Nordic VLSI ASA

# nRF24XX Mouse Keyboard Demo **nAN24-06**

## INTRODUCTION

The nRF24XX Mouse Keyboard Demo is a demonstration of how the nRF24XX chipset can be used in a wireless mouse/keyboard application. The demo contains two "Combined Mouse/Keyboard 2.4GHz Transmitter" boards and one "nRF2401 Low Cost USB Dongle." The "nRF2401 Low Cost USB Dongle" is a stand-alone design, ready to interface with any wireless mouse and wireless keyboard using the ShockBurst™ technology. It is able to receive data from both the mouse and the keyboard simultaneously by using the DuoCeiver™ technology. The "nRF2401 Low Cost USB Dongle" design is described in the application note nAN24-04.

In Figure 1 a typical set-up of the demo is shown. A standard PS/2 compatible keyboard is connected to one of the "Combined Mouse/Keyboard 2.4GHz Transmitter" boards and a standard PS/2 mouse to the other. The one "nRF2401 Low Cost USB Dongle" is inserted into a PC USB port. The mouse and the keyboard will now act and behave as wireless devices.

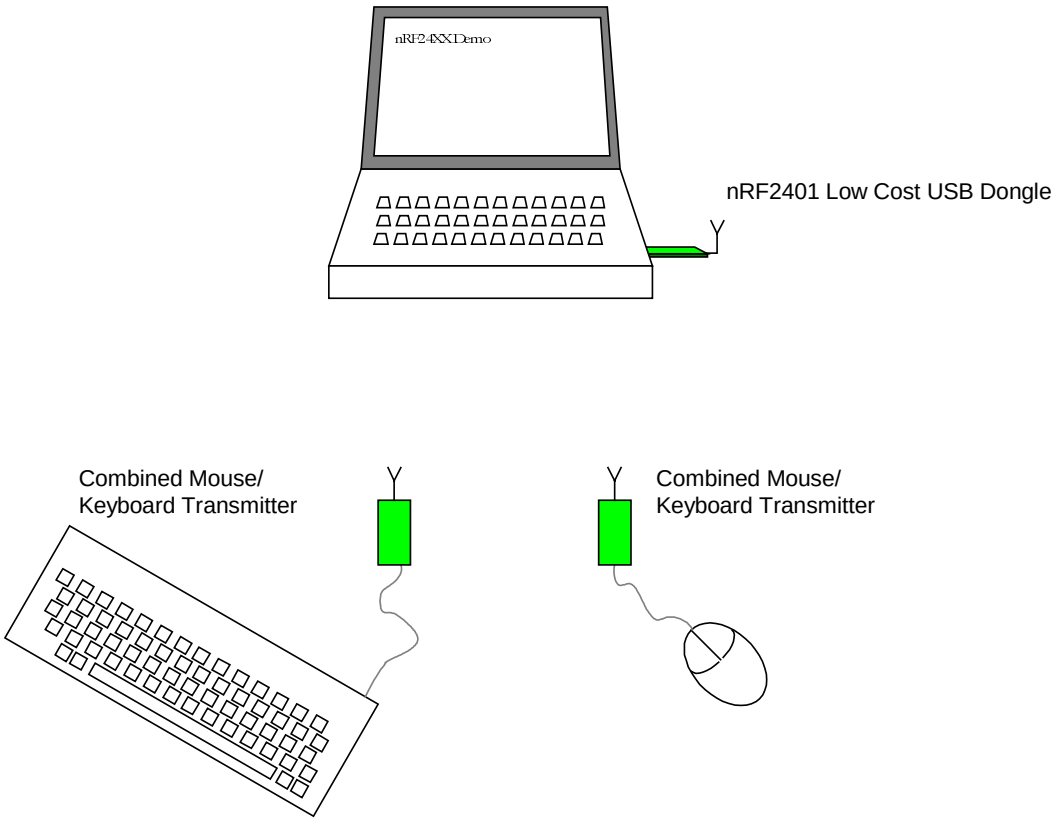


Figure 1: Typical demo set-up

# The "nRF24XX Mouse Keyboard Demo" Users guide

## DuoCeiver Demonstration

1. Insert the "nRF2401 Low Cost USB Dongle" into one of the PCs USB ports. The green LED on the board will flash five times to indicate USB enumeration success. The PC will detect the dongle as new hardware if it has never been used on that PC before. If the PC asks for a driver for the new hardware, please install the current operation systems general human interface device drivers to continue.
2. Ensure that both "Combined Mouse/Keyboard 2.4GHz Transmitter" boards are switched off.
3. Connect a PS/2 compatible mouse to one of the "Combined Mouse/Keyboard 2.4GHz Transmitter" board and switch it on.
4. Connect a PS/2 compatible keyboard to the other "Combined Mouse/Keyboard 2.4GHz Transmitter" board and switch it on.
5. Press the "Ctrl" key on the keyboard attached to the "Combined Mouse/Keyboard 2.4GHz Transmitter" board.
6. Use the wireless mouse to open a text editor on the PC.
7. Use the wireless keyboard to write some text in the text editor and move the mouse around to show the DuoCeiver™ technology in use.

