LAMPIRAN A

FOTO ROBOT DAN REMOTE CONTROL

Tampak Depan

Tampak Belakang

Tampak Samping

Tampak Atas

Tampak Bawah

Remote Kontrol

LAMPIRAN B

Instruksi Program Pemancar

```
/*****************************************************
This program was produced by the CodeWizardAVR

Project                 : R O V
Version         : 1.0
Date            : 11/12/2009
Author          : Asri Asmarariani Putri
Company         : Lab UKM
Comments        : Program Transmitter

Chip type                       : ATmega16
Program type            : Application
Clock frequency         : 11.059200 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 256
*****************************************************/
#include <mega16.h>
#include <delay.h>

// Alphanumeric LCD Module functions
#asm
        .equ __lcd_port=0x18 ;PORTB
#endasm
#include <lcd.h>

#define up                      PINC.0
#define down            PINC.1
#define left                    PINC.2
#define right           PINC.3
#define forward         PINC.4
#define backward        PINC.5

#define RXB8 1
#define TXB8 0
#define UPE 2
#define OVR 3
#define FE 4
#define UDRE 5
#define RXC 7


#define FRAMING_ERROR (1<<FE)
```

```
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<OVR)
#define DATA_REGISTER_EMPTY (1<<UDRE)


#define RX_COMPLETE (1<<RXC)

flash const unsigned char dat_1[]={"   UP    "};
flash const unsigned char dat_2[]={"  DOWN   "};
flash const unsigned char dat_3[]={"  LEFT   "};
flash const unsigned char dat_4[]={"  RIGHT  "};
flash const unsigned char dat_5[]={" FORWARD "};
flash const unsigned char dat_6[]={" BACKWARD"};
flash const unsigned char dat_7[]={" ........"};

flash const unsigned char men_1[]={" ROBOT DALAM AIR"};
flash const unsigned char men_2[]={" Menggunakan RC "};

flash const unsigned char men_3[]={"   designed by  "};
flash const unsigned char men_4[]={"Asri Asmarariani"};

flash const unsigned char men_5[]={"U = Up          "};
flash const unsigned char men_6[]={"D = Down        "};

flash const unsigned char men_7[]={"F = Forward     "};
flash const unsigned char men_8[]={"B = Backward    "};

flash const unsigned char men_9[] ={"L = Turn Left   "};
flash const unsigned char men_10[]={"R = Turn Right  "};

flash const unsigned char men_11[]={"    --.-- meter "};
flash const unsigned char men_12[]={" ........       "};

flash const unsigned char men_13[]={"  --* MENU *--  "};
flash const unsigned char men_14[]={"               "};

// Declare your global variables here
unsigned char tombol;
unsigned char save[2];
unsigned char angka;
```

```
// USART Receiver buffer
#define RX_BUFFER_SIZE 8
char rx_buffer[RX_BUFFER_SIZE];

#if RX_BUFFER_SIZE<256
unsigned char rx_wr_index,rx_rd_index,rx_counter;
#else
unsigned int rx_wr_index,rx_rd_index,rx_counter;
#endif

// This flag is set on USART Receiver buffer overflow
bit rx_buffer_overflow;

// USART Receiver interrupt service routine
interrupt [USART_RXC] void usart_rx_isr(void)
{
   char status,data;
   unsigned char i;
   status=UCSRA;
   data=UDR;
   save[angka]=data;
   angka++;
   if(angka == 5)
   {
      for(i=0; i<5; i++)
      {
         lcd_gotoxy(i+4,0);
         lcd_putchar(save[i]);
         lcd_gotoxy(9,0);
         lcd_putsf(" meter");
         delay_ms(1);
      }
      angka =0;
   }

   if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
   {
      rx_buffer[rx_wr_index]=data;
      if (++rx_wr_index == RX_BUFFER_SIZE) rx_wr_index=0;
      if (++rx_counter == RX_BUFFER_SIZE)
      {
         rx_counter=0;
         rx_buffer_overflow=1;
```

```
      };
   };
}

#ifndef _DEBUG_TERMINAL_IO_
// Get a character from the USART Receiver buffer
#define _ALTERNATE_GETCHAR_
#pragma used+
char getchar(void)
{
   char data;
   while (rx_counter==0);
   data = rx_buffer[rx_rd_index];
   if (++rx_rd_index == RX_BUFFER_SIZE) rx_rd_index=0;
   #asm("cli")
   --rx_counter;
   #asm("sei")
   return data;
}
#pragma used-
#endif

// USART Transmitter buffer
#define TX_BUFFER_SIZE 8
char tx_buffer[TX_BUFFER_SIZE];

#if TX_BUFFER_SIZE<256
unsigned char tx_wr_index,tx_rd_index,tx_counter;
#else
unsigned int tx_wr_index,tx_rd_index,tx_counter;
#endif

// USART Transmitter interrupt service routine
interrupt [USART_TXC] void usart_tx_isr(void)
{
  if (tx_counter)
  {
      --tx_counter;
      UDR=tx_buffer[tx_rd_index];
      if (++tx_rd_index == TX_BUFFER_SIZE) tx_rd_index=0;
  };
}
```

```
#ifndef _DEBUG_TERMINAL_IO_
// Write a character to the USART Transmitter buffer
#define _ALTERNATE_PUTCHAR_
#pragma used+
void putchar(char c)
{
   while (tx_counter == TX_BUFFER_SIZE);
   #asm("cli")
   if (tx_counter || ((UCSRA & DATA_REGISTER_EMPTY)==0))
   {
      tx_buffer[tx_wr_index]=c;
      if (++tx_wr_index == TX_BUFFER_SIZE) tx_wr_index=0;
      ++tx_counter;
   }
   else
      UDR=c;
   #asm("sei")
}
#pragma used-
#endif

// Standard Input/Output functions
#include <stdio.h>

void baca_tombol(void)
{
  if(up == 0)
  {    tombol = 'U';
       delay_ms(50);
  }
  else if(down == 0)
  {    tombol = 'D';
       delay_ms(50);
  }
  else if(left == 0)
  {    tombol = 'L';
       delay_ms(50);
  }
  else if(right == 0)
  {    tombol = 'R';
       delay_ms(50);
  }
  else if(forward == 0)
```

```
  {    tombol = 'F';
       delay_ms(50);
  }
  else if(backward == 0)
  {    tombol = 'B';
       delay_ms(50);
  }
  else
  {    tombol = ' ';
       delay_ms(50);
  }
}


void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
   PORTA=0x00;
   DDRA =0x00;

// Port B initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out Func1=Out
Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
   PORTB= 0x00;
   DDRB = 0xFF;

// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
   PORTC= 0x00;
   DDRC = 0x00;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
   PORTD= 0x00;
   DDRD = 0x00;
```

```
// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
   TCCR0= 0x00;
   TCNT0= 0x00;
   OCR0 = 0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
   TCCR1A= 0x00;
   TCCR1B= 0x00;
   TCNT1H= 0x00;
   TCNT1L= 0x00;
   ICR1H = 0x00;
   ICR1L = 0x00;
   OCR1AH= 0x00;
   OCR1AL= 0x00;
   OCR1BH= 0x00;
   OCR1BL= 0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
   ASSR = 0x00;
   TCCR2= 0x00;
   TCNT2= 0x00;
   OCR2 = 0x00;
```

```
// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
    MCUCR = 0x00;
    MCUCSR= 0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
    TIMSK = 0x00;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud Rate: 9600
    UCSRA = 0x00;
    UCSRB = 0xD8;
    UCSRC = 0x86;
    UBRRH = 0x00;
    UBRRL = 0x47;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
    ACSR  = 0x80;
    SFIOR = 0x00;

// LCD module initialization
    angka =0;

    lcd_init(16);
    lcd_clear();

    lcd_gotoxy(0,0);
    lcd_putsf(men_1);
    lcd_gotoxy(0,1);
    lcd_putsf(men_2);
    delay_ms(1500);

    lcd_clear();
    lcd_gotoxy(0,0);
    lcd_putsf(men_3);
```

```
    lcd_gotoxy(0,1);
    lcd_putsf(men_4);
    delay_ms(1500);

    lcd_clear();
    lcd_gotoxy(0,0);
    lcd_putsf(men_13);
    lcd_gotoxy(0,1);
    lcd_putsf(men_14);
    delay_ms(1500);

    lcd_clear();
    lcd_gotoxy(0,0);
    lcd_putsf(men_5);
    lcd_gotoxy(0,1);
    lcd_putsf(men_6);
    delay_ms(1500);


    lcd_clear();
    lcd_gotoxy(0,0);
    lcd_putsf(men_7);
    lcd_gotoxy(0,1);
    lcd_putsf(men_8);
    delay_ms(1500);

    lcd_clear();
    lcd_gotoxy(0,0);
    lcd_putsf(men_9);
    lcd_gotoxy(0,1);
    lcd_putsf(men_10);
    delay_ms(1500);

    lcd_clear();
    lcd_gotoxy(0,0);
    lcd_putsf(men_11);
    lcd_gotoxy(0,1);
    lcd_putsf(men_12);
    delay_ms(2000);

// Global enable interrupts
#asm("sei")
    tombol = ' ';
```

```
while (1)
{
   baca_tombol();
   if(tombol != ' ')
   {
      switch(tombol)
      {
         case 'U':
               lcd_gotoxy(0,1);
               //sprintf(dat_1);
               lcd_putsf(dat_1);
               delay_ms(50);
               putchar('U');
               tombol = ' ';
               break;
         case 'D':
               lcd_gotoxy(0,1);
               //sprintf(dat_2);
               lcd_putsf(dat_2);
               delay_ms(50);
               putchar('D');
               tombol = ' ';
               break;
         case 'L':
               lcd_gotoxy(0,1);
               //sprintf(dat_3);
               lcd_putsf(dat_3);
               delay_ms(50);
               putchar('L');
               tombol = ' ';
               break;
         case 'R':
               lcd_gotoxy(0,1);
               //sprintf(dat_4);
               lcd_putsf(dat_4);
               delay_ms(50);
               putchar('R');
               tombol = ' ';
               break;
         case 'F':
               lcd_gotoxy(0,1);
               lcd_putsf(dat_5);
```

```
                delay_ms(50);
                putchar('F');
                tombol = ' ';
                break;
            case 'B':
                lcd_gotoxy(0,1);
                lcd_putsf(dat_6);
                delay_ms(50);
                putchar('B');
                tombol = ' ';
                break;
            default:
                lcd_gotoxy(0,1);
                lcd_putsf(dat_7);
                delay_ms(50);
                putchar(' ');
                tombol = ' ';
                break;
          }
        }
        else
        {
          lcd_gotoxy(0,1);
          lcd_putsf(dat_7);
          delay_ms(50);
          putchar(' ');
          tombol = ' ';
        }
    };


}
```

