PROJECT REPORT

[YUKESH P - V23CE9Y2]

BODY CONTROL MODULE IN AUTOMOTIVES USING CAN PROTOCOL

AIM OF THE PROJECT

The aim of this project is to develop a vehicle control system using the CAN (Controller Area Network) protocol and LPC2129 development kits. This system is designed to provide centralized control over the vehicle's left and right indicators as well as its movement.

ABOUT THE PROJECT

This project involves creating a network of four LPC2129 development kits connected through the CAN protocol. One kit serves as the master node, while the other three serve as slave nodes responsible for controlling the left indicator, right indicator, and motor driver, respectively. This setup demonstrates the capabilities of the CAN protocol in an automotive control application.

OBJECTIVE

The primary objective of this project is to design and implement a reliable and efficient vehicle control system that uses the CAN protocol to manage various vehicle functions. The system aims to enhance control, scalability, and reliability in automotive applications.

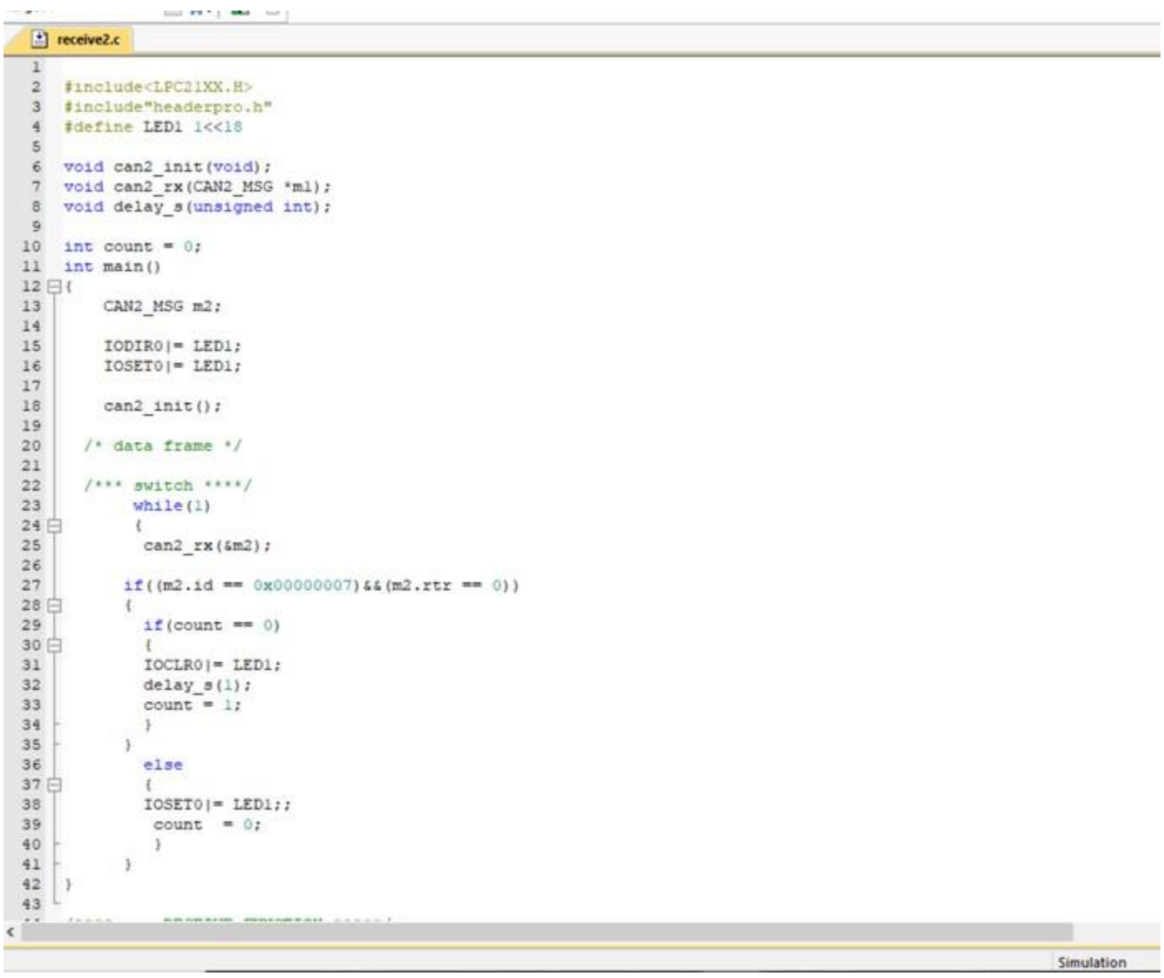
PROJECT GOALS

- Develop a CAN-based communication system: Establish a robust network using LPC2129 kits interconnected through the CAN protocol.
- Design and implement slave nodes: Create three slave nodes to control the left indicator, right indicator, and motor driver.
- Centralized control: Implement a master node to send control commands to the slave nodes, facilitating centralized management of vehicle functions.
- Enhance operational efficiency and reliability: Improve the vehicle's control system through the use of CAN protocol, ensuring robust and error-free communication.

WORKING OF THE PROJECT

- CAN Communication: The master and slave nodes communicate via the CAN protocol. The master node sends control messages with specific identifiers to address each slave node.
- Slave Nodes Functionality:
  - Left Indicator Control: The first slave node receives CAN messages to turn the left indicator on or off.
  - Right Indicator Control: The second slave node receives CAN messages to control the right indicator.
  - Motor Control: The third slave node controls the motor driver (L293D) based on commands for vehicle movement (forward, backward, left, right).
- Master Node Operation: The master node interfaces with user inputs (e.g., buttons or a remote control) and sends appropriate CAN messages to the slave nodes to execute the desired actions.