## Use the "Combined Mouse/Keyboard 2.4GHz Transmitter" as a Power Point presentation controller

1. Insert the "nRF2401 Low Cost USB Dongle" into one of the PCs USB ports. The green LED on the board will flash five times to indicate USB enumeration success. The PC will detect the dongle as new hardware if it has never been used on that PC before. If the PC asks for a driver for the new hardware, please install the current operation systems general human interface device drivers to continue.
2. Switch on one of the "Combined Mouse/Keyboard 2.4GHz Transmitter" boards without any PS/2 equipment attached.
3. Press and release the button on the board to move one slide ahead.
4. Press and hold the button for 1 second to move one slide back. Continue to hold the button to step even more slides back.
5. During this mode of operation, the "Combined Mouse/Keyboard 2.4GHz Transmitter" board will transmit a packet every 40 second telling the PC that the "Ctrl" key has been pressed. This will prevent the screensaver to be activated.

## This demo vs. a real Mouse/Keyboard design

In a real wireless mouse/keyboard design, it will not be necessary to make the mouse/keyboard end of the link so complicated as this demo does. This demo uses off-the-shelf PS/2 mouse and keyboard to demonstrate the *functionality* that can be achieved by using the nRF24XX chipset.

In a real wireless mouse/keyboard application it will be sufficient to use only the nRF24E2. To learn how to make a wireless mouse based on the nRF24E2, read white paper: “3axis mouse using nRF24Ex.” To learn how to make a wireless keyboard based on the nRF24E2, read white paper: “Wireless keyboard using nRF24Ex.”

The “nRF2401 Low Cost USB Dongle” is a design that is ready to use in any wireless mouse/keyboard application based on the nRF24XX chipset. Please read application note nAN24-04 to learn more about the “nRF2401 Low Cost USB Dongle.”

# HARDWARE

As shown in the schematics, Figure 2, the RF part is based on the nRF24Ex reference design that can be downloaded from www.nvlsi.no. The design has a 16MHz crystal and an external EPROM for firmware storage. The firmware will use the ShockBurst™ technology to transmit the packets from the keyboard. The ShockBurst™ technology is designed to minimize the current consumption per transmitted bit in order to extend the battery lifetime. Read more about the ShockBurst™ technology in the nRF24Ex datasheet and in a white paper that can be downloaded from www.nvlsi.no.

In addition a PS/2 interface is made out of two NPN transistors and two voltage dividers as shown in Figure 3.

