        if(num==4){
            fprintf(PORT2,"motor.t3.txt=");delay_ms(10);
            fprintf(PORT2,"\"");delay_ms(10);
            fprintf(PORT2,TXT);
            fprintf(PORT2,"\"");
            fprintf(PORT2,"\xFF\xFF\xFF");
      }
        if(num==5){
            fprintf(PORT2,"motor.t4.txt=");delay_ms(10);
            fprintf(PORT2,"\"");delay_ms(10);
            fprintf(PORT2,TXT);
            fprintf(PORT2,"\"");
            fprintf(PORT2,"\xFF\xFF\xFF");
      }
        if(num==6){
            fprintf(PORT2,"motor.t5.txt=");delay_ms(10);
            fprintf(PORT2,"\"");delay_ms(10);
            fprintf(PORT2,TXT);
            fprintf(PORT2,"\"");
            fprintf(PORT2,"\xFF\xFF\xFF");
      }
        if(num==7){
            fprintf(PORT2,"motor.t6.txt=");delay_ms(10);
            fprintf(PORT2,"\"");delay_ms(10);
```

```
            fprintf(PORT2,TXT);
            fprintf(PORT2,"\"");
            fprintf(PORT2,"\xFF\xFF\xFF");
      }
        if(num==8){
            fprintf(PORT2,"motor.t7.txt=");delay_ms(10);
            fprintf(PORT2,"\"");delay_ms(10);
            fprintf(PORT2,TXT);
            fprintf(PORT2,"\"");
            fprintf(PORT2,"\xFF\xFF\xFF");
      }
        if(num==9){
            fprintf(PORT2,"motor.t8.txt=");delay_ms(10);
            fprintf(PORT2,"\"");delay_ms(10);
            fprintf(PORT2,TXT);
            fprintf(PORT2,"\"");
            fprintf(PORT2,"\xFF\xFF\xFF");
      }

}
```

SOURCE CODE NODE 4

```
#include <16F18325.h>
#device ADC=10
#use delay(internal=32000000)
      #define EXT_CAN_CS   PIN_C2
      #define EXT_CAN_SI   PIN_C4
      #define EXT_CAN_SO   PIN_C3
      #define EXT_CAN_SCK  PIN_C5

   // #define CAN_USE_EXTENDED_ID         TRUE
     #define CAN_USE_EXTENDED_ID         false
     #define CAN_BRG_SYNCH_JUMP_WIDTH  0
     #define CAN_BRG_PRESCALAR  3
     #define CAN_BRG_SAM 0
     #define CAN_BRG_PHASE_SEGMENT_1  5
     #define CAN_BRG_PROPAGATION_TIME 2
     #define CAN_BRG_WAKE_FILTER FALSE
     #define CAN_BRG_PHASE_SEGMENT_2 5
     #define CAN_USE_RX_DOUBLE_BUFFER TRUE
     #define CAN_ENABLE_DRIVE_HIGH 0
     #define CAN_ENABLE_CAN_CAPTURE 1

     #define CAN_ENABLE_CAN_CAPTURE 1

#INCLUDE "can-mcp251x.C"

  struct struct_RXB1CTRL  rxmode_1;
  struct struct_RXB0CTRL  rxmode_0;

  struct rx_stat rxstat;
   int32 rx_id;
  unsigned int8 in_data[8];
   int8 rx_len;
//send a request (tx_rtr=1) for 8 bytes of data (tx_len=8) from id 24 (tx_id=24)
   int8 out_data[8];
   int32 tx_id=0B00000011001;
   int1 tx_rtr=0;
   int1 tx_ext=0;
   int8 tx_len=2;
   int8 tx_pri=3;
   int8 buffer[8];
   int8 i;
   int8 n=0;
   unsigned int8 temp_motor;
```

```
  unsigned int8 temp_drive;

void main()
{
 setup_spi(SPI_MASTER | SPI_SCK_IDLE_LOW | SPI_CLK_DIV_4);
 setup_adc_ports(sAN2 |sAN4, VSS_VDD);
 setup_adc(ADC_CLOCK_INTERNAL);
  can_init();

    buffer[0] = 101;
    buffer[1] = 201;
    buffer[2] = 255;
    buffer[3] = 255;
    buffer[4] = 255;
    buffer[5] = 255;
    buffer[6] = 255;
    buffer[7] = 255;


   while(n<10)  {output_toggle(pin_c0);delay_ms(100);n++;output_toggle(pin_a5);output_toggle(pin_c1);}

  while(TRUE)
  {

set_adc_channel(2);
delay_us(10);
temp_motor = read_adc();buffer[0] = temp_motor;

set_adc_channel(4);
delay_us(10);
temp_drive = read_adc(); buffer[1] = temp_drive  ;

    if ( can_tbe()){

       i= can_putd(tx_id,buffer,tx_len,tx_pri,tx_ext,tx_rtr); delay_ms(10);

        if (i != 0xFF)
         {
              n=0;while(n<20){output_toggle(pin_c0);n++;delay_ms(60);} //
         }
         else
         {n=0;
           while(n<10)
           {
             output_toggle(pin_c0);n++;delay_ms(50);
           }


