```
/*********************************************/
/******        RF Relay Box V1.0         ******/
/******          Declarations            ******/
/*********************************************/

/****** Relay dec ****************************/
/*
#define Relay1      PD4
#define Relay2      PD5
#define Relay3      PD6
#define Relay4      PD7

#define Relay_PORT      PORTD
#define Relay_DDR       DDRD
#define Relay_PIN       PIND
*/
/****** DipSwitch dec ************************/

#define DIP_1   PA7
#define DIP_2   PA6
#define DIP_3   PA5
#define DIP_4   PA4
#define DIP_5   PA3
#define DIP_6   PA2
#define DIP_7   PA1
#define DIP_8   PA0

#define DIP_PORTS   PORTA
#define DIP_DDR     DDRA
#define DIP_PIN     PINA

/****** Functions ****************************/

void DipSwitch_Init();

void EEPROM_Init();
```

```
/*********************************************/
/******        RF Relay Box V1.1         ******/
/******            Programm              ******/
/*********************************************/

#include <avr/io.h>
#include <stdint.h>
#include <util/delay.h>
#include <avr/interrupt.h>

#include "NRF24L01_Dec.h"
#include "Pin_changed_Sensing_Dec.h"
#include "RF_Relay_Box_V1.3_Dec.h"
#include "SoftwareSPI_Dec.h"
#include "Converter.h"
#include "SwitchLights_Dec.h"
#include "PWM_Dec_ATmega164P.h"
#include "NRF_Send_Receive_Dec.h"
#include "USART_Programmer_Mode.h"
#include "USART_Dec.h"

extern volatile unsigned char GateAddress[4];
extern volatile unsigned char RelayAddress[4][4];
extern volatile unsigned char RelayPWM[4][1];
extern volatile uint8_t RFChannel;
extern volatile uint8_t RFSendPower;
extern volatile uint8_t SendRetry;
volatile unsigned char RelayState = 0x00;
extern volatile unsigned char newSwitch;
extern volatile unsigned char oldSwitch;
extern volatile uint8_t configFlag;

uint8_t tmp_sreg;
extern volatile uint8_t USARTFlag;
extern volatile uint8_t Received;

int main()
{
    uint8_t status_reg = 0x00, try = 0x00;
    uint8_t Pipe = 0x00, Value = 0x00;

    EEPROM_Init();
    SSPI_Init();
    DipSwitch_Init();

    if((DIP_PIN & 0x80) == 0x80)
    {
            USART_Init();
            configFlag = 0xFF;
    }

    while(1)
    {
        if(((DIP_PIN & 0x80) == 0x80) && configFlag == 0xFF)
        {
            sei();
        }

        while(((DIP_PIN & 0x80) == 0x80) && configFlag == 0xFF)
        {
            if(Received == 0xFF)
            {
                if(USARTFlag != 0xB3)
                {
```

```
                    SendtoProgrammer(USARTFlag);
                }
                else
                {
                    configFlag = 0x00;
                }
                Received = 0x00;
            }
        }

        cli();

        OC1A_B_PWM_Init();
        OC2A_B_PWM_Init();
        OC0A_Timer_Init();

        Switchsensing_Init();

        NRF24L01_CustomInit(RFChannel, RFSendPower, SendRetry, RelayAddress);

        NRF_SetRXState();

        NRF_CE_Hi;

        sei();

        while(!configFlag)
        {
            while(NRF_IRQ_PIN & (1<<NRF_IRQ))
            {
                if(oldSwitch != newSwitch)
                {
                    tmp_sreg = SREG;            // save Interruptstateregister

                    cli();

                    SearchforChanges();

                    SREG = tmp_sreg;
                }
            }

            status_reg = NRF_ReadRegister(STATUS);

            if((status_reg & RX_DR) == RX_DR)
            {
                tmp_sreg = SREG;                // save Interruptstateregister

                cli();

                Payloadconvert.u256i = NRF_ReadRXPayload();

                GateAddress[0] = Payloadconvert.c[1];
                GateAddress[1] = Payloadconvert.c[2];
                GateAddress[2] = Payloadconvert.c[3];
                GateAddress[3] = Payloadconvert.c[4];

                switch(Payloadconvert.c[0])
                {
                    case 0xAC :                                 // command from Gateway
                    case 0xBF :                                 // command from "Controlle
rbox"
                        Pipe = status_reg & 0x0E;
```

```
                        if(Payloadconvert.c[7] > 0x00)
                        {
                            switch(Pipe)
                            {
                                case 0x02 :
                                    Relay1_ON(RelayPWM[0][0]);
                                    RelayState |= 0x01;
                                    Pipe = 0x01;
                                    break;
                                case 0x04 :
                                    Relay2_ON(RelayPWM[1][0]);
                                    RelayState |= 0x02;
                                    Pipe = 0x02;
                                    break;
                                case 0x06 :
                                    Relay3 ON(RelayPWM[2][0]);
                                    RelayState |= 0x04;
                                    Pipe = 0x03;
                                    break;
                                case 0x08 :
                                    Relay4_ON(RelayPWM[3][0]);
                                    RelayState |= 0x08;
                                    Pipe = 0x04;
                                    break;
                                default :
                                    break;