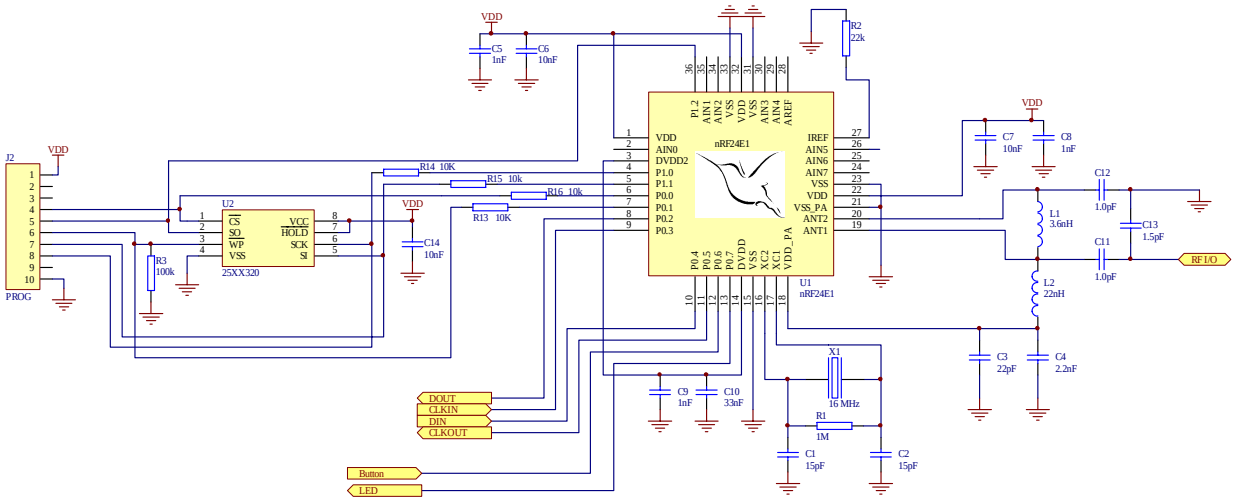


Figure 2: nRF24E1 core schematics

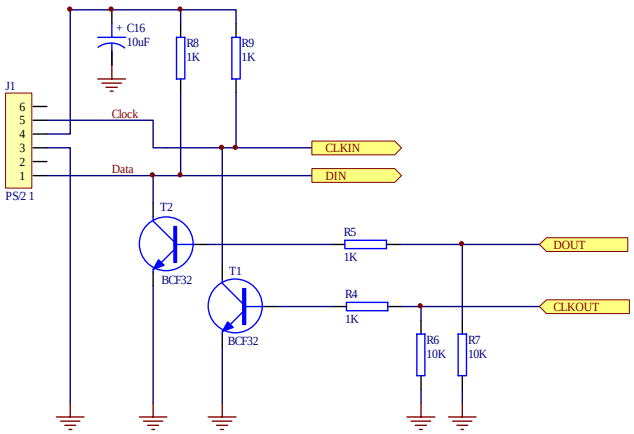


Figure 3: PS/2 interface

The PS/2 interface is connected to four general-purpose IO lines on the nRF24E1. Two lines are used to pull down the clock and data line, and the two others are used to read the value on the data and clock line.

# SOFTWARE

The system functionality is quite simple; The two “Combined Mouse/Keyboard 2.4GHz Transmitter” boards will read data from the mouse and the keyboard and transmit it to the “nRF2401 Low Cost USB Dongle” that will present the data to the PC. The mouse uses data channel 2 and the keyboard uses data channel 1. Data channel 1 is set to be 2410MHz, since the spacing between data channel 1 and 2 is 8MHz this will cause the data channel 2 to be at 2418MHz. The “nRF2401 Low Cost USB Dongle” is actually receiving at both these channels simultaneously.

The two “Combined Mouse/Keyboard 2.4GHz Transmitter” board will detect if it is connected to a mouse or a keyboard at power on. It will select channel and PS/2 protocol based on what equipment it detects. If no equipment is detected it will go into power point remote mode. The button on the board will then act as a left mouse button when it is pressed and released, if it is pressed and held down it will act as the “Page Up” key on the keyboard.

For details on functionality of the “nRF2401 Low Cost USB Dongle,” please refer to the application note nAN24-04 that can be downloaded from www.nvlsi.no.

# CONCLUSION

The wireless mouse/keyboard demo is intended for demonstration of the DuoCeiver™ technology, ShockBurst™ technology, interfacing to USB and custom interfacing, in this case the PS/2 interface.

The “nRF2401 Low Cost USB Dongle” can be reused in a high volume application. It is designed with low cost in mind and has few components.

The “Combined Mouse/Keyboard 2.4GHz Transmitter” board must not be compared with what is needed for a wireless mouse/keyboard based on a nRF24xx chip. A real wireless mouse/keyboard will not need many of the components that you find on the “Combined Mouse/Keyboard 2.4GHz Transmitter” board.

To see what a wireless mouse or a wireless keyboard based on the nRF24Ex needs of external components, please refer to the white papers: “3axis mouse using nRF24Ex.” and “Wireless keyboard using nRF24Ex.”