         }

    }

 }

}
```

SOURCE CODE NODE 5

```
#include <16F1618.h>
#device ADC=8
//#fuses INTRC_IO
#use delay(internal=32000000)
#fuses NOWDT
#fuses NOMCLR
#fuses BROWNOUT
#pin_select U1TX=PIN_B5
#pin_select U1RX=PIN_B6

#use rs232(UART1,baud=9600)


//#define CAN_USE_EXTENDED_ID         TRUE
#define CAN_USE_EXTENDED_ID         false
 #define CAN_BRG_SYNCH_JUMP_WIDTH  0
 //  #define CAN_BRG_PRESCALAR  9  // for xtal 20 mhz
    #define CAN_BRG_PRESCALAR  3

     #define CAN_BRG_SEG_2_PHASE_TS TRUE
     #define CAN_BRG_SAM 0
    // #define CAN_BRG_SAM 1
     #define CAN_BRG_PHASE_SEGMENT_1  5
      #define CAN_BRG_PROPAGATION_TIME 2
      #define CAN_BRG_WAKE_FILTER FALSE
       #define CAN_BRG_PHASE_SEGMENT_2 5
       #define CAN_USE_RX_DOUBLE_BUFFER TRUE
        #define CAN_ENABLE_DRIVE_HIGH 0
        #define CAN_ENABLE_CAN_CAPTURE 0


      #define EXT_CAN_CS   PIN_C4
      #define EXT_CAN_SI   PIN_C6
      #define EXT_CAN_SO   PIN_C5
      #define EXT_CAN_SCK  PIN_C7

  ////////////////////////////////////////////////// clock in ////////////////////////////////
#pin_select T3CK = pin_c0
#pin_select T5CK = pin_c1
//////////////////////////////////////////////////  can mcp2515 library//////////////////////////
#INCLUDE "can-mcp251x.C"


#bit trisa5 = 0x08c.5
#bit d4 = 0x10c.5

#bit trisa0 = 0x08c.0

#bit trisb7 = 0x08d.7
#bit trisb4 = 0X08D.4

#bit d6 = 0x10d.4
#bit d1 = 0x10d.7


  struct rx_stat rxstat;
   int32 rx_id;
  unsigned int8 in_data[8];
   int8 rx_len;

   int8 out_data[8];
   int32 tx_id=0B00000011010;
   int1 tx_rtr=0;
   int1 tx_ext=0;
   int8 tx_len=2;
```

```
   int8 tx_pri=1;
   int8 buffer[8];
   int8 i;
   int8 n=0;
   int8 remote=0;


   unsigned int8 count_t1=0;
   unsigned int16 count_adc=0;
   unsigned int16 rpm=0;
   unsigned int16 puls_a=0;

  UNSIGNED int8 an4;
  unsigned int8 puls_h_byte;
  unsigned int8 puls_l_byte;
   struct struct_RXB0CTRL b_rxb0ctrl;

     char str[8];
  unsigned int8 speed_command=0;

#INT_TIMER1
void  TIMER1_isr(void)
{
 count_t1++;
 count_adc++;

 if(count_t1>100)
 {

 printf("%03u",speed_command);    printf("\r\n");
rpm =get_timer5();
set_timer5(0);
 count_t1=0;
  d4=~d4;//count_adc=0;
 }
 if(count_adc>10)
 {
    if(remote==0)
    {
    set_adc_channel(2);
    delay_us(10);
    an4 = read_adc();dac_write(an4);
    buffer[0] = an4;
    count_adc=0;
    }
    else
    {

    }

 }

}


void main()
{
 setup_spi(SPI_MASTER | SPI_SCK_IDLE_LOW | SPI_CLK_DIV_16);
 setup_vref(VREF_ON);
 setup_dac(DAC_VREF|DAC_OUTPUT1);//dac_write(100);
 setup_adc_ports(sAN2, VSS_VDD);
 setup_adc(ADC_CLOCK_INTERNAL);
 setup_timer_2(T2_DIV_BY_1 | T2_CLK_INTERNAL,0,1);
 setup_timer_1(T1_INTERNAL|T1_DIV_BY_1);
 setup_timer_5(T5_EXTERNAL | T5_DIV_BY_1);
 setup_timer_3(T3_EXTERNAL | T3_DIV_BY_1);// get puls A from encoder
```