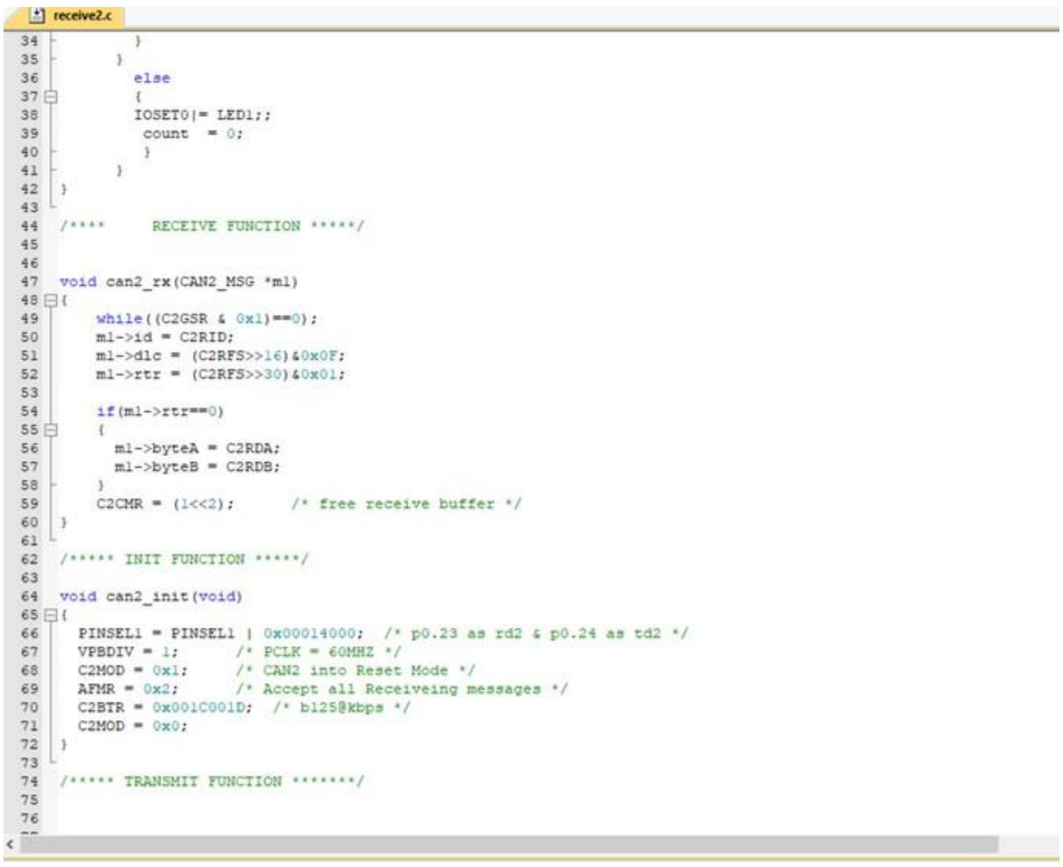
MAIN PROGRAM OF THE PROJECT



receive2.c

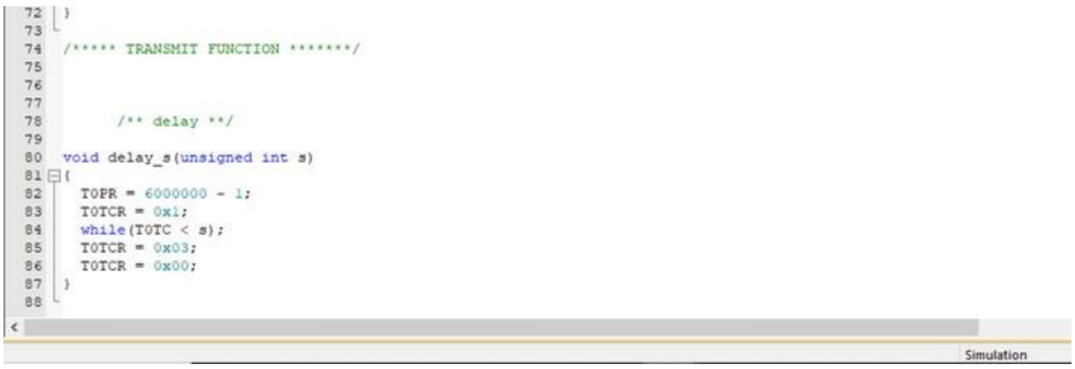
```c

#include<LPC21XX.H>
#include"headerpro.h"
#define LED1 1<<18

void can2_init(void);
void can2_rx(CAN2_MSG *m1);
void delay_s(unsigned int);

int count = 0;
int main()
{
    CAN2_MSG m2;

    IODIR0|= LED1;
    IOSET0|= LED1;

    can2_init();

  /* data frame */

  /*** switch ****/
       while(1)
       {
        can2_rx(&m2);

      if((m2.id == 0x00000007)&&(m2.rtr == 0))
      {
        if(count == 0)
        {
        IOCLR0|= LED1;
        delay_s(1);
        count = 1;
        }
      }
        else
        {
        IOSET0|= LED1;;
         count  = 0;
         }
      }
}