# APPENDIX

Listing of source code for the “Combined Mouse/Keyboard 2.4GHz Transmitter.

```
#include <Nordic\reg24e1.h>

sbit DATAIN  = P0^3;
sbit CLKIN   = P0^4;
sbit DATAOUT = P0^5;
sbit CLKOUT  = P0^2;
sbit BUTTON  = P0^6;



struct RFConfig
{
    unsigned char n;
    unsigned char buf[15];
};

struct ScanCodeSet
{
      unsigned char buf[96];
};



typedef struct RFConfig RFConfig;
typedef struct ScanCodeSet ScanCodeSet;


#define ADDR_INDEX  7   // Index to address bytes in RFConfig.buf
#define ADDR_COUNT  5   // Number of address bytes

const char PWMHIGHSET=180;
volatile unsigned char counter1, counter2,trigger,screen=0;
volatile unsigned char counter3,mode=0,syskeys=0,sleep=0;


const RFConfig tconf =
{
    14,
    0x20, 0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0xBB,
    0xBB, 0xBB, 0xBB, 0xBB, 0xA1, 0x6F, 0x24
};

idata const ScanCodeSet scanmatrix =

{ // Used to transelate between PS/2 ScanCode Set 1 and USB Scan codes
       0x00, 0x29, 0x1E, 0x1F, 0x20, 0x21, 0x22, 0x23,    //0 - 7
       0x24, 0x25, 0x26, 0x27, 0x2D, 0x2E, 0x2A, 0x2B,    //8 - 15
       0x14, 0x1A, 0x08, 0x15, 0x17, 0x1C, 0x18, 0x0C,    //16 - 23
       0x12, 0x13, 0x2F, 0x30, 0x28, 0xE0, 0x04, 0x16,    //24 - 31
       0x07, 0x09, 0x0A, 0x0B, 0x0D, 0x0E, 0x0F, 0x33,    //32 - 39
       0x34, 0x35, 0xE1, 0x31, 0x1D, 0x1B, 0x06, 0x19,    //40 - 47
       0x05, 0x11, 0x10, 0x36, 0x37, 0x38, 0xE5, 0x55,    //48 - 55
       0xE2, 0x2C, 0x39, 0x3A, 0x3B, 0x3C, 0x3D, 0x3E,    //56 - 63
       0x3F, 0x40, 0x41, 0x42, 0x43, 0x00, 0x00, 0x5F,    //64 - 71
       0x60, 0x61, 0x56, 0x5C, 0x5D, 0x5E, 0x57, 0x59,    //72 - 79
       0x5A, 0x5B, 0x62, 0x63, 0x00, 0x00, 0x64, 0x44,    //80 - 87
       0x45, 0x67, 0x00, 0x00, 0x00, 0x68, 0x69, 0x6A    //88 - 95
};
```

```
void TXMousePacket(unsigned char b,x,y,z);
void Transmitter(unsigned char channel);
void TXKeyPacket(unsigned char b,x);

void Delay100us(volatile unsigned char n)
{
    unsigned char i;
    while(n--)
        for(i=0;i<32;i++)
            ;
}


void Timer (void) interrupt 1 using 2
{
    if (++counter1 == 75)
    {
        if (++counter2 == 50)
        {
                  if (++counter3 == 6)
             {
                     PWMDUTY=PWMHIGHSET;
                          counter3=0;
                          if(ADCDATAH<0xAE)
                          {
                                 counter3=5;
                                 PWMDUTY=PWMHIGHSET+50;
                          }
                          screen++;
                    }
                    counter2 = 0;
                    trigger=0;
        }
        counter1 = 0;
             if(PWMDUTY>20)
                    PWMDUTY=PWMDUTY-20;
             else
             {
                    if(PWMDUTY)
                           PWMDUTY--;
                    if(PWMDUTY)
                           PWMDUTY--;
                    if(PWMDUTY)
                           PWMDUTY--;
             }

    }
}


unsigned char SpiReadWrite(unsigned char b)
{
    EXIF &= ~0x20;               // Clear SPI interrupt
    SPI_DATA = b;                // Move byte to send to SPI data register
    while((EXIF & 0x20) == 0x00) // Wait until SPI hs finished transmitting
        ;
    return SPI_DATA;
}

void Transmitter(unsigned char channel)
{
       unsigned char b;
       CE=0;
       CS = 1;
    Delay100us(0);
    for(b=0;b<tconf.n;b++)
```

```
    {
        SpiReadWrite(tconf.buf[b]);
    }
    SpiReadWrite(channel);
    CS = 0;

}

void TXMousePacket(unsigned char b,x,y,z)
{
    EA = 0;
    CE=1;
    y=~y+1;
    z=~z+1;
    SpiReadWrite(0xBB);
    SpiReadWrite(0xE7);
    SpiReadWrite(0xBB);
    SpiReadWrite(0xE7);
    SpiReadWrite(0xBB);
    SpiReadWrite(b);
    SpiReadWrite(x);
    SpiReadWrite(y);
    SpiReadWrite(z);
    for(b=0;b<6;b++)
        SpiReadWrite(0);
    CE=0;
    counter1=0;
    counter2=0;
    counter3=0;
    sleep=0;
    EA = 1;
}


void TXKeyPacket(unsigned char b,x)
{
    EA = 0;
    CE=1;
    SpiReadWrite(0xE7);
    SpiReadWrite(0xE7);
    SpiReadWrite(0xBB);
    SpiReadWrite(0xE7);
    SpiReadWrite(0xE7);
    SpiReadWrite(b);
    SpiReadWrite(x);
    for(b=0;b<8;b++)
        SpiReadWrite(0);
    CE=0;
    counter1=0;
    counter2=0;
    counter3=0;
    sleep=0;
    EA = 1;
}

void CheckButton(void)
{
    unsigned char stepback=1;
    if(screen==10)
        if(!mode)
            {       // Transmitt a "prevent screensaver" packet
                PWMDUTY = PWMHIGHSET;
                screen=0;
                Transmitter(0x14);
                Delay100us(40);
                TXKeyPacket(1,0);
```

```
                        Delay100us(40);
                        TXKeyPacket(0,0);
                        Delay100us(40);
                        Transmitter(mode);