Instruksi Program Penerima

```
/*****************************************************
This program was produced by the CodeWizardAVR V2.03

Project                 : R O V
Version         : 1.0
Date            : 11/12/2009
Author          : Asri Asmarariani Putri
Company         : Lab UKM
Comments        : Program Receiver

Chip type                       : ATmega16
Program type            : Application
Clock frequency         : 11.059200 MHz
Memory model            : Small
External RAM size       : 0
Data Stack size         : 256
*****************************************************/

#include <mega16.h>
#include <stdio.h>
#include <delay.h>


#define RXB8 1
#define TXB8 0
#define UPE 2
#define OVR 3
#define FE 4
#define UDRE 5
#define RXC 7

#define FRAMING_ERROR (1<<FE)
#define PARITY_ERROR (1<<UPE)
#define DATA_OVERRUN (1<<OVR)
#define DATA_REGISTER_EMPTY (1<<UDRE)
#define RX_COMPLETE (1<<RXC)

#define mot_kan1        PORTC.0
#define mot_kan2        PORTC.1
#define mot_kir1        PORTC.2
#define mot_kir2        PORTC.3

#define en_mot_kir      PORTC.4
#define en_mot_kan      PORTC.5

#define pom_sed1        PORTD.2
#define pom_sed2        PORTD.3
```

```
#define pom_bua1      PORTD.4
#define pom_bua2      PORTD.5

#define en_pom_bua  PORTD.6
#define en_pom_sed   PORTD.7

// Declare your global variables here
unsigned char save;

unsigned char data1;
unsigned char data2;
unsigned char data3;
unsigned char data4;
unsigned char data_adc;
unsigned int data_buffer;


void diam_diam(void);
void maju(void);
void mundur(void);
void belok_kanan(void);
void belok_kiri(void);
void naik(void);
void turun(void);
void calculate(unsigned char data_buffer_1);

// USART Receiver buffer
#define RX_BUFFER_SIZE 8
char rx_buffer[RX_BUFFER_SIZE];

#if RX_BUFFER_SIZE<256
unsigned char rx_wr_index,rx_rd_index,rx_counter;
#else
unsigned int rx_wr_index,rx_rd_index,rx_counter;
#endif


// This flag is set on USART Receiver buffer overflow
bit rx_buffer_overflow;

// USART Receiver interrupt service routine
interrupt [USART_RXC] void usart_rx_isr(void)
{
   char status,data;
   status = UCSRA;
   data   = UDR;
   save   = data;
```

```
   if(save != 0x20)
   {
    switch(save)
    {
      case 'U':naik();save=0x20;break;
      case 'D':turun();save=0x20;break;
      case 'L':belok_kiri();save=0x20;break;
      case 'R':belok_kanan();save=0x20;break;
      case 'F':maju();save=0x20;break;
      case 'B':mundur();save=0x20;break;
      case ' ':diam_diam();break;
    }
   }
   if ((status & (FRAMING_ERROR | PARITY_ERROR |
DATA_OVERRUN))==0)
   {
      rx_buffer[rx_wr_index]=data;
      if (++rx_wr_index == RX_BUFFER_SIZE) rx_wr_index=0;
      if (++rx_counter == RX_BUFFER_SIZE)
      {
         rx_counter=0;
         rx_buffer_overflow=1;
      };
   };
}

#ifndef _DEBUG_TERMINAL_IO_
// Get a character from the USART Receiver buffer
#define _ALTERNATE_GETCHAR_
#pragma used+
char getchar(void)
{
   char data;
   while (rx_counter==0);
   data = rx_buffer[rx_rd_index];
   if (++rx_rd_index == RX_BUFFER_SIZE) rx_rd_index=0;
   #asm("cli")
   --rx_counter;
   #asm("sei")
   return data;
}
#pragma used-
#endif

// USART Transmitter buffer
#define TX_BUFFER_SIZE 8
char tx_buffer[TX_BUFFER_SIZE];
```

```
#if TX_BUFFER_SIZE<256
unsigned char tx_wr_index,tx_rd_index,tx_counter;
#else
unsigned int tx_wr_index,tx_rd_index,tx_counter;
#endif

// USART Transmitter interrupt service routine
interrupt [USART_TXC] void usart_tx_isr(void)
{
   if (tx_counter)
   {
      --tx_counter;
      UDR = tx_buffer[tx_rd_index];
      if (++tx_rd_index == TX_BUFFER_SIZE) tx_rd_index=0;
   };
}

#ifndef _DEBUG_TERMINAL_IO_
// Write a character to the USART Transmitter buffer
#define _ALTERNATE_PUTCHAR_
#pragma used+
void putchar(char c)
{
   while (tx_counter == TX_BUFFER_SIZE);
   #asm("cli")
   if (tx_counter || ((UCSRA & DATA_REGISTER_EMPTY)==0))
   {
      tx_buffer[tx_wr_index]=c;
      if (++tx_wr_index == TX_BUFFER_SIZE) tx_wr_index=0;
      ++tx_counter;
   }
   else
      UDR=c;
   #asm("sei")
}
#pragma used-
#endif

// Standard Input/Output functions
#define ADC_VREF_TYPE 0x20

// Read the 8 most significant bits
// of the AD conversion result
unsigned char read_adc(unsigned char adc_input)
{
   ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
```

```
// Delay needed for the stabilization of the ADC input voltage
   delay_us(10);
// Start the AD conversion
   ADCSRA|=0x40;
// Wait for the AD conversion to complete
   while ((ADCSRA & 0x10)==0);
   ADCSRA|=0x10;
   return ADCH;
}


void main(void)
{
// Declare your local variables here

// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
   PORTA=0x00;
   DDRA=0x00;


// Port B initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out
Func1=Out Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
   PORTB=0x00;
   DDRB=0xFF;

// Port C initialization
// Func7=Out Func6=Out Func5=Out Func4=Out Func3=Out Func2=Out
Func1=Out Func0=Out
// State7=0 State6=0 State5=0 State4=0 State3=0 State2=0 State1=0 State0=0
   PORTC=0x00;
   DDRC=0xFF;

// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In
Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
   PORTD=0x00;
   DDRD=0xFC;

// Timer/Counter 0 initialization
// Clock source: System Clock
```

```
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
   TCCR0=0x00;
   TCNT0=0x00;
   OCR0=0x00;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
   TCCR1A=0x00;
   TCCR1B=0x00;
   TCNT1H=0x00;
   TCNT1L=0x00;
   ICR1H=0x00;
   ICR1L=0x00;
   OCR1AH=0x00;
   OCR1AL=0x00;
   OCR1BH=0x00;
   OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
   ASSR=0x00;
   TCCR2=0x00;
   TCNT2=0x00;
   OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
   MCUCR=0x00;
   MCUCSR=0x00;
```

```
// Timer(s)/Counter(s) Interrupt(s) initialization
   TIMSK=0x00;

// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: On
// USART Mode: Asynchronous
// USART Baud Rate: 9600
   UCSRA=0x00;
   UCSRB=0xD8;
   UCSRC=0x86;
   UBRRH=0x00;
   UBRRL=0x47;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
   ACSR=0x80;
   SFIOR=0x00;

// ADC initialization
// ADC Clock frequency: 345.600 kHz
// ADC Voltage Reference: AREF pin
// ADC Auto Trigger Source: None
// Only the 8 most significant bits of
// the AD conversion result are used
   ADMUX=ADC_VREF_TYPE & 0xff;
   ADCSRA=0x85;

// Global enable interrupts
#asm("sei")

   while (1)
   {
   };
}

void diam_diam(void)
{
   delay_ms(10);
   en_pom_sed = 0;
   en_pom_bua = 0;

   en_mot_kir   = 0;
   en_mot_kan = 0;
   delay_ms(10);
```

```
}

void maju(void)
{
   en_mot_kir   = 1;
   en_mot_kan = 1;

   en_pom_sed = 0;
   en_pom_bua = 0;

   mot_kan1 = 1;
   mot_kan2 = 0;
   mot_kir1 = 1;
   mot_kir2 = 0;
   delay_ms(50);
   diam_diam();
}

void mundur(void)
{
   en_mot_kir   = 1;
   en_mot_kan = 1;

   en_pom_sed = 0;
   en_pom_bua = 0;
   mot_kan1 = 0;
   mot_kan2 = 1;
   mot_kir1   = 0;
   mot_kir2   = 1;
   delay_ms(50);
   diam_diam();
}

void belok_kanan(void)
{
   en_mot_kir    = 0;
   en_mot_kan  = 1;
   en_pom_sed = 0;
   en_pom_bua = 0;

   mot_kan1 = 1;
   mot_kan2 = 0;
   mot_kir1   = 0;
   mot_kir2   = 0;
   delay_ms(50);
   diam_diam();
}
```

```
void belok_kiri(void)
{
   en_mot_kir   = 1;
   en_mot_kan = 0;
   en_pom_sed = 0;
   en_pom_bua = 0;
   mot_kan1 = 0;
   mot_kan2 = 0;
   mot_kir1   = 1;
   mot_kir2   = 0;
   delay_ms(50);
   diam_diam();
}

void naik(void)
{
   en_pom_sed   = 0;
   en_pom_bua   = 1;
   en_mot_kir      = 0;
   en_mot_kan    = 0;
   pom_sed1        = 0;
   pom_sed2        = 0;
   pom_bua1       = 1;
   pom_bua2       = 0;
   delay_ms(50);
   data_adc = read_adc(1);
   calculate(data_adc);
   diam_diam();
}

void turun(void)
{
   en_pom_sed  = 1;
   en_pom_bua  = 0;
   en_mot_kir      = 0;
   en_mot_kan    = 0;
   pom_sed1  = 1;
   pom_sed2  = 0;
   pom_bua1 = 1;
   pom_bua2 = 0;
   delay_ms(50);
   data_adc = read_adc(1);
   calculate(data_adc);
   diam_diam();
}
```

```
void calculate(unsigned char data_buffer_1)
{
   data_buffer = (1-(256/ data_buffer))/14;
   if (data_buffer_1 < 100)
   {
      data1 =0x20;
      data2 =0x30;
      data3 =(data_buffer/10)+0x30;
      data4 =(data_buffer%10)+0x30;
      putchar(data1);
      putchar(data2);
      putsf(".");
      putchar(data3);
      putchar(data4);
   }
   else if(data_buffer < 1000)
   {
      data1 = 0x20;
      data2 =(data_buffer/100)+0x30;
      data3 =((data_buffer%100)/10)+0x30;
      data4 =(data_buffer%10)+0x30;
      putchar(data1);
      putchar(data2);
      putsf(".");
      putchar(data3);
      putchar(data4);
   }
   else
   {
      data1 =(data_buffer/1000)+0x30;
      data2 =((data_buffer%1000)/100)+0x30;
      data3 =((data_buffer%100)/10)+0x30;
      data4 =(data_buffer%10)+0x30;
      putchar(data1);
      putchar(data2);
      putsf(".");
      putchar(data3);
      putchar(data4);
    }
}
```

LAMPIRAN C

DATA SHEET

DATA SHEET

X-BEE PRO

XBee™/XBee-PRO™ OEM RF Modules

Product Manual v1.xAx - 802.15.4 Protocol
For OEM RF Module Part Numbers: XB24-...-001, XBP24-...-001

IEEE® 802.15.4 OEM RF Modules by MaxStream, Inc.

MaxStream
355 South 520 West, Suite 180
Lindon, UT 84042
Phone: (801) 765-9885
Fax: (801) 765-9895
rf-xperts@maxstream.net
www.MaxStream.net (live chat support)

M100232
2006.10.13

1. XBee/XBee-PRO OEM RF Modules

The XBee and XBee-PRO OEM RF Modules were engineered to meet IEEE 802.15.4 standards and support the unique needs of low-cost, low-power wireless sensor networks. The modules require minimal power and provide reliable delivery of data between devices.

The modules operate within the ISM 2.4 GHz frequency band and are pin-for-pin compatible with each other.