                            }
                            Value = 0xFF;
                        }
                        else
                        {
                            switch(Pipe)
                            {
                                case 0x02 :
                                    Relay1_OFF();
                                    RelayState &= ~0x01;
                                    Pipe = 0x01;
                                    break;
                                case 0x04 :
                                    Relay2_OFF();
                                    RelayState &= ~0x02;
                                    Pipe = 0x02;
                                    break;
                                case 0x06 :
                                    Relay3_OFF();
                                    RelayState &= ~0x04;
                                    Pipe = 0x03;
                                    break;
                                case 0x08 :
                                    Relay4_OFF();
                                    RelayState &= ~0x08;
                                    Pipe = 0x04;
                                    break;
                            }
                            Value = 0x00;
                        }

                        NRF_WriteRegister(0xff, STATUS);
                        NRF_FlushRXBuffer();

                        switch(Payloadconvert.c[0])
                        {
                            case 0xAC:                                  // Answer to Gatew
ay
```

```
                                Payloadconvert.c[0] = 0xAC;
                                Payloadconvert.c[1] = 0x00;
                                Payloadconvert.c[2] = 0x00;
                                Payloadconvert.c[3] = Value;
                                SendBack(Payloadconvert.u256i, Pipe);
                                break;

                            case 0xBF:                                  // Message to Gate
way
                                switch(Pipe)
                                {
                                    case 0x01 :
                                        Addressconvert.u32i = NRF_ReadRXAddress(RX_ADDR_P1
);
                                        break;
                                    case 0x02 :
                                        Addressconvert.u32i = NRF_ReadRXAddress(RX_ADDR_P2
);
                                        break;
                                    case 0x03 :
                                        Addressconvert.u32i = NRF_ReadRXAddress(RX_ADDR_P3
);
                                        break;
                                    case 0x04 :
                                        Addressconvert.u32i = NRF_ReadRXAddress(RX_ADDR_P4
);
                                        break;
                                }

                                Payloadconvert.c[0] = 0xFE;
                                Payloadconvert.c[1] = Addressconvert.c[0];
                                Payloadconvert.c[2] = Addressconvert.c[1];
                                Payloadconvert.c[3] = Addressconvert.c[2];
                                Payloadconvert.c[4] = Addressconvert.c[3];
                                Payloadconvert.c[5] = 0x00;
                                Payloadconvert.c[6] = 0x00;
                                Payloadconvert.c[7] = Value;
                                SendToGate(Payloadconvert.u256i);
                                break;
                        }
                        break;

                    default:
                        break;
                }
                SREG = tmp_sreg;
            }
            else
            {
                if((status_reg & TX_DS) == TX_DS)
                {
                    tmp_sreg = SREG;   // save Interruptstateregister

                    cli();

                    NRF_WriteRegister(0xff, STATUS);
                    NRF_CE_Lo;
                    NRF_SetRXState();
                    NRF_CE_Hi;

                    SREG = tmp_sreg;
                }
                else
                {
```

```
                    if((status_reg & MAX_RT) == MAX_RT)
                    {
                        tmp_sreg = SREG;   // save Interruptstateregister

                        cli();
                        if(try < 3)
                        {
                            NRF_WriteRegister(0xff, STATUS);
                            SendBack(Payloadconvert.u256i, Pipe);
                            try++;
                        }
                        else
                        {
                            NRF_CE_Lo;
                            NRF_SetRXState();
                            NRF_CE_Hi;
                        }

                        SREG = tmp_sreg;
                    }
                    else
                    {
                    }
                }
            }

            status_reg = 0x00;
        }
    }
}

void DipSwitch_Init()
{
    DIP_DDR &= ~(1<<DIP_1) | ~(1<<DIP_2) | ~(1<<DIP_3) | ~(1<<DIP_4) | ~(1<<DIP_5) | ~(1<<
DIP_6) | ~(1<<DIP_7) | ~(1<<DIP_8); // Set DIPSwitches as INPUT
    DIP_PORTS |= (1<<DIP_1) | (1<<DIP_2) | (1<<DIP_3) | (1<<DIP_4) | (1<<DIP_5) | (1<<DIP_
6) | (1<<DIP_7) | (1<<DIP_8);           // Set PullupResistor

    _delay_ms(10);
    RFChannel = (DIP_PIN & 0x7F);
}
```