                }
        if(!BUTTON)
        {
                Delay100us(100);                        // Debounce
                if(!BUTTON)
                {
                        Transmitter(0x14);
                        trigger=1;
                        counter1=0;
                        counter2=0;
                        counter3=0;
                        while(!BUTTON)
                                if(!trigger)
                                {
                                    EA = 0;
                                       PWMDUTY = PWMHIGHSET;
                                       TXKeyPacket(0,0x4B);

                                       trigger=1;
                                       counter2=15;
                                       counter3=0;
                                       stepback=0;
                                    EA = 1;
                                };

                        if(trigger&stepback)
                        {
                                EA = 0;
                                PWMDUTY = PWMHIGHSET;
                                Transmitter(0x24);
                                Delay100us(40);
                                TXMousePacket(1,0,0,0);
                                Delay100us(40);
                                TXMousePacket(1,0,0,0);
                                Delay100us(40);
                                TXMousePacket(1,0,0,0);
                                Delay100us(40);
                                TXMousePacket(0,0,0,0);
                                Delay100us(40);
                                TXMousePacket(0,0,0,0);
                                Delay100us(40);
                                TXMousePacket(0,0,0,0);
                            EA = 1;
                        }
                        while(!BUTTON)
                        ;
                        Delay100us(100);    // Debounce
                }
                Transmitter(mode);          // Set it back to original;
        }
}

void Init(void)
{
        unsigned char i;
    P0_DIR = 0x58;          // P0.3, P0.4, P0.6 is input, the rest is output
       P0_ALT = 0x80;       // Enable PWM
       PWMCON = 0xFF;
       PWMDUTY = 0;
    P0 = 0x7F;

       ADCCON=0x68;         // set up ADC to make samples of VDD
```

```
    ADCSTATIC=0x21;            // 8 bit resolution

  PWR_UP = 1;                 // Turn on Radio
    for(i=0;i<100;i++)
        Delay100us(30);      // Wait > 3ms
  SPICLK = 0;                  // Max SPI clock (XTAL/8)
  SPI_CTRL = 0x02;             // Connect internal SPI controller to Radio
    Transmitter(0x24);
    for(i=0;i<100;i++)
    {
          PWMDUTY = i;
          Delay100us(100-i);
    }
    for(i=0;i<100;i++)
    {
          PWMDUTY = 99-i;
          Delay100us(i);
    }

    TL0 = 0;
  TMOD = 0x22;
  TCON = 0x50;
  ET0 = 1;
  EA = 1;

}


unsigned char calculateparity(unsigned char b)
{
unsigned char nibtab[] = {0, 1, 1, 2, 1, 2, 2, 3, 1, 2, 2, 3 , 2, 3, 3, 4};
return !((nibtab[b & 0x0f] + nibtab[b >> 4]) & 0x01);

}

unsigned char receivePS2byte(void)
{
          unsigned char i,b=0;

          while(CLKIN)
                CheckButton();
          while(!CLKIN)
                CheckButton();
          for(i=0;i<8;i++)
          {
                b=b>>1;
                while(CLKIN)
                      CheckButton();
                if(DATAIN)
                {
                      b|=0x80;
                }

                while(!CLKIN)
                      CheckButton();
          }
          while(CLKIN)
                CheckButton();
          while(!CLKIN)
                CheckButton();
          while(CLKIN)
                CheckButton();
          while(!CLKIN)
                CheckButton();
          return b;
}
```

```
void sendPS2byte(unsigned char b)
{
	unsigned char i,parity;
	parity=calculateparity(b);
	CLKOUT=1;			// Bring the clock low for 100us
	Delay100us(1);
	DATAOUT=1;			// Bring the Data low
	Delay100us(1);
	CLKOUT=0;			// Release it

	while(!CLKIN)
		CheckButton();
	for(i=0;i<8;i++)
	{
		while(CLKIN)
			CheckButton();
		if(b&0x01)
			DATAOUT=0;
		else
			DATAOUT=1;
		b=b>>1;
		while(!CLKIN)
			CheckButton();
	}
		while(CLKIN)
			CheckButton();
		if(parity)
			DATAOUT=0;
		else
			DATAOUT=1;
		while(!CLKIN)
			CheckButton();
		while(CLKIN)
			CheckButton();
		DATAOUT=0;
		while(!CLKIN)
			CheckButton();
		while(CLKIN)
			CheckButton();
		while(!CLKIN)
			CheckButton();

}



void SetResolution(unsigned char res)
{
	sendPS2byte(0xE8);
	receivePS2byte();
	sendPS2byte(res);
	receivePS2byte();
}

unsigned char ReadDeviceType(void)
{
	sendPS2byte(0xF2);
	receivePS2byte();
	return receivePS2byte();
}

void SetSamplRate(unsigned char rate)