```
 enable_interrupts(GLOBAL);
 enable_interrupts(INT_TIMER1);
 disable_interrupts(INT_TIMER2);

 trisa5 = 0;
 trisa0 = 0;

 trisb4 = 0;
 trisb7 = 0;

 d6=0;d1=0;
 d4=0;
while(i<10)
{
   d1=~d1; d4=~d4; d6=~d6;delay_ms(100);i++;

}
d6=0;d4=0;d1=0;
 //  can_init();

      buffer[0] = 11;      buffer[1] = 22;
      buffer[2] = 33;      buffer[3] = 44;
      buffer[4] = 55;      buffer[5] = 66;
      buffer[6] = 77;      buffer[7] = 88;
      delay_ms(1000);


   mcp2510_init();

   can_set_mode(CAN_OP_CONFIG);   //must be in config mode before params can be set
   can_set_baud();

   memset(&b_rxb0ctrl,0,1);
  // b_rxb0ctrl=0;
   b_rxb0ctrl.rxm=CAN_RX_VALID;
   b_rxb0ctrl.bukt=CAN_USE_RX_DOUBLE_BUFFER;
   mcp2510_write(RXB0CTRL, (unsigned int8)b_rxb0ctrl);
   mcp2510_write(RXB1CTRL, (unsigned int8)b_rxb0ctrl);

   //if you want to configure the TXnRTS pins, do it here.  default is off

   can_set_id(RX0MASK, 0xfff, 0);  //set mask 0 (RX BUFFER 0)
   can_set_id(RX0FILTER0, 0x04f, 0);  //set filter 0 of mask 0 (RX BUFFER 0)
   can_set_id(RX0FILTER1, 0x04f, 0);  //set filter 1 of mask 0 (RX BUFFER 0)

   can_set_id(RX1MASK, 0xfff, 0);  //set mask 1 (RX BUFFER 1)
   can_set_id(RX1FILTER2, 0x04f, 0);  //set filter 0 of mask 1 (RX BUFFER 1)
   can_set_id(RX1FILTER3, 0x04f, 0);  //set filter 1 of mask 1 (RX BUFFER 1)
   can_set_id(RX1FILTER4, 0x04f, 0);  //set filter 2 of mask 1 (RX BUFFER 1)
   can_set_id(RX1FILTER5, 0x04f, 0);  //set filter 3 of mask 1 (RX BUFFER 1)

   can_set_mode(CAN_OP_NORMAL);

////////////////////////////////////////////////////////
```

```
  while(TRUE)
   {

buffer[1] = rpm;

    if ( can_tbe()){

         i= can_putd(tx_id,buffer,tx_len,tx_pri,tx_ext,tx_rtr); delay_ms(10);

          if (i != 0xFF)
           {
               // output_toggle(pin_c2);
                n=0;while(n<20){d1=~d1;n++;delay_ms(20);} //
           }


      }

    if ( can_kbhit() )   //if data is waiting in buffer...
      {
////////////////////////////////////////////////////recive massage ////////////////////////////////////
        if(can_getd(rx_id, &in_data[0], rx_len, rxstat)) { //...then get data from buffer


         remote=in_data[0];
         speed_command=in_data[1];dac_write(speed_command);

          for (i=0;i<rx_len;i++) {//output_toggle(pin_e3);

              d6=~d6;
             delay_ms(20);
         } d6=0;


          if(remote==0)
          {
            d6=0;
          }
          else
          {
           d6=1;
          }

        }
        else {

        }

      }



  }

}
```

SOURCE CODE NODE 6

```
#include <16F18325.h>
#device ADC=8
#use delay(internal=32000000)

  #define EXT_CAN_CS   PIN_C2

  #define EXT_CAN_SI   PIN_C4
  #define EXT_CAN_SO   PIN_C3
  #define EXT_CAN_SCK  PIN_C5

   // #define CAN_USE_EXTENDED_ID         TRUE
     #define CAN_USE_EXTENDED_ID        false
     #define CAN_BRG_SYNCH_JUMP_WIDTH  0
     #define CAN_BRG_PRESCALAR  3
     #define CAN_BRG_SAM 0
     #define CAN_BRG_PHASE_SEGMENT_1  5
     #define CAN_BRG_PROPAGATION_TIME 2
     #define CAN_BRG_WAKE_FILTER FALSE
     #define CAN_BRG_PHASE_SEGMENT_2 5
     #define CAN_USE_RX_DOUBLE_BUFFER TRUE
     #define CAN_ENABLE_DRIVE_HIGH 0
     #define CAN_ENABLE_CAN_CAPTURE 1