```

Simulation



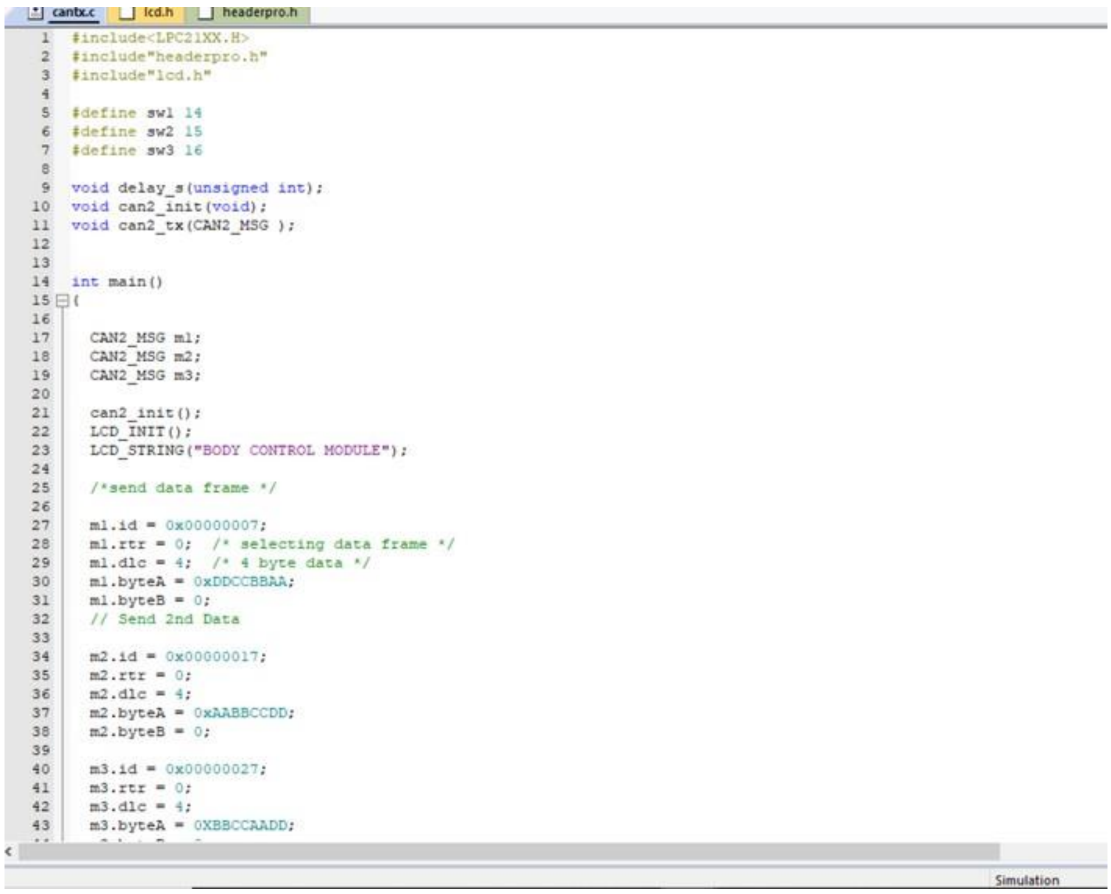
receive2.c

```c
            }
        }
          else
          {
          IOSET0|= LED1;;
           count  = 0;
           }
        }
}

/****     RECEIVE FUNCTION *****/


void can2_rx(CAN2_MSG *ml)
{
    while((C2GSR & 0x1)==0);
    ml->id = C2RID;
    ml->dlc = (C2RFS>>16)&0x0F;
    ml->rtr = (C2RFS>>30)&0x01;

    if(ml->rtr==0)
    {
      ml->byteA = C2RDA;
      ml->byteB = C2RDB;
    }
    C2CMR = (1<<2);       /* free receive buffer */
}

/***** INIT FUNCTION *****/

void can2_init(void)
{
  PINSEL1 = PINSEL1 | 0x00014000;  /* p0.23 as rd2 & p0.24 as td2 */
  VPBDIV = 1;       /* PCLK = 60MHZ */
  C2MOD = 0x1;      /* CAN2 into Reset Mode */
  AFMR = 0x2;       /* Accept all Receiveing messages */
  C2BTR = 0x001C001D;  /* b125@kbps */
  C2MOD = 0x0;
}

/***** TRANSMIT FUNCTION *******/