1.1. Key Features

Long Range Data Integrity

XBee
- Indoor/Urban: up to 100' (30 m)
- Outdoor line-of-sight: up to 300' (100 m)
- Transmit Power: 1 mW (0 dBm)
- Receiver Sensitivity: -92 dBm

XBee-PRO
- Indoor/Urban: up to 300' (100 m)
- Outdoor line-of-sight: up to 1 mile (1500 m)
- Transmit Power: 100 mW (20 dBm) EIRP
- Receiver Sensitivity: -100 dBm

RF Data Rate: 250,000 bps

Low Power

XBee
- TX Current: 45 mA (@3.3 V)
- RX Current: 50 mA (@3.3 V)
- Power-down Current: < 10 µA

XBee-PRO
- TX Current: 215 mA (@3.3 V)
- RX Current: 55 mA (@3.3 V)
- Power-down Current: < 10 µA

ADC and I/O line support

Analog-to-digital conversion, Digital I/O

I/O Line Passing

Advanced Networking & Security

Retries and Acknowledgements

DSSS (Direct Sequence Spread Spectrum)

Each direct sequence channels has over 65,000 unique network addresses available

Source/Destination Addressing

Unicast & Broadcast Communications

Point-to-point, point-to-multipoint and peer-to-peer topologies supported

Coordinator/End Device operations

Easy-to-Use

No configuration necessary for out-of box RF communications

Free X-CTU Software (Testing and configuration software)

AT and API Command Modes for configuring module parameters

Extensive command set

Small form factor

Free & Unlimited RF-XPert Support

1.1.1. Worldwide Acceptance

FCC Approval (USA) Refer to Appendix A [p57] for FCC Requirements. Systems that contain XBee/XBee-PRO RF Modules inherit MaxStream Certifications.

ISM (Industrial, Scientific & Medical) **2.4 GHz frequency band**

Manufactured under **ISO 9001:2000** registered standards

XBee/XBee-PRO RF Modules are optimized for use in the **United States, Canada, Australia, Israel and Europe**. Contact MaxStream for complete list of government agency approvals.

1.2. Specifications

Table 1-01. Specifications of the XBee/XBee-PRO OEM RF Modules

Specification	XBee	XBee-PRO
Performance		
Indoor/Urban Range	up to 100 ft. (30 m)	Up to 300' (100 m)
Outdoor RF line-of-sight Range	up to 300 ft. (100 m)	Up to 1 mile (1500 m)
Transmit Power Output (software selectable)	1mW (0 dBm)	60 mW (18 dBm) conducted, 100 mW (20 dBm) EIRP*
RF Data Rate	250,000 bps	250,000 bps
Serial Interface Data Rate (software selectable)	1200 - 115200 bps (non-standard baud rates also supported)	1200 - 115200 bps (non-standard baud rates also supported)
Receiver Sensitivity	-92 dBm (1% packet error rate)	-100 dBm (1% packet error rate)
Power Requirements		
Supply Voltage	2.8 – 3.4 V	2.8 – 3.4 V
Transmit Current (typical)	45mA (@ 3.3 V)	If PL=0 (10dBm): 137mA(@3.3V), 139mA(@3.0V) PL=1 (12dBm): 155mA (@3.3V), 153mA(@3.0V) PL=2 (14dBm): 170mA (@3.3V), 171mA(@3.0V) PL=3 (16dBm): 188mA (@3.3V), 195mA(@3.0V) PL=4 (18dBm): 215mA (@3.3V), 227mA(@3.0V)
Idle / Receive Current (typical)	50mA (@ 3.3 V)	55mA (@ 3.3 V)
Power-down Current	< 10 µA	< 10 µA
General		
Operating Frequency	ISM 2.4 GHz	ISM 2.4 GHz
Dimensions	0.960" x 1.087" (2.438cm x 2.761cm)	0.960" x 1.297" (2.438cm x 3.294cm)
Operating Temperature	-40 to 85° C (industrial)	-40 to 85° C (industrial)
Antenna Options	Integrated Whip, Chip or U.FL Connector	Integrated Whip, Chip or U.FL Connector
Networking & Security		
Supported Network Topologies	Point-to-point, Point-to-multipoint & Peer-to-peer	
Number of Channels (software selectable)	16 Direct Sequence Channels	12 Direct Sequence Channels
Addressing Options	PAN ID, Channel and Addresses	PAN ID, Channel and Addresses
Agency Approvals		
United States (FCC Part 15.247)	OUR-XBEE	OUR-XBEEPRO
Industry Canada (IC)	4214A XBEE	4214A XBEEPRO
Europe (CE)	ETSI	ETSI (Max. 10 dBm transmit power output)*
Japan	n/a	005NYCA0378 (Max. 10 dBm transmit power output)**

* When operating in Europe: XBee-PRO RF Modules must be configured to operate at a maximum transmit power output level of 10 dBm. The power output level is set using the PL command. The PL parameter must equal "0" (10 dBm).

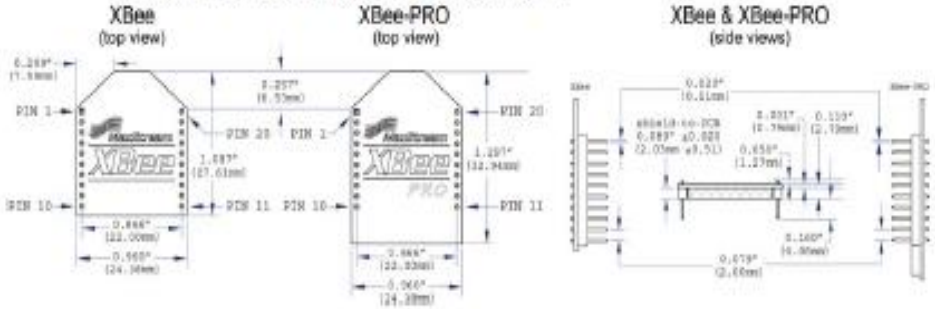
Additionally, European regulations stipulate an EIRP power maximum of 12.86 dBm (19 mW) for the XBee-PRO and 12.11 dBm for the XBee when integrating high-gain antennas.

** When operating in Japan: Transmit power output is limited to 10 dBm. A special part number is required when ordering modules approved for use in Japan. Contact MaxStream for more information [call 1-801-765-9885 or send e-mails to sales@maxstream.net].

Antenna Options: The ranges specified are typical when using the integrated Whip (1.5 dBi) and Dipole (2.1 dBi) antennas. The Chip antenna option provides advantages in its form factor; however, it typically yields shorter range than the Whip and Dipole antenna options when transmitting outdoors. For more information, refer to the "XBee Antenna" application note located on MaxStream's web site (http://www.maxstream.net/support/knowledgebase/article.php?kb=158).

1.3. Mechanical Drawings

Figure 1-01. Mechanical drawings of the XBee/XBee-PRO OEM RF Modules (antenna options not shown)
The XBee and XBee-PRO RF Modules are pin-for-pin compatible.



1.4. Mounting Considerations

The XBee/XBee-PRO RF Module was designed to mount into a receptacle (socket) and therefore does not require any soldering when mounting it to a board. The XBee Development Kits contain RS-232 and USB interface boards which use two 20-pin receptacles to receive modules.

Figure 1-02. XBee Module Mounting to an RS-232 Interface Board.

The receptacles used on MaxStream development boards are manufactured by Century Interconnect. Several other manufacturers provide comparable mounting solutions; however, MaxStream currently uses the following receptacles:

- Through-hole single-row receptacles -
  Samtec P/N: MMS-110-01-L-SV (or equivalent)
- Surface-mount double-row receptacles -
  Century Interconnect P/N: CPRMSL20-D-0-1 (or equivalent)
- Surface-mount single-row receptacles -
  Samtec P/N: SMM-110-02-SM-S

MaxStream also recommends printing an outline of the module on the board to indicate the orientation the module should be mounted.

1.5. Pin Signals

Figure 1-03. XBee/XBee-PRO RF Module Pin Numbers
(top sides shown - shields on bottom)

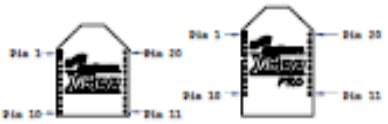


Table 1-02. Pin Assignments for the XBee and XBee-PRO Modules
(Low-asserted signals are distinguished with a horizontal line above signal name.)

Pin #	Name	Direction	Description
1	VCC	-	Power supply
2	DOUT	Output	UART Data Out
3	DIN / CONFIG	Input	UART Data In
4	DO8*	Output	Digital Output 8
5	RESET	Input	Module Reset (reset pulse must be at least 200 ns)
6	PWM0 / RSSI	Output	PWM Output 0 / RX Signal Strength Indicator
7	PWM1	Output	PWM Output 1
8	[reserved]	-	Do not connect
9	$\overline{\text{DTR}}$ / SLEEP_RQ / DI8	Input	Pin Sleep Control Line or Digital Input 8
10	GND	-	Ground
11	AD4 / DIO4	Either	Analog Input 4 or Digital I/O 4
12	CTS / DIO7	Either	Clear-to-Send Flow Control or Digital I/O 7
13	ON / SLEEP	Output	Module Status Indicator
14	VREF	Input	Voltage Reference for A/D Inputs
15	Associate / AD5 / DIO5	Either	Associated Indicator, Analog Input 5 or Digital I/O 5
16	RTS / AD6 / DIO6	Either	Request-to-Send Flow Control, Analog Input 6 or Digital I/O 6
17	AD3 / DIO3	Either	Analog Input 3 or Digital I/O 3
18	AD2 / DIO2	Either	Analog Input 2 or Digital I/O 2
19	AD1 / DIO1	Either	Analog Input 1 or Digital I/O 1
20	AD0 / DIO0	Either	Analog Input 0 or Digital I/O 0

* Function is not supported at the time of this release

Design Notes:

- Minimum connections: VCC, GND, DOUT & DIN
- Minimum connections for updating firmware: VCC, GND, DIN, DOUT, RTS & DTR
- Signal Direction is specified with respect to the module
- Module includes a 50k Ω pull-up resistor attached to $\overline{\text{RESET}}$
- Several of the input pull-ups can be configured using the PR command
- Unused pins should be left disconnected

1.6. Electrical Characteristics

Table 1-03. DC Characteristics (VCC = 2.8 - 3.4 VDC)

Symbol	Characteristic	Condition	Min	Typical		Max	Unit
V_{IL}	Input Low Voltage	All Digital Inputs	-	-		0.35 * VCC	V
V_{IH}	Input High Voltage	All Digital Inputs	0.7 * VCC	-		-	V
V_{OL}	Output Low Voltage	I_{OL} = 2 mA, VCC >= 2.7 V	-	-		0.5	V
V_{OH}	Output High Voltage	I_{OH} = -2 mA, VCC >= 2.7 V	VCC - 0.5	-		-	V
I_{IN}	Input Leakage Current	V_{IN} = VCC or GND, all inputs, per pin	-	0.025		1	µA
I_{OZ}	High Impedance Leakage Current	V_{IN} = VCC or GND, all I/O High-Z, per pin	-	0.025		1	µA
TX	Transmit Current	VCC = 3.3 V	-	45 (XBee)	215 (PRO)	-	mA
RX	Receive Current	VCC = 3.3 V	-	50 (XBee)	55 (PRO)	-	mA
PWR-DWN	Power-down Current	SM parameter = 1	-	< 10		-	µA

Table 1-04. ADC Characteristics (Operating)

Symbol	Characteristic	Condition	Min	Typical	Max	Unit
V_{REFH}	VREF - Analog-to-Digital converter reference range		2.08	-	V_{DDAD}	V
I_{REF}	VREF - Reference Supply Current	Enabled	-	200	-	µA
		Disabled or Sleep Mode	-	< 0.01	0.02	µA
V_{INDC}	Analog Input Voltage[1]		V_{SSAD} - 0.3	-	V_{DDAD} + 0.3	V

1. Maximum electrical operating range, not valid conversion range.

Table 1-05. ADC Timing/Performance Characteristics[1]