     #define CAN_ENABLE_CAN_CAPTURE 1

#INCLUDE "can-mcp251x.C"

#bit trisa5=0x08c.5
#bit direct   = 0x10c.5

#bit trisc0 = 0x08e.0
#bit trisc1 = 0x08e.1

#bit d2= 0x10e.0
#bit d4=0x10e.1

  struct struct_RXB1CTRL  rxmode_1;
  struct struct_RXB0CTRL  rxmode_0;

  struct rx_stat rxstat;
   int32 rx_id;
  unsigned int8 in_data[8];
   int8 rx_len;

   int8 out_data[8];
   int32 tx_id=0B00000001100;
   int1 tx_rtr=0;
   int1 tx_ext=0;
   int8 tx_len=2;
   int8 tx_pri=1;
   int8 buffer[8];
   int8 i;
   int8 n=0;

   int8 dir=0;


  unsigned int8 vbat;
  unsigned int8 current;
  struct struct_RXB0CTRL b_rxb0ctrl;

void main()
{
 setup_spi(SPI_MASTER | SPI_SCK_IDLE_LOW | SPI_CLK_DIV_4);
 setup_adc_ports(sAN2 |sAN4, VSS_VDD);
 setup_adc(ADC_CLOCK_INTERNAL);
```

```
  // can_init();

      buffer[0] = 101;
      buffer[1] = 102;
      buffer[2] = 103;
      buffer[3] = 104;
      buffer[4] = 105;
      buffer[5] = 106;
      buffer[6] = 107;
      buffer[7] = 108;
////////////////////////////////////////////////////////////


   mcp2510_init();
   can_set_mode(CAN_OP_CONFIG);   //must be in config mode before params can be set
   can_set_baud();

   memset(&b_rxb0ctrl,0,1);
  // b_rxb0ctrl=0;
   b_rxb0ctrl.rxm=CAN_RX_VALID;
   b_rxb0ctrl.bukt=CAN_USE_RX_DOUBLE_BUFFER;
   mcp2510_write(RXB0CTRL, (unsigned int8)b_rxb0ctrl);
   mcp2510_write(RXB1CTRL, (unsigned int8)b_rxb0ctrl);

   //if you want to configure the TXnRTS pins, do it here.  default is off

   can_set_id(RX0MASK, 0xfff, 0);  //set mask 0 (RX BUFFER 0)
   can_set_id(RX0FILTER0, 0x05f, 0);  //set filter 0 of mask 0 (RX BUFFER 0)
   can_set_id(RX0FILTER1, 0x05f, 0);  //set filter 1 of mask 0 (RX BUFFER 0)

   can_set_id(RX1MASK, 0xfff, 0);  //set mask 1 (RX BUFFER 1)
   can_set_id(RX1FILTER2, 0x05f, 0);  //set filter 0 of mask 1 (RX BUFFER 1)
   can_set_id(RX1FILTER3, 0x05f, 0);  //set filter 1 of mask 1 (RX BUFFER 1)
   can_set_id(RX1FILTER4, 0x05f, 0);  //set filter 2 of mask 1 (RX BUFFER 1)
   can_set_id(RX1FILTER5, 0x05f, 0);  //set filter 3 of mask 1 (RX BUFFER 1)

   can_set_mode(CAN_OP_NORMAL);


////////////////////////////////////////////////////////////
trisc0 =trisc1 =0;trisa5=0;direct   =0;
     d2=d4=0;
    while(n<10)  {delay_ms(100);n++;d2=~d2;d4=~d4;}

   while(TRUE)
   {delay_ms(30);

set_adc_channel(2);
delay_us(10);
current = read_adc();buffer[0] = current;

set_adc_channel(4);
delay_us(10);
vbat = read_adc(); buffer[1] = vbat ;


      if ( can_tbe()){

         i= can_putd(tx_id,buffer,tx_len,tx_pri,tx_ext,tx_rtr); delay_ms(10);

          if (i != 0xFF)
           {
               // output_toggle(pin_c2);
                 n=0;while(n<20){d2=~d2;n++;delay_ms(20);} //
```

```
            }

        }
        ////////////////////////////////////////////////


         if ( can_kbhit() )   //if data is waiting in buffer...
        {
////////////////////////////////////////////////////recive massage ////////////////////////////////////


          if(can_getd(rx_id, &in_data[0], rx_len, rxstat)) { //...then get data from buffer


              dir=in_data[0];
              if(dir==0)
              {
                d4=0;direct   =0;
              }
              if(dir==255)
              {
                d4=1;direct   =1;
              }


          }


        }

    }
```