```

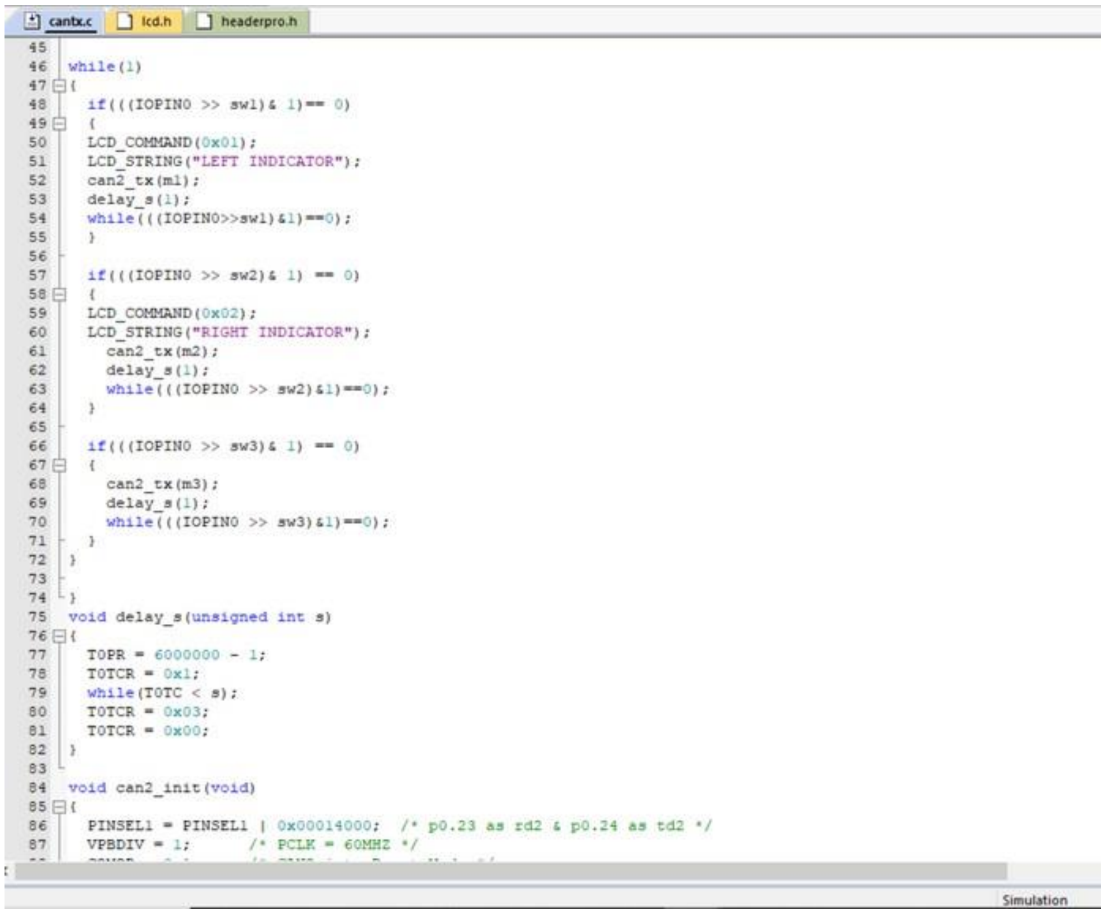


```c
}

/***** TRANSMIT FUNCTION *******/



      /** delay **/

void delay_s(unsigned int s)
{
  T0PR = 6000000 - 1;
  T0TCR = 0x1;
  while(T0TC < s);
  T0TCR = 0x03;
  T0TCR = 0x00;
}