Symbol	Characteristic	Condition	Min	Typical	Max	Unit
R_{AS}	Source Impedance at Input[2]		-	-	10	kΩ
V_{AIN}	Analog Input Voltage[3]		V_{REFL}		V_{REFH}	V
RES	Ideal Resolution (1 LSB)[4]	2.08V ≤ V_{DDAD} ≤ 3.6V	2.031	-	3.516	mV
DNL	Differential Non-linearity[5]		-	±0.5	±1.0	LSB
INL	Integral Non-linearity[6]		-	±0.5	±1.0	LSB
E_{ZS}	Zero-scale Error[7]		-	±0.4	±1.0	LSB
E_{FS}	Full-scale Error[8]		-	±0.4	±1.0	LSB
E_{IL}	Input Leakage Error[9]		-	±0.05	±5.0	LSB
E_{TU}	Total Unadjusted Error[10]		-	±1.1	±2.5	LSB

1. All ACCURACY numbers are based on processor and system being in WAIT state (very little activity and no IO switching) and that adequate low-pass filtering is present on analog input pins (filter with 0.01 µF to 0.1 µF capacitor between analog input and VREFL). Failure to observe these guidelines may result in system or microcontroller noise causing accuracy errors which will vary based on board layout and the type and magnitude of the activity.
Data transmission and reception during data conversion may cause some degradation of these specifications, depending on the number and timing of packets. It is advisable to test the ADCs in your installation if best accuracy is required.
2. R_{AS} is the real portion of the impedance of the network driving the analog input pin. Values greater than this amount may not fully charge the input circuitry of the ATD resulting in accuracy error.
3. Analog input must be between V_{REFL} and V_{REFH} for valid conversion. Values greater than V_{REFH} will convert to $3FF.
4. The resolution is the ideal step size or 1LSB = (V_{REFH}–V_{REFL})/1024
5. Differential non-linearity is the difference between the current code width and the ideal code width (1LSB). The current code width is the difference in the transition voltages to and from the current code.
6. Integral non-linearity is the difference between the transition voltage to the current code and the adjusted ideal transition voltage for the current code. The adjusted ideal transition voltage is (Current Code–1/2)*(1/(((V_{REFH}+E_{FS})–(V_{REFL}+E_{ZS}))).
7. Zero-scale error is the difference between the transition to the first valid code and the ideal transition to that code. The ideal transition voltage to a given code is (Code–1/2)*(1/(V_{REFH}–V_{REFL})).
8. Full-scale error is the difference between the transition to the last valid code and the ideal transition to that code. The ideal transition voltage to a given code is (Code–1/2)*(1/(V_{REFH}–V_{REFL})).
9. Input leakage error is error due to input leakage across the real portion of the impedance of the network driving the analog pin. Reducing the impedance of the network reduces this error.
10. Total unadjusted error is the difference between the transition voltage to the current code and the ideal straight-line transfer function. This measure of error includes inherent quantization error (1/2LSB) and circuit error (differential, integral, zero-scale, and full-scale) error. The specified value of E_{TU} assumes zero E_{IL} (no leakage or zero real source impedance).

2. RF Module Operation

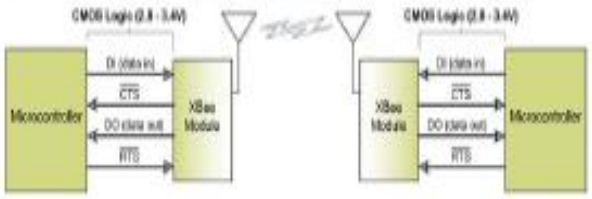
2.1. Serial Communications

The XBee/XBee-PRO OEM RF Modules interface to a host device through a logic-level asynchronous serial port. Through its serial port, the module can communicate with any logic and voltage compatible UART; or through a level translator to any serial device (For example: Through a MaxStream proprietary RS-232 or USB interface board).

2.1.1. UART Data Flow

Devices that have a UART interface can connect directly to the pins of the RF module as shown in the figure below.

Figure 2-01. System Data Flow Diagram in a UART-interfaced environment
(Low-asserted signals distinguished with horizontal line over signal name.)

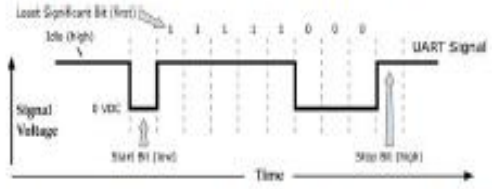


Serial Data

Data enters the module UART through the DI pin (pin 3) as an asynchronous serial signal. The signal should idle high when no data is being transmitted.

Each data byte consists of a start bit (low), 8 data bits (least significant bit first) and a stop bit (high). The following figure illustrates the serial bit pattern of data passing through the module.

Figure 2-02. UART data packet 0x1F (decimal number "31") as transmitted through the RF module
Example Data Format is 8-N-1 (bits - parity - # of stop bits)



The module UART performs tasks, such as timing and parity checking, that are needed for data communications. Serial communications depend on the two UARTs to be configured with compatible settings (baud rate, parity, start bits, stop bits, data bits).

2.1.2. Transparent Operation

By default, XBee/XBee-PRO RF Modules operate in Transparent Mode. When operating in this mode, the modules act as a serial line replacement - all UART data received through the DI pin is queued up for RF transmission. When RF data is received, the data is sent out the DO pin.

Serial-to-RF Packetization

Data is buffered in the DI buffer until one of the following causes the data to be packetized and transmitted:

1. No serial characters are received for the amount of time determined by the RO (Packetization Timeout) parameter. If RO = 0, packetization begins when a character is received.
2. The maximum number of characters that will fit in an RF packet (100) is received.
3. The Command Mode Sequence (GT + CC + GT) is received. Any character buffered in the DI buffer before the sequence is transmitted.

If the module cannot immediately transmit (for instance, if it is already receiving RF data), the serial data is stored in the DI Buffer. The data is packetized and sent at any RO timeout or when 100 bytes (maximum packet size) are received.

If the DI buffer becomes full, hardware or software flow control must be implemented in order to prevent overflow (loss of data between the host and module).

2.1.3. API Operation

API (Application Programming Interface) Operation is an alternative to the default Transparent Operation. The frame-based API extends the level to which a host application can interact with the networking capabilities of the module.

When in API mode, all data entering and leaving the module is contained in frames that define operations or events within the module.

Transmit Data Frames (received through the DI pin (pin 3)) include:

- RF Transmit Data Frame
- Command Frame (equivalent to AT commands)

Receive Data Frames (sent out the DO pin (pin 2)) include:

- RF-received data frame
- Command response
- Event notifications such as reset, associate, disassociate, etc.

The API provides alternative means of configuring modules and routing data at the host application layer. A host application can send data frames to the module that contain address and payload information instead of using command mode to modify addresses. The module will send data frames to the application containing status packets; as well as source, RSSI and payload information from received data packets.

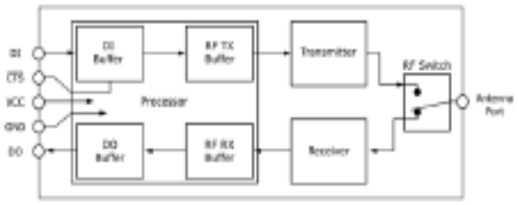
The API operation option facilitates many operations such as the examples cited below:

-> Transmitting data to multiple destinations without entering Command Mode
-> Receive success/failure status of each transmitted RF packet
-> Identify the source address of each received packet

To implement API operations, refer to API sections [p52].

2.1.4. Flow Control

Figure 2-03. Internal Data Flow Diagram



DI (Data In) Buffer

When serial data enters the RF module through the DI pin (pin 3), the data is stored in the DI Buffer until it can be processed.

Hardware Flow Control ($\overline{\text{CTS}}$). When the DI buffer is 17 bytes away from being full; by default, the module de-asserts $\overline{\text{CTS}}$ (high) to signal to the host device to stop sending data [refer to D7 (DIO7 Configuration) parameter]. $\overline{\text{CTS}}$ is re-asserted after the DI Buffer has 34 bytes of memory available.

How to eliminate the need for flow control:

1. Send messages that are smaller than the DI buffer size.
2. Interface at a lower baud rate [BD (Interface Data Rate) parameter] than the throughput data rate.

Case in which the DI Buffer may become full and possibly overflow:

If the module is receiving a continuous stream of RF data, any serial data that arrives on the DI pin is placed in the DI Buffer. The data in the DI buffer will be transmitted over-the-air when the module is no longer receiving RF data in the network.

Refer to the RO (Packetization Timeout), BD (Interface Data Rate) and D7 (DIO7 Configuration) command descriptions for more information.

DO (Data Out) Buffer

When RF data is received, the data enters the DO buffer and is sent out the serial port to a host device. Once the DO Buffer reaches capacity, any additional incoming RF data is lost.

Hardware Flow Control ($\overline{\text{RTS}}$). If $\overline{\text{RTS}}$ is enabled for flow control (D6 (DIO6 Configuration) Parameter = 1), data will not be sent out the DO Buffer as long as $\overline{\text{RTS}}$ (pin 16) is de-asserted.

Two cases in which the DO Buffer may become full and possibly overflow:

1. If the RF data rate is set higher than the interface data rate of the module, the module will receive data from the transmitting module faster than it can send the data to the host.
2. If the host does not allow the module to transmit data out from the DO buffer because of being held off by hardware or software flow control.

Refer to the D6 (DIO6 Configuration) command description for more information.

2.2. ADC and Digital I/O Line Support

The XBee/XBee-PRO RF Modules support ADC (Analog-to-digital conversion) and digital I/O line passing. The following pins support multiple functions:

Table 2-01. Pin functions and their associated pin numbers and commands
AD = Analog-to-Digital Converter, DIO = Digital Input/Output
Pin functions not applicable to this section are denoted within (parenthesis).

Pin Function	Pin#	AT Command
AD0 / DIO0	20	D0
AD1 / DIO1	19	D1
AD2 / DIO2	18	D2
AD3 / DIO3 / (COORD_SEL)	17	D3
AD4 / DIO4	11	D4
AD5 / DIO5 / (ASSOCIATE)	15	D5
DIO6 / (RTS)	16	D6
DIO7 / (CTS)	12	D7
DI8 / (DTR) / (Sleep_RQ)	9	D8

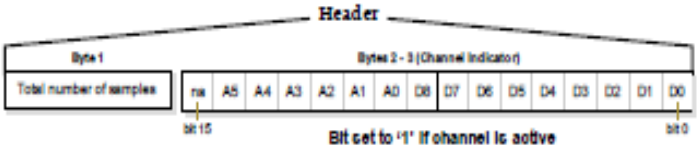
To enable ADC and DIO pin functions:

For ADC Support:	Set ATDn = 2
For Digital Input support:	Set ATDn = 3
For Digital Output Low support:	Set ATDn = 4
For Digital Output High support:	Set ATDn = 5

2.2.1. I/O Data Format

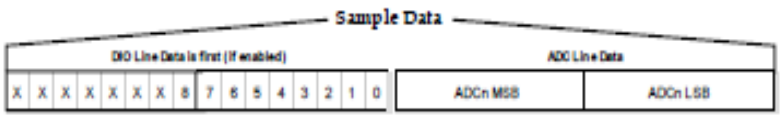
I/O data begins with a header. The first byte of the header defines the number of samples forthcoming. A sample is comprised of input data and the inputs can contain either DIO or ADC. The last 2 bytes of the header (Channel Indicator) define which inputs are active. Each bit represents either a DIO line or ADC channel.

Figure 2-04. Header



Sample data follows the header and the channel indicator frame is used to determine how to read the sample data. If any of the DIO lines are enabled, the first 2 bytes are the DIO data and the ADC data follows. ADC channel data is stored as an unsigned 10-bit value right-justified on a 16-bit boundary.

Figure 2-05. Sample Data



2.2.2. API Support

I/O data is sent out the UART using an API frame. All other data can be sent and received using Transparent Operation [refer to p10] or API framing if API mode is enabled (AP > 0).