{
	sendPS2byte(0xF3);
```

```
        receivePS2byte();
        sendPS2byte(rate);
        receivePS2byte();
}

unsigned char ResetMouse(void)
{
        sendPS2byte(0xFF);
        receivePS2byte();
        receivePS2byte();
        return receivePS2byte();
}

unsigned char GetSpecialScan(unsigned char b)
{
        switch(b)
        {
                case 28:      {return 0x58; break;}
                case 29:      {return 0xE4; break;}
                case 53:      {return 0x54; break;}
                case 56:      {return 0xE6; break;}
                case 71:      {return 0x4A; break;}
                case 72:      {return 0x52; break;}
                case 73:      {return 0x4B; break;}
                case 75:      {return 0x50; break;}
                case 77:      {return 0x4F; break;}
                case 79:      {return 0x4D; break;}
                case 80:      {return 0x51; break;}
                case 81:      {return 0x4E; break;}
                case 82:      {return 0x49; break;}
                case 83:      {return 0x4C; break;}
                case 91:      {return 0xE3; break;}
                case 92:      {return 0xE7; break;}
                case 93:      {return 0x65; break;}
                case 94:      {return 0x81; break;}
                case 95:      {return 0x82; break;}
                case 99:      {return 0x83; break;}
                default:       return 0x00;
        }
}

void main(void)
{
        unsigned char b,x,y,z=0;
        unsigned char ScrollMouse=0;
    Init();
        if(ResetMouse()==0) // Detect if it is a mouse or a keyboard
        {
                mode=0x24;
                ResetMouse();
                ResetMouse();
                SetSamplRate(200);
                SetSamplRate(100);
                SetSamplRate(80);
                if(ReadDeviceType()==3)
                {
                        ScrollMouse=1;
                        SetSamplRate(10);
                        ReadDeviceType();
                        SetResolution(3);// Set resolution 8 counts/mm
                        sendPS2byte(0xE6);
                        receivePS2byte();
                        SetSamplRate(100);
                        sendPS2byte(0xF4);
                        receivePS2byte();
                }
```

```
            else
            {
                  ScrollMouse=0xFF;
                  ResetMouse();
                  sendPS2byte(0xF6);
                  receivePS2byte();
                  SetResolution(3);// Set resolution 8 counts/mm
                  SetSamplRate(100);
                  sendPS2byte(0xF4);
                  receivePS2byte();
            }
          while(1)
            {
                  b=receivePS2byte();
                  x=receivePS2byte();
                  PWMDUTY = PWMHIGHSET;
                  y=receivePS2byte();
                  if(ScrollMouse==1)
                        z=receivePS2byte();
                  else
                        z=0;
                  counter1=0;
                  counter2=0;
                  counter3=0;
                  PWMDUTY = 0;
                  TXMousePacket(b,x,y,z);

            }
      }
      else
      {
            mode=0x14;
            Transmitter(0x14);
            sendPS2byte(0xED);
            receivePS2byte();
            sendPS2byte(0x00);
            receivePS2byte();
            sendPS2byte(0xF0);
            receivePS2byte();
            sendPS2byte(0x01);
            receivePS2byte();
            sendPS2byte(0xF3);
            receivePS2byte();
            sendPS2byte(0x00);
            receivePS2byte();
            while(1)
            {
                  b=receivePS2byte();
                  if(b!=0xE0)
                  {
                        switch(b)
                        {
                              case 0xB8://leftaltclr
                              {
                                    syskeys &= ~0x04;
                                    break;
                              }
                              case 0xAA://leftshiftclr
                              {
                                    syskeys &= ~0x02;
                                    break;
                              }
                              case 0x9D://leftctrlclr
                              {
                                    syskeys &= ~0x01;
                                    break;
```

```
                        }
                        case 0xB6://rightshiftclr
                        {
                                syskeys &= ~0x20;
                                break;
                        }
                        default:;
                }
                if(b<0x80)
                {