```

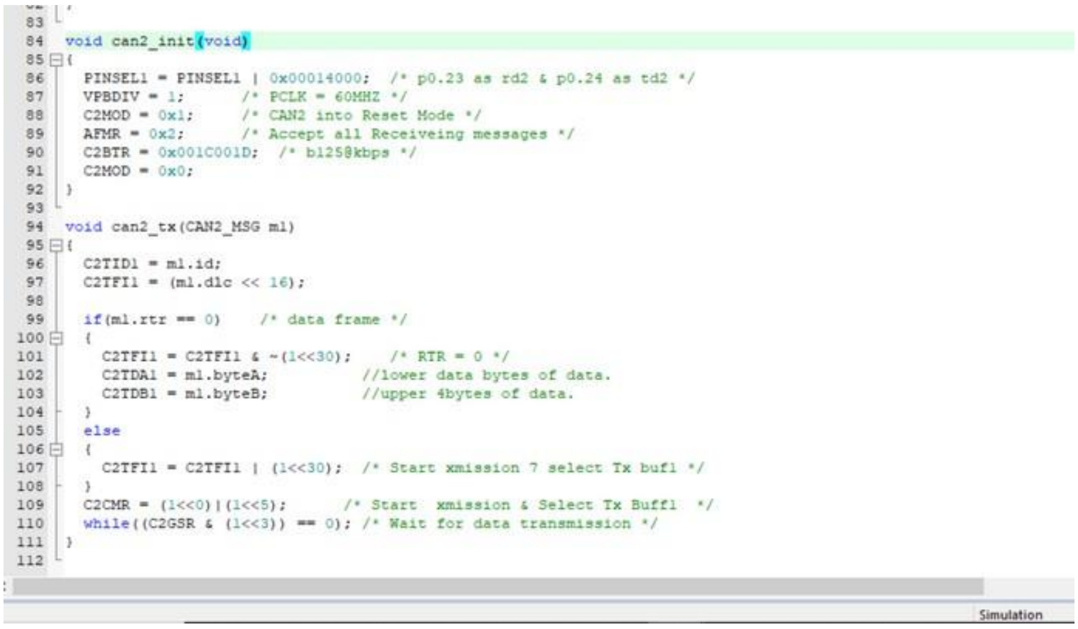
Simulation



```
#include<LPC21XX.H>
#include"headerpro.h"
#include"lcd.h"

#define sw1 14
#define sw2 15
#define sw3 16

void delay_s(unsigned int);
void can2_init(void);
void can2_tx(CAN2_MSG );


int main()
{

  CAN2_MSG m1;
  CAN2_MSG m2;
  CAN2_MSG m3;

  can2_init();
  LCD_INIT();
  LCD_STRING("BODY CONTROL MODULE");

  /*send data frame */

  m1.id = 0x00000007;
  m1.rtr = 0;  /* selecting data frame */
  m1.dlc = 4;  /* 4 byte data */
  m1.byteA = 0xDDCCBBAA;
  m1.byteB = 0;
  // Send 2nd Data

  m2.id = 0x00000017;
  m2.rtr = 0;
  m2.dlc = 4;
  m2.byteA = 0xAABBCCDD;
  m2.byteB = 0;

  m3.id = 0x00000027;
  m3.rtr = 0;
  m3.dlc = 4;
  m3.byteA = 0XBBCCAADD;
```



```

while(1)
{
  if(((IOPIN0 >> sw1)& 1)== 0)
  {
  LCD_COMMAND(0x01);
  LCD_STRING("LEFT INDICATOR");
  can2_tx(m1);
  delay_s(1);
  while(((IOPIN0>>sw1)&1)==0);
  }

  if(((IOPIN0 >> sw2)& 1) == 0)
  {
  LCD_COMMAND(0x02);
  LCD_STRING("RIGHT INDICATOR");
    can2_tx(m2);
    delay_s(1);
    while(((IOPIN0 >> sw2)&1)==0);
  }

  if(((IOPIN0 >> sw3)& 1) == 0)
  {
    can2_tx(m3);
    delay_s(1);
    while(((IOPIN0 >> sw3)&1)==0);
  }
}

}
void delay_s(unsigned int s)
{
  T0PR = 6000000 - 1;
  T0TCR = 0x1;
  while(T0TC < s);
  T0TCR = 0x03;
  T0TCR = 0x00;
}

void can2_init(void)
{
  PINSEL1 = PINSEL1 | 0x00014000;  /* p0.23 as rd2 & p0.24 as td2 */
  VPBDIV = 1;        /* PCLK = 60MHZ */
```



```c
void can2_init(void)
{
  PINSEL1 = PINSEL1 | 0x00014000;  /* p0.23 as rd2 & p0.24 as td2 */
  VPBDIV = 1;        /* PCLK = 60MHZ */
  C2MOD = 0x1;       /* CAN2 into Reset Mode */