API Operations support two RX (Receive) frame identifiers for I/O data:

- 0x82 for RX (Receive) Packet: 64-bit address I/O
- 0x83 for RX (Receive) Packet: 16-bit address I/O

The API command header is the same as shown in the "RX (Receive) Packet: 64-bit Address" and "RX (Receive) Packet: 64-bit Address" API types [refer to p56]. RX data follows the format described in the I/O Data Format section [p12].

Applicable Commands: AP (API Enable)

2.2.3. Sleep Support

When an RF module wakes, it will always do a sample based on any active ADC or DIO lines. This allows sampling based on the sleep cycle whether it be Cyclic Sleep (SM parameter = 4 or 5) or Pin Sleep (SM = 1 or 2). To gather more samples when awake, set the IR (Sample Rate) parameter.

For Cyclic Sleep modes: If the IR parameter is set, the module will stay awake until the IT (Samples before TX) parameter is met. The module will stay awake for ST (Time before Sleep) time.

Applicable Commands: IR (Sample Rate), IT (Samples before TX), SM (Sleep Mode), IC (DIO Change Detect)

2.2.4. DIO Pin Change Detect

When "DIO Change Detect" is enabled (using the IC command), DIO lines 0-7 are monitored. When a change is detected on a DIO line, the following will occur:

1. An RF packet is sent with the updated DIO pin levels. This packet will not contain any ADC samples.
2. Any queued samples are transmitted before the change detect data. This may result in receiving a packet with less than IT (Samples before TX) samples.

Note: Change detect will not affect Pin Sleep wake-up. The D8 pin (DTR/Sleep_RQ/DI8) is the only line that will wake a module from Pin Sleep. If not all samples are collected, the module will still enter Sleep Mode after a change detect packet is sent.

Applicable Commands: IC (DIO Change Detect), IT (Samples before TX)

NOTE: Change detect is only supported when the Dx (DIOx Configuration) parameter equals 3,4 or 5.

2.2.5. Sample Rate (Interval)

The Sample Rate (Interval) feature allows enabled ADC and DIO pins to be read periodically on modules that are not configured to operate in Sleep Mode. When one of the Sleep Modes is enabled and the IR (Sample Rate) parameter set, the module will stay awake until IT (Samples before TX) samples have been collected.

Once a particular pin is enabled, the appropriate sample rate must be chosen. The maximum sample rate that can be achieved while using one A/D line is 1 sample/ms or 1 KHz (Note that the modem will not be able to keep up with transmission when IR & IT are equal to "1").

Applicable Commands: IR (Sample Rate), IT (Samples before TX), SM (Sleep Mode)

2.2.6. I/O Line Passing

Virtual wires can be set up between XBee/XBee-PRO Modules. When an RF data packet is received that contains I/O data, the receiving module can be setup to update any enabled outputs (PWM and DIO) based on the data it receives.

Note that I/O lines are mapped in pairs. For example: AD0 can only update PWM0 and DI5 can only update DO5). The default setup is for outputs not to be updated, which results in the I/O data being sent out the UART (refer to the IU (Enable I/O Output) command). To enable the outputs to be updated, the IA (I/O Input Address) parameter must be setup with the address of the module that has the appropriate inputs enabled. This effectively binds the outputs to a particular module's input. This does not affect the ability of the module to receive I/O line data from other modules - only its ability to update enabled outputs. The IA parameter can also be setup to accept I/O data for output changes from any module by setting the IA parameter to 0xFFFF.

When outputs are changed from their non-active state, the module can be setup to return the output level to it non-active state. The timers are set using the Tn (Dn Output Timer) and PT (PWM Output Timeout) commands. The timers are reset every time a valid I/O packet (passed IA check) is received. The IC (Change Detect) and IR (Sample Rate) parameters can be setup to keep the output set to their active output if the system needs more time than the timers can handle.

Note: DI8 can not be used for I/O line passing.

Applicable Commands: IA (I/O Input Address), Tn (Dn Output Timeout), P0 (PWM0 Configuration), P1 (PWM1 Configuration), M0 (PWM0 Output Level), M1 (PWM1 Output Level), PT (PWM Output Timeout), RP (RSSSI PWM Timer)

2.2.7. Configuration Example

As an example for a simple A/D link, a pair of RF modules could be set as follows:

Remote Configuration	Base Configuration
DL = 0x1234	DL = 0x5678
MY = 0x5678	MY = 0x1234
D0 = 2	P0 = 2
D1 = 2	P1 = 2
IR = 0x14	IU = 1
IT = 5	IA = 0x5678 (or 0xFFFF)

These settings configure the remote module to sample AD0 and AD1 once each every 20 ms. It then buffers 5 samples each before sending them back to the base module. The base should then receive a 32-Byte transmission (20 Bytes data and 12 Bytes framing) every 100 ms.

2.3. XBee/XBee-PRO Networks

The following IEEE 802.15.4 network types are supported by the XBee/XBee-PRO RF modules:

- NonBeacon
- NonBeacon (w/ Coordinator)

The following terms will be used to explicate the network operations:

Table 2-02. Terms and definitions

Term	Definition
PAN	Personal Area Network - A data communication network that includes one or more End Devices and optionally a Coordinator.
Coordinator	A Full-function device (FFD) that provides network synchronization by polling nodes (NonBeacon (w/ Coordinator) networks only)
End Device	*When in the same network as a Coordinator* - RF modules that rely on a Coordinator for synchronization and can be put into states of sleep for low-power applications.
Association	The establishment of membership between End Devices and a Coordinator. Association is only applicable in NonBeacon (w/Coordinator) networks.

2.3.1. NonBeacon

By default, XBee/XBee-PRO RF Modules are configured to support NonBeacon communications. NonBeacon systems operate within a Peer-to-Peer network topology and therefore are not dependent upon Master/Slave relationships. This means that modules remain synchronized without use of master/server configurations and each module in the network shares both roles of master and slave. MaxStream's peer-to-peer architecture features fast synchronization times and fast cold start times. This default configuration accommodates a wide range of RF data applications.

Figure 2-06. NonBeacon Peer-to-Peer Architecture

A peer-to-peer network can be established by configuring each module to operate as an End Device (CE = 0), disabling End Device Association on all modules (A1 = 0) and setting ID and CH parameters to be identical across the network.

2.3.2. NonBeacon (w/ Coordinator)

A device is configured as a Coordinator by setting the CE (Coordinator Enable) parameter to "1". Coordinator power-up is governed by the A2 (Coordinator Association) parameter.

In a NonBeacon (w/ Coordinator) system, the Coordinator can be configured to use direct or indirect transmissions. If the SP (Cyclic Sleep Period) parameter is set to "0", the Coordinator will send data immediately. Otherwise, the SP parameter determines the length of time the Coordinator will retain the data before discarding it. Generally, SP (Cyclic Sleep Period) and ST (Time before Sleep) parameters should be set to match the SP and ST settings of the End Devices.

Association plays a critical role in the implementation of a NonBeacon (w/ Coordinator) system. Refer to the Association section [next page] for more information.

2.3.3. Association

Association is the establishment of membership between End Devices and a Coordinator and is only applicable in NonBeacon (w/ Coordinator) networks. The establishment of membership is useful in scenarios that require a central unit (Coordinator) to relay messages to or gather data from several remote units (End Devices), assign channels or assign PAN IDs.

An RF data network that consists of one Coordinator and one or more End Devices forms a PAN (Personal Area Network). Each device in a PAN has a PAN Identifier [ID (PAN ID) parameter]. PAN IDs must be unique to prevent miscommunication between PANs. The Coordinator PAN ID is set using the ID (PAN ID) and A2 (Coordinator Association) commands.

An End Device can associate to a Coordinator without knowing the address, PAN ID or channel of the Coordinator. The A1 (End Device Association) parameter bit fields determine the flexibility of an End Device during association. The A1 parameter can be used for an End Device to dynamically set its destination address, PAN ID and/or channel.

For example: If the PAN ID of a Coordinator is known, but the operating channel is not; the A1 command on the End Device should be set to enable the 'Auto_Associate' and 'Reassign_Channel' bits. Additionally, the ID parameter should be set to match the PAN ID of the associated Coordinator.

Coordinator / End Device Setup and Operation

To configure a module to operate as a Coordinator, set the CE (Coordinator Enable) parameter to '1'. Set the CE parameter of End Devices to '0' (default). Coordinator and End Devices should contain matching firmware versions.

NonBeacon (w/ Coordinator) Systems

In a NonBeacon (w/ Coordinator) system, the Coordinator can be configured to use direct or indirect transmissions. If the SP (Cyclic Sleep Period) parameter is set to '0', the Coordinator will send data immediately. Otherwise, the SP parameter determines the length of time the Coordinator will retain the data before discarding it. Generally, SP (Cyclic Sleep Period) and ST (Time before Sleep) parameters should be set to match the SP and ST settings of the End Devices.

Coordinator Power-up

Coordinator power-up is governed by the A2 (Coordinator Association) command. On power-up, the Coordinator undergoes the following sequence of events:

1. **Check A2 parameter- Reassign_PANID Flag**

   Set (bit 0 = 1) - The Coordinator issues an Active Scan. The Active Scan selects one channel and transmits a BeaconRequest command to the broadcast address (0xFFFF) and broadcast PAN ID (0xFFFF). It then listens on that channel for beacons from any Coordinator operating on that channel. The listen time on each channel is determined by the SD (Scan Duration) parameter value.

   Once the time expires on that channel, the Active Scan selects another channel and again transmits the BeaconRequest as before. This process continues until all channels have been scanned, or until 5 PANs have been discovered. When the Active Scan is complete, the results include a list of PAN IDs and Channels that are being used by other PANs. This list is used to assign an unique PAN ID to the new Coordinator. The ID parameter will be retained if it is not found in the Active Scan results. Otherwise, the ID (PAN ID) parameter setting will be updated to a PAN ID that was not detected.

   Not Set (bit 0 = 0) - The Coordinator retains its ID setting. No Active Scan is performed.

2. Check A2 parameter - Reassign_Channel Flag (bit 1)

Set (bit 1 = 1) - The Coordinator issues an Energy Scan. The Energy Scan selects one channel and scans for energy on that channel. The duration of the scan is specified by the SD (Scan Duration) parameter. Once the scan is completed on a channel, the Energy Scan selects the next channel and begins a new scan on that channel. This process continues until all channels have been scanned.

When the Energy Scan is complete, the results include the maximal energy values detected on each channel. This list is used to determine a channel where the least energy was detected. If an Active Scan was performed (Reassign_PANID Flag set), the channels used by the detected PANs are eliminated as possible channels. Thus, the results of the Energy Scan and the Active Scan (if performed) are used to find the best channel (channel with the least energy that is not used by any detected PAN). Once the best channel has been selected, the CH (Channel) parameter value is updated to that channel.

Not Set (bit 1 = 0) - The Coordinator retains its CH setting. An Energy Scan is not performed.

3. Start Coordinator

The Coordinator starts on the specified channel (CH parameter) and PAN ID (ID parameter). Note, these may be selected in steps 1 and/or 2 above. The Coordinator will only allow End Devices to associate to it if the A2 parameter "AllowAssociation" flag is set. Once the Coordinator has successfully started, the Associate LED will blink 1 time per second. (The LED is solid if the Coordinator has not started.)

4. Coordinator Modifications

Once a Coordinator has started:
Modifying the A2 (Reassign_Channel or Reassign_PANID bits), ID, CH or MY parameters will cause the Coordinator's MAC to reset (The Coordinator RF module (including volatile RAM) is not reset). Changing the A2 AllowAssociation bit will not reset the Coordinator's MAC. In a non-beaconing system, End Devices that associated to the Coordinator prior to a MAC reset will have knowledge of the new settings on the Coordinator. Thus, if the Coordinator were to change its ID, CH or MY settings, the End Devices would no longer be able to communicate with the non-beacon Coordinator. Once a Coordinator has started, the ID, CH, MY or A2 (Reassign_Channel or Reassign_PANID bits) should not be changed.

End Device Power-up

End Device power-up is governed by the A1 (End Device Association) command. On power-up, the End Device undergoes the following sequence of events:

1. Check A1 parameter - AutoAssociate Bit

Set (bit 2 = 1) - End Device will attempt to associate to a Coordinator. (refer to steps 2-3).

Not Set (bit 2 = 0) - End Device will not attempt to associate to a Coordinator. The End Device will operate as specified by its ID, CH and MY parameters. Association is considered complete and the Associate LED will blink quickly (5 times per second). When the AutoAssociate bit is not set, the remaining steps (2-3) do not apply.

2. Discover Coordinator (if Auto-Associate Bit Set)

The End Device issues an Active Scan. The Active Scan selects one channel and transmits a BeaconRequest command to the broadcast address (0xFFFF) and broadcast PAN ID (0xFFFF). It then listens on that channel for beacons from any Coordinator operating on that channel. The listen time on each channel is determined by the SD parameter.

Once the time expires on that channel, the Active Scan selects another channel and again transmits the BeaconRequest command as before. This process continues until all channels have been scanned, or until 5 PANs have been discovered. When the Active Scan is complete, the results include a list of PAN IDs and Channels that are being used by detected PANs.

The End Device selects a Coordinator to associate with according to the A1 parameter "Reassign_PANID" and "Reassign_Channel" flags:

Reassign_PANID Bit Set (bit 0 = 1)- End Device can associate with a PAN with any ID value.

Reassign_PANID Bit Not Set (bit 0 = 0) - End Device will only associate with a PAN whose ID setting matches the ID setting of the End Device.

Reassign_Channel Bit Set (bit 1 = 1) - End Device can associate with a PAN with any CH value.

Reassign_Channel Bit Not Set (bit 1 = 0)- End Device will only associate with a PAN whose CH setting matches the CH setting of the End Device.

After applying these filters to the discovered Coordinators, if multiple candidate PANs exist, the End Device will select the PAN whose transmission link quality is the strongest. If no valid Coordinator is found, the End Device will either go to sleep (as dictated by its SM (Sleep Mode) parameter) or retry Association.

Note - An End Device will also disqualify Coordinators if they are not allowing association (A2 - AllowAssociation bit); or, if the Coordinator is not using the same NonBeacon scheme as the End Device. (They must both be programmed with NonBeacon code.)

3. Associate to Valid Coordinator

Once a valid Coordinator is found (step 2), the End Device sends an AssociationRequest message to the Coordinator. It then waits for an AssociationConfirmation to be sent from the Coordinator. Once the Confirmation is received, the End Device is Associated and the Associate LED will blink rapidly (2 times per second). The LED is solid if the End Device has not associated.

4. End Device Changes once an End Device has associated

Changing A1, ID or CH parameters will cause the End Device to disassociate and restart the Association procedure.

If the End Device fails to associate, the AI command can give some indication of the failure.

2.4. XBee/XBee-PRO Addressing

Every RF data packet sent over-the-air contains a Source Address and Destination Address field in its header. The RF module conforms to the 802.15.4 specification and supports both short 16-bit addresses and long 64-bit addresses. A unique 64-bit IEEE source address is assigned at the factory and can be read with the SL (Serial Number Low) and SH (Serial Number High) commands. Short addressing must be configured manually. A module will use its unique 64-bit address as its Source Address if its MY (16-bit Source Address) value is "0xFFFF" or "0xFFFE".

To send a packet to a specific module using 64-bit addressing: Set Destination Address (DL + DH) to match the Source Address (SL + SH) of the intended destination module.

To send a packet to a specific module using 16-bit addressing: Set DL (Destination Address Low) parameter to equal the MY parameter and set the DH (Destination Address High) parameter to '0'.

2.4.1. Unicast Mode

By default, the RF module operates in Unicast Mode. Unicast Mode is the only mode that supports retries. While in this mode, receiving modules send an ACK (acknowledgement) of RF packet reception to the transmitter. If the transmitting module does not receive the ACK, it will re-send the packet up to three times or until the ACK is received.

Short 16-bit addresses. The module can be configured to use short 16-bit addresses as the Source Address by setting (MY < 0xFFFE). Setting the DH parameter (DH = 0) will configure the Destination Address to be a short 16-bit address (if DL < 0xFFFE). For two modules to communicate using short addressing, the Destination Address of the transmitter module must match the MY parameter of the receiver.

The following table shows a sample network configuration that would enable Unicast Mode communications using short 16-bit addresses.

Table 2-03. Sample Unicast Network Configuration (using 16-bit addressing)

Parameter	RF Module 1	RF Module 2
MY (Source Address)	0x01	0x02
DH (Destination Address High)	0	0
DL (Destination Address Low)	0x02	0x01

Long 64-bit addresses. The RF module's serial number (SL parameter concatenated to the SH parameter) can be used as a 64-bit source address when the MY (16-bit Source Address) parameter is disabled. When the MY parameter is disabled (set MY = 0xFFFF or 0xFFFE), the module's source address is set to the 64-bit IEEE address stored in the SH and SL parameters.

When an End Device associates to a Coordinator, its MY parameter is set to 0xFFFE to enable 64-bit addressing. The 64-bit address of the module is stored as SH and SL parameters. To send a packet to a specific module, the Destination Address (DL + DH) on one module must match the Source Address (SL + SH) of the other.

2.4.2. Broadcast Mode

Any RF module within range will accept a packet that contains a broadcast address. When configured to operate in Broadcast Mode, receiving modules do not send ACKs (Acknowledgements) and transmitting modules do not automatically re-send packets as is the case in Unicast Mode.

To send a broadcast packet to all modules regardless of 16-bit or 64-bit addressing, set the destination addresses of all the modules as shown below.

Sample Network Configuration (All modules in the network):

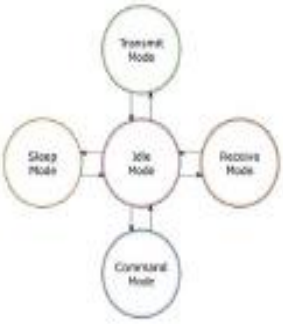
- DL (Destination Low Address) = 0x0000FFFF
- DH (Destination High Address) = 0x00000000 (default value)

NOTE: When programming the module, parameters are entered in hexadecimal notation (without the "0x" prefix). Leading zeros may be omitted.

2.5. Modes of Operation

XBee/XBee-PRO RF Modules operate in five modes.

Figure 2-07. Modes of Operation



2.5.1. Idle Mode

When not receiving or transmitting data, the RF module is in Idle Mode. The module shifts into the other modes of operation under the following conditions:

- Transmit Mode (Serial data is received in the DI Buffer)
- Receive Mode (Valid RF data is received through the antenna)
- Sleep Mode (Sleep Mode condition is met)
- Command Mode (Command Mode Sequence is issued)

2.5.2. Transmit/Receive Modes

RF Data Packets

Each transmitted data packet contains a Source Address and Destination Address field. The Source Address matches the address of the transmitting module as specified by the MY (Source Address) parameter (if MY >= 0xFFFE), the SH (Serial Number High) parameter or the SL (Serial Number Low) parameter. The <Destination Address> field is created from the DH (Destination Address High) and DL (Destination Address Low) parameter values. The Source Address and/or Destination Address fields will either contain a 16-bit short or long 64-bit long address.

The RF data packet structure follows the 802.15.4 specification.

[Refer to the XBee/XBee-PRO Addressing section for more information]

Direct and Indirect Transmission

There are two methods to transmit data:

- Direct Transmission - data is transmitted immediately to the Destination Address
- Indirect Transmission - A packet is retained for a period of time and is only transmitted after the destination module (Source Address = Destination Address) requests the data.

Indirect Transmissions can only occur on a Coordinator. Thus, if all nodes in a network are End Devices, only Direct Transmissions will occur. Indirect Transmissions are useful to ensure packet delivery to a sleeping node. The Coordinator currently is able to retain up to 2 indirect messages.

Direct Transmission

A NonBeaconing Coordinator can be configured to use only Direct Transmission by setting the SP (Cyclic Sleep Period) parameter to "0". Also, a NonBeaconing Coordinator using indirect transmissions will revert to direct transmission if it knows the destination module is awake.

To enable this behavior, the ST (Time before Sleep) value of the Coordinator must be set to match the ST value of the End Device. Once the End Device either transmits data to the Coordinator or polls the Coordinator for data, the Coordinator will use direct transmission for all subsequent data transmissions to that module address until ST time (or number of beacons) occurs with no activity (at which point it will revert to using indirect transmissions for that module address). "No activity" means no transmission or reception of messages with a specific address. Global messages will not reset the ST timer.

Indirect Transmission

To configure Indirect Transmissions in a PAN (Personal Area Network), the SP (Cyclic Sleep Period) parameter value on the Coordinator must be set to match the longest sleep value of any End Device. The SP parameter represents time in NonBeacon systems and beacons in Beacon-enabled systems. The sleep period value on the Coordinator determines how long (time or number of beacons) the Coordinator will retain an indirect message before discarding it.

In NonBeacon networks, an End Device must poll the Coordinator once it wakes from Sleep to determine if the Coordinator has an indirect message for it. For Cyclic Sleep Modes, this is done automatically every time the module wakes (after SP time). For Pin Sleep Modes, the A1 (End Device Association) parameter value must be set to enable Coordinator polling on pin wake-up. Alternatively, an End Device can use the FP (Force Poll) command to poll the Coordinator as needed.

CCA (Clear Channel Assessment)

Prior to transmitting a packet, a CCA (Clear Channel Assessment) is performed on the channel to determine if the channel is available for transmission. The detected energy on the channel is compared with the CA (Clear Channel Assessment) parameter value. If the detected energy exceeds the CA parameter value, the packet is not transmitted.

Also, a delay is inserted before a transmission takes place. This delay is settable using the RN (Backoff Exponent) parameter. If RN is set to "0", then there is no delay before the first CCA is performed. The RN parameter value is the equivalent of the "minBE" parameter in the 802.15.4 specification. The transmit sequence follows the 802.15.4 specification.

By default, the MM (MAC Mode) parameter = 0. On a CCA failure, the module will attempt to resend the packet up to two additional times.

When in Unicast packets with RR (Retries) = 0, the module will execute two CCA retries. Broadcast packets always get two CCA retries.

Acknowledgement

If the transmission is not a broadcast message, the module will expect to receive an acknowledgement from the destination node. If an acknowledgement is not received, the packet will be resent up to 3 more times. If the acknowledgement is not received after all transmissions, an ACK failure is recorded.

2.5.3. Sleep Mode

Sleep Modes enable the RF module to enter states of low-power consumption when not in use. In order to enter Sleep Mode, one of the following conditions must be met (in addition to the module having a non-zero SM parameter value):

- Sleep_RQ (pin 9) is asserted.
- The module is idle (no data transmission or reception) for the amount of time defined by the ST (Time before Sleep) parameter. [NOTE: ST is only active when SM = 4-5.]

Table 2-04. Sleep Mode Configurations

Sleep Mode Setting	Transition into Sleep Mode	Transition out of Sleep Mode (wake)	Characteristics	Related Commands	Power Consumption
Pin Hibernate (SM = 1)	Assert (high) Sleep_RQ (pin 9)	De-assert (low) Sleep_RQ	Pin/Host-controlled / NonBeacon systems only / Lowest Power	(SM)	< 10 µA (@3.0 VCC)
Pin Doze (SM = 2)	Assert (high) Sleep_RQ (pin 9)	De-assert (low) Sleep_RQ	Pin/Host-controlled / NonBeacon systems only / Fastest wake-up	(SM)	< 50 µA
Cyclic Sleep (SM = 4 - 5)	Automatic transition to Sleep Mode as defined by the SM (Sleep Mode) and ST (Time before Sleep) parameters.	Transition occurs after the cyclic sleep time interval elapses. The time interval is defined by the SP (Cyclic Sleep Period) parameter.	RF module wakes in pre-determined time intervals to detect if RF data is present / When SM = 5, NonBeacon systems only	(SM), SP, ST	< 50 µA when sleeping

The SM command is central to setting Sleep Mode configurations. By default, Sleep Modes are disabled (SM = 0) and the module remains in Idle/Receive Mode. When in this state, the module is constantly ready to respond to serial or RF activity.

Higher Voltages. Sleep Mode current consumption is highly sensitive to voltage. Voltages above 3.0V will cause much higher current consumption.

Table 2-05. Sample Sleep Mode Currents

	XBee			XBee-PRO		
Vcc (V)	SM=1	SM=2	SM=4,5	SM=1	SM=2	SM=4,5
2.8–3.0	<3 µA	<35uA	<34uA	<4uA	<34uA	<34uA
3.1	6uA	37mA	36uA	12uA	39uA	37uA
3.2	35uA	48uA	49uA	45uA	60uA	55uA
3.3	101uA	83uA	100uA	130uA	115uA	120uA
3.4	255uA	170uA	240uA	310uA	260uA	290uA

Pin/Host-controlled Sleep Modes

The transient current when waking from pin sleep (SM = 1 or 2) does not exceed the idle current of the module. The current ramps up exponentially to its idle current.

Pin Hibernate (SM = 1)

- Pin/Host-controlled
- Typical power-down current: < 10 µA (@3.0 VCC)
- Wake-up time: 13.2 msec

Pin Hibernate Mode minimizes quiescent power (power consumed when in a state of rest or inactivity). This mode is voltage level-activated; when Sleep_RQ is asserted, the module will finish any transmit, receive or association activities, enter Idle Mode and then enter a state of sleep. The module will not respond to either serial or RF activity while in pin sleep.

To wake a sleeping module operating in Pin Hibernate Mode, de-assert Sleep_RQ (pin 9). The module will wake when Sleep_RQ is de-asserted and is ready to transmit or receive when the $\overline{\text{CTS}}$ line is low. When waking the module, the pin must be de-asserted at least two 'byte times' after $\overline{\text{CTS}}$ goes low. This assures that there is time for the data to enter the DI buffer.

Pin Doze (SM = 2)

- Pin/Host-controlled
- Typical power-down current: < 50 µA
- Wake-up time: 2 msec

Pin Doze Mode functions as does Pin Hibernate Mode; however, Pin Doze features faster wake-up time and higher power consumption.

To wake a sleeping module operating in Pin Doze Mode, de-assert Sleep_RQ (pin 9). The module will wake when Sleep_RQ is de-asserted and is ready to transmit or receive when the $\overline{\text{CTS}}$ line is low. When waking the module, the pin must be de-asserted at least two 'byte times' after $\overline{\text{CTS}}$ goes low. This assures that there is time for the data to enter the DI buffer.

Cyclic Sleep Modes

Cyclic Sleep Remote (SM = 4)

- Typical Power-down Current: < 50 µA (when asleep)
- Wake-up time: 2 msec

The Cyclic Sleep Modes allow modules to periodically check for RF data. When the SM parameter is set to '4', the module is configured to sleep, then wakes once a cycle to check for data from a module configured as a Cyclic Sleep Coordinator (SM = 0, CE = 1). The Cyclic Sleep Remote sends a poll request to the coordinator at a specific interval set by the SP (Cyclic Sleep Period) parameter. The coordinator will transmit any queued data addressed to that specific remote upon receiving the poll request.

If no data is queued for the remote, the coordinator will not transmit and the remote will return to sleep for another cycle. If queued data is transmitted back to the remote, it will stay awake to allow for back and forth communication until the ST (Time before Sleep) timer expires.

Also note that $\overline{\text{CTS}}$ will go low each time the remote wakes, allowing for communication initiated by the remote host if desired.

Cyclic Sleep Remote with Pin Wake-up (SM = 5)

Use this mode to wake a sleeping remote module through either the RF interface or by the de-assertion of Sleep_RQ for event-driven communications. The cyclic sleep mode works as described above (Cyclic Sleep Remote) with the addition of a pin-controlled wake-up at the remote module. The Sleep_RQ pin is edge-triggered, not level-triggered. The module will wake when a low is detected then set $\overline{\text{CTS}}$ low as soon as it is ready to transmit or receive.

Any activity will reset the ST (Time before Sleep) timer so the module will go back to sleep only after there is no activity for the duration of the timer. Once the module wakes (pin-controlled), further pin activity is ignored. The module transitions back into sleep according to the ST time regardless of the state of the pin.

[Cyclic Sleep Coordinator (SM = 6)]

- Typical current = Receive current
- Always awake

NOTE: The SM=6 parameter value exists solely for backwards compatibility with firmware version 1.x60. If backwards compatibility with the older firmware version is not required, always use the CE (Coordinator Enable) command to configure a module as a Coordinator.

This mode configures a module to wake cyclic sleeping remotes through RF interfacing. The Coordinator will accept a message addressed to a specific remote 16 or 64-bit address and hold it in a buffer until the remote wakes and sends a poll request. Messages not sent directly (buffered and requested) are called "Indirect messages". The Coordinator only queues one indirect message at a time. The Coordinator will hold the indirect message for a period 2.5 times the sleeping period indicated by the SP (Cyclic Sleep Period) parameter. The Coordinator's SP parameter should be set to match the value used by the remotes.

2.5.4. Command Mode

To modify or read RF Module parameters, the module must first enter into Command Mode - a state in which incoming characters are interpreted as commands. Two Command Mode options are supported: AT Command Mode [refer to section below] and API Command Mode [p52].

AT Command Mode

To Enter AT Command Mode:

Send the 3-character command sequence "+++" and observe guard times before and after the command characters. [Refer to the "Default AT Command Mode Sequence" below.]

Default AT Command Mode Sequence (for transition to Command Mode):

- No characters sent for one second [GT (Guard Times) parameter = 0x3E8]
- Input three plus characters ("+++") within one second [CC (Command Sequence Character) parameter = 0x2B.]
- No characters sent for one second [GT (Guard Times) parameter = 0x3E8]

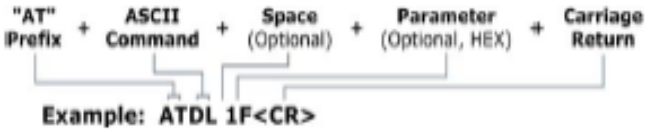
All of the parameter values in the sequence can be modified to reflect user preferences.

NOTE: Failure to enter AT Command Mode is most commonly due to baud rate mismatch. Ensure the 'Baud' setting on the "PC Settings" tab matches the interface data rate of the RF module. By default, the BD parameter = 3 (9600 bps).

To Send AT Commands:

Send AT commands and parameters using the syntax shown below.

Figure 2-08. Syntax for sending AT Commands



To read a parameter value stored in the RF module's register, omit the parameter field.

The preceding example would change the RF module Destination Address (Low) to "0x1F". To store the new value to non-volatile (long term) memory, subsequently send the WR (Write) command.

For modified parameter values to persist in the module's registry after a reset, changes must be saved to non-volatile memory using the WR (Write) Command. Otherwise, parameters are restored to previously saved values after the module is reset.

System Response. When a command is sent to the module, the module will parse and execute the command. Upon successful execution of a command, the module returns an "OK" message. If execution of a command results in an error, the module returns an "ERROR" message.

To Exit AT Command Mode:

1. Send the ATCN (Exit Command Mode) command (followed by a carriage return).

   [OR]

2. If no valid AT Commands are received within the time specified by CT (Command Mode Timeout) Command, the RF module automatically returns to Idle Mode.

For an example of programming the RF module using AT Commands and descriptions of each configurable parameter, refer to the RF Module Configuration chapter [p25].

DATA SHEET

FSR-01

State-of-the-Art Pointing Solutions for the OEM

FSR®
Force Sensing Resistor® Integration Guide and Evaluation Parts Catalog

400 Series Evaluation Parts
With Suggested Electrical Interfaces

INTERLINK ELECTRONICS
546 Flynn Road • Camarillo, CA 93012
(805) 484-1331 • Fax (805) 484-8989
http://www.interlinkelectronics.com

INTERLINK ELECTRONICS

Force Sensing Resistors
An Overview of the Technology

Force Sensing Resistors (FSR) are a polymer thick film (PTF) device which exhibits a decrease in resistance with an increase in the force applied to the active surface. Its force sensitivity is optimized for use in human touch control of electronic devices. FSRs are not a load cell or strain gauge, though they have similar properties. FSRs are not suitable for precision measurements.

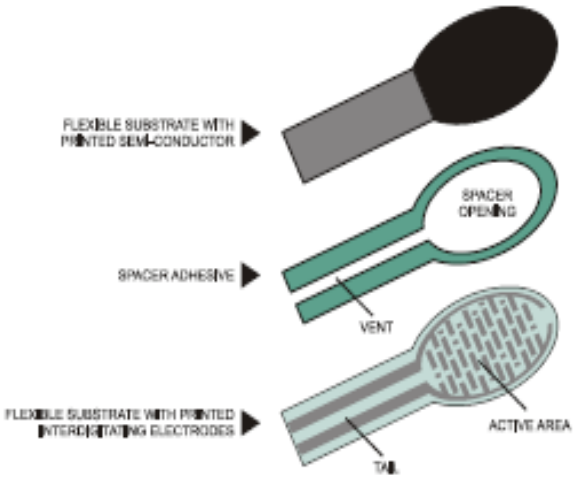


Figure 1: FSR Construction

Force vs. Resistance

The force vs. resistance characteristic shown in Figure 2 provides an overview of FSR typical response behavior. For interpretational convenience, the force vs. resistance data is plotted on a log/log format. These data are representative of our typical devices, with this particular force-resistance characteristic being the response of evaluation part # 402 (0.5" [12.7 mm] diameter circular active area). A stainless steel actuator with a 0.4" [10.0 mm] diameter hemispherical tip of 60 durometer polyurethane rubber was used to actuate the FSR device. In general, FSR response approximately follows an inverse power-law characteristic (roughly 1/R).

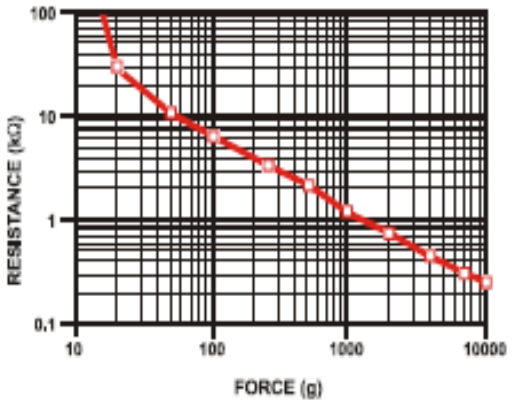


Figure 2: Resistance vs. Force

Referring to Figure 2, at the low force end of the force-resistance characteristic, a switch-like response is evident. This turn-on threshold, or "break force", that swings the resistance from greater than 100 kΩ to about 10 kΩ (the beginning of the dynamic range that follows a power-law) is determined by the substrate and overlay thickness and flexibility, size and shape of the actuator, and spacer-adhesive thickness (the gap between the facing conductive elements). Break force increases with increasing substrate and overlay rigidity, actuator size, and spacer-adhesive thickness. Eliminating the adhesive, or keeping it well away from the area where the force is being applied, such as the center of a large FSR device, will give it a lower rest resistance (e.g. stand-off resistance).

At the high force end of the dynamic range, the response deviates from the power-law behavior, and eventually saturates to a point where increases in force yield little or no decrease in resist-ance. Under these conditions of Figure 2, this saturation force is beyond 10 kg. The saturation point is more a function of pressure than force. The saturation pressure of a typical FSR is on the order of 100 to 200 psi. For the data shown in Figures 2, 3 and 4, the actual measured pressure range is 0 to 175 psi (0 to 22 lbs applied over 0.125 in^2). Forces higher than the saturation force can be measured by spreading the force over a greater area; the overall pressure is then kept below the saturation point, and dynamic response is maintained. However, the converse of this effect is also true, smaller actuators will saturate FSRs earlier in the dynamic range, since the saturation point is reached at a lower force.

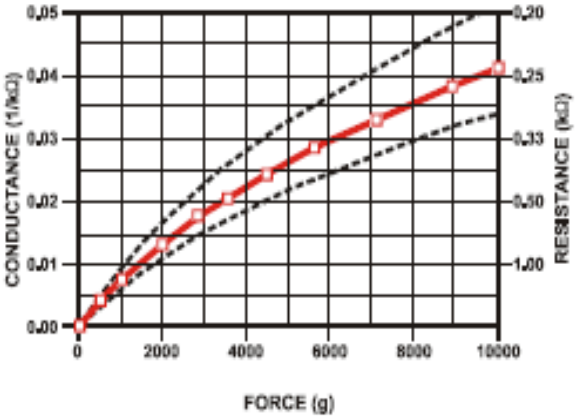


Figure 3:
Conductance vs. Force (0-10Kg)

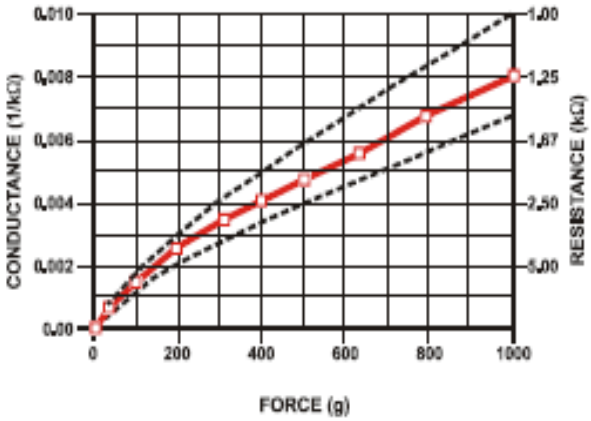


Figure 4:
Conductance vs. Force (0-1Kg) Low Force Range

Force vs. Conductance

In Figure 3, the conductance is plotted vs. force (the inverse of resistance: 1/r). This format allows interpretation on a linear scale. For reference, the corresponding resistance values are also included on the right vertical axis. A simple circuit called a current-to-voltage converter (see page 21) gives a voltage output directly proportional to FSR conductance and can be useful where response linearity is desired. Figure 3 also includes a typical part-to-part repeatability envelope. This error band determines the maximum accuracy of any general force measurement. The spread or width of the band is strongly dependent on the repeatability of any actuating and measuring system, as well as the repeatability tolerance held by Interlink Electronics during FSR production. Typically, the part-to-part repeatability tolerance held during manufacturing ranges from ± 15% to ± 25% of an established nominal resistance.

Figure 4 highlights the 0-1 kg (0-2.2 lbs) range of the conductance-force characteristic. As in Figure 3, the corresponding resistance values are included for reference. This range is common to human interface applications. Since the conductance response in this range is fairly linear, the force resolution will be uniform and data interpretation simplified. The typical part-to-part error band is also shown for this touch range. In most human touch control applications this error is insignificant, since human touch is fairly inaccurate. Human factors studies have shown that in this force range repeatability errors of less than ± 50% are difficult to discern by touch alone.

FSR®
Force Sensing Resistors™

FSR Integration Notes
A Step-by-Step Guide to Optimal Use

For best results, follow these seven steps when beginning any new product design, proof-of-concept, technology evaluation, or first prototype implementation:

1. Start with Reasonable Expectations (Know Your Sensor)

The FSR sensor is not a strain gauge, load cell or pressure transducer. While it can be used for dynamic measurement, only qualitative results are generally obtainable. Force accuracy ranges from approximately ± 5% to ± 25% depending on the consistency of the measurement and actuation system, the repeatability tolerance held in manufacturing, and the use of part calibration.

Accuracy should not be confused with resolution. The force resolution of FSR devices is better than ± 0.5% of full use force.

2. Choose the Sensor that Best Fits the Geometry of Your Application

Usually sensor size and shape are the limiting parameters in FSR integration, so any evaluation part should be chosen to fit the desired mechanical actuation system. In general, standard FSR products have a common semiconductor make-up and only by varying actuation methods (e.g. overlays and actuator areas) or electrical interfaces can different response characteristics be achieved.

3. Set-up a Repeatable and Reproducible Mechanical Actuation System

When designing the actuation mechanics, follow these guidelines to achieve the best force repeatability:

- Provide a consistent force distribution. FSR response is very sensitive to the distribution of the applied force. In general, this precludes the use of dead weights for characterization since exact duplication of the weight distribution is rarely repeatable cycle-to-cycle. A consistent weight (force) distribution is more difficult to achieve than merely obtaining a consistent total applied weight (force). As long as the distribution is the same cycle-to-cycle, then repeatability will be maintained. The use of a thin elastomer between the applied force and the FSR can help absorb error from inconsistent force distributions.

- Keep the actuator area, shape, and compliance constant. Charges in these parameters significantly alter the response characteristic of a given sensor. Any test, mock-up, or evaluation conditions should be closely matched to the final use conditions. The greater the cycle-to-cycle consistency of these parameters, the greater the device repeatability. In human interface applications where a finger is the mode of actuation, perfect control of these parameters is not generally possible. However, human force sensing is somewhat inaccurate; it is rarely sensitive enough to detect differences of less than ± 50%.

- Control actuator placement. In cases where the actuator is to be smaller than the FSR active area, cycle-to-cycle consistency of actuator placement is necessary. (Caution: FSR layers are held together by an adhesive that surrounds the electrically active areas. If force is applied over an area which includes the adhesive, the resulting response characteristic will be drastically altered.) In an extreme case (e.g., a large, flat, hard actuator that bridges the bordering adhesive), the adhesive can present FSR actuation

- Keep actuation cycle time consistent. Because of the time dependence of the FSR resistance to an applied force, it is important when characterizing the sensor system to assure that increasing loads (e.g. force ramps) are applied at consistent rates (cycle-to-cycle). Likewise, static force measurements must take into account FSR mechanical setting time. This time is dependent on the mechanics of actuation and the amount of force applied and is usually on the order of seconds.

4. Use the Optimal Electronic Interface

In most product designs, the critical characteristic is Force vs. Output Voltage, which is controlled by the choice of interface electronics. A variety of interface solutions are detailed in the TechNote section of this guide. Summarized here are some suggested circuits for common FSR applications.

- For FSR Pressure or Force Switches, use the simple interfaces detailed on pages 16 and 17.

- For dynamic FSR measurements or Variable Controls, a current-to-voltage converter (see pages 18 and 19) is recommended. This circuit produces an output voltage that is inversely proportional to FSR resistance. Since the FSR resistance is roughly inversely proportional to applied force, the end result is a direct proportionality between force and voltage; in other words, this circuit gives roughly linear increases in output voltage for increases in applied force. This linearization of the response optimizes the resolution and simplifies data interpretation.

5. Develop a Nominal Voltage Curve and Error Spread

When a repeatable and reproducible system has been established, data from a group of FSR parts can be collected. Test several FSR parts in the system. Record the output voltage at various pre-selected force points throughout the range of interest. Once a family of curves is obtained, a nominal force vs. output voltage curve and the total force accuracy of the system can be determined.

6. Use Part Calibration if Greater Accuracy is Required

For applications requiring the highest obtainable force accuracy, part calibration will be necessary. Two methods can be utilized: gain and offset trimming, and curve fitting.

- Gain and offset trimming can be used as a simple method of calibration. The reference voltage and feedback resistor of the current-to-voltage converter are adjusted for each FSR to pull their responses closer to the nominal curve.

- Curve fitting is the most complete calibration method. A parametric curve fit is done for the nominal curve of a set of FSR devices, and the resultant equation is stored for future use. Fit parameters are then established for each individual FSR (or sending element in an array) in the set. These parameters, along with the measured sensor resistance (or voltage), are inserted into the equation to obtain the force reading. If needed, temperature compensation can also be included in the equation.

7. Refine the System

Spurious results can normally be traced to sensor error or system error. If you have any questions, contact Interlink Electronics' Sales Engineers to discuss your system and final data.

FSR®
Force Sensing Resistors™

FSR Usage Tips
The Do's and Don'ts

- **Do** follow the seven steps of the FSR Integration Guide.
- **Do**, if possible, use a firm, flat and smooth mounting surface.
- **Do** be careful if applying FSR devices to curved surfaces. Pre-loading of the device can occur as the two opposed layers are forced into contact by the bending tension. The device will still function, but the dynamic range may be reduced and resistance drift could occur. The degree of curvature over which an FSR can be bent is a function of the size of the active area. The smaller the active area, the less effect a given curvature will have on the FSR's response.
- **Do** avoid air bubbles and contamination when laminating the FSR to any surface. Use only thin, uniform adhesives, such as Scotch® brand double-sided laminating adhesives. Cover the entire surface of the sensor.
- **Do** be careful of kinks or dents in active areas. They can cause false triggering of the sensors.
- **Do** protect the device from sharp objects. Use an overlay, such as a polycarbonate film or an elastomer, to prevent gouging of the FSR device.
- **Do** use soft rubber or a spring as part of the actuator in designs requiring some travel.

- **Do not** kink or crease the tail of the FSR device if you are bending it; this can cause breaks in the printed silver traces. The smallest suggested bend radius for the tails of evaluation parts is about 0.1" [2.5 mm]. In custom sensor designs, tails have been made that bend over radii of 0.03" (0.8 mm]. Also, be careful if bending the tail near the active area. This can cause stress on the active area and may result in pre-loading and false readings.
- **Do not** block the vent. FSR devices typically have an air vent that runs from the open active area down the length of the tail and out to the atmosphere. This vent assures pressure equilibrium with the environment, as well as allowing even loading and unloading of the device. Blocking this vent could cause FSRs to respond to any actuation in a non-repeatable manner. Also note, that if the device is to be used in a pressure chamber, the vented end will need to be kept vented to the outside of the chamber. This allows for the measurement of the differential pressure.
- **Do not** solder directly to the exposed silver traces. With flexible substrates, the solder joint will not hold and the substrate can easily melt and distort during the soldering. Use Interlink Electronics' standard connection techniques, such as solderable tabs, housed female contacts, Z-axis conductive tapes, or ZIF (zero insertion force) style connectors.
- **Do not** use cyanoacrylate adhesives (e.g. Krazy Glue®) and solder flux removing agents. These degrade the substrate and can lead to cracking.
- **Do not** apply excessive shear force. This can cause delamination of the layers.
- **Do not** exceed 1mA of current per square centimeter of applied force (actuator area). This can irreversibly damage the device.

Evaluation Parts

Descriptions and Dimensions