                        switch(b)
                        {
                                case 0x38://leftaltset
                                {
                                        syskeys |= 0x04;
                                        break;
                                }
                                case 0x2A://leftshiftset
                                {
                                        syskeys |= 0x02;
                                        break;
                                }
                                case 0x1D://leftctrlset
                                {
                                        syskeys |= 0x01;
                                        break;
                                }
                                case 0x36://rightshiftset
                                {
                                        syskeys |= 0x20;
                                        break;
                                }
                                default:
                                {
                                        x=scanmatrix.buf[b];
                                        PWMDUTY = PWMHIGHSET;
                                        TXKeyPacket(syskeys,x);
                                        PWMDUTY = 0;
                                }
                        }
                }
        }
        else    // b=0xE0 - New byte will arrive
        {
                b=receivePS2byte(); //read new byte
                switch(b)
                {
                        case 0x5B://leftguiset
                        {
                                PWMDUTY = PWMHIGHSET;

                                syskeys |= 0x08;
                                TXKeyPacket(syskeys,0);
                                PWMDUTY = 0;
                                break;
                        }
                        case 0x5C://rightguiset
                        {
                                PWMDUTY = PWMHIGHSET;

                                syskeys |= 0x10;
                                TXKeyPacket(syskeys,0);
                                PWMDUTY = 0;
                                break;
                        }
                        case 0x38://rightaltset
```

```
                                {
                                        syskeys |= 0x40;
                                        break;
                                }
                                case 0x1D://rightctrlset
                                {
                                        syskeys |= 0x80;
                                        break;
                                }
                                case 0xB8://rightaltclr
                                {
                                        syskeys &= ~0x40;
                                        break;
                                }
                                case 0x9D://rightguiclr
                                {
                                        syskeys &= ~0x10;
                                        break;
                                }
                                case 0xDB://leftguiclr
                                {
                                        PWMDUTY = PWMHIGHSET;

                                        syskeys &= ~0x08;

                                        TXKeyPacket(syskeys,0);
                                        PWMDUTY = 0;
                                        break;
                                }
                                case 0xDC://rightguiclr
                                {
                                        PWMDUTY = PWMHIGHSET;
                                        syskeys &= ~0x10;
                                        TXKeyPacket(syskeys,0);
                                        PWMDUTY = 0;
                                        break;
                                }
                                default:
                                {
                                        if(b<0x80)
                                        {
                                                x=GetSpecialScan(b);
                                                PWMDUTY = PWMHIGHSET;
                                                TXKeyPacket(syskeys,x);
                                                PWMDUTY = 0;
                                        }
                                }
                        }

                }
        }
}
```

# YOUR NOTES

# For Your nearest dealer, please see http://www.nvlsi.no

**Main Office:**
Vestre Rosten 81, N-7075 Tiller, Norway
Phone: +47 72 89 89 00, Fax: +47 72 89 89 89

**Visit the Nordic VLSI ASA website at http://www.nvlsi.no**