  AFMR = 0x2;        /* Accept all Receiveing messages */
  C2BTR = 0x001C001D;  /* b125@kbps */
  C2MOD = 0x0;
}

void can2_tx(CAN2_MSG m1)
{
  C2TID1 = m1.id;
  C2TFI1 = (m1.dlc << 16);

  if(m1.rtr == 0)    /* data frame */
  {
    C2TFI1 = C2TFI1 & ~(1<<30);     /* RTR = 0 */
    C2TDA1 = m1.byteA;              //lower data bytes of data.
    C2TDB1 = m1.byteB;              //upper 4bytes of data.
  }
  else
  {
    C2TFI1 = C2TFI1 | (1<<30);  /* Start xmission 7 select Tx buf1 */
  }
  C2CMR = (1<<0)|(1<<5);      /* Start  xmission & Select Tx Buff1  */
  while((C2GSR & (1<<3)) == 0); /* Wait for data transmission */
}
```

PROBLEMS FACED

- Synchronization Issues: Ensuring that all nodes are synchronized correctly to prevent communication errors.
- Signal Interference: Managing potential interference in the CAN network, which could affect communication reliability.
- Power Supply Stability: Maintaining stable power supply to all nodes to avoid unexpected resets or failures.

CONCLUSION

This project successfully demonstrates the use of the CAN protocol in an automotive application, providing a reliable and efficient solution for vehicle control. The system's modular design allows for easy scalability and adaptation to additional functionalities, showcasing the versatility and robustness of the CAN protocol.

RESULT

The implemented vehicle control system allows for centralized management of the left indicator, right indicator, and motor control through a master node. The CAN protocol ensures reliable and error-free communication between the nodes, resulting in improved operational efficiency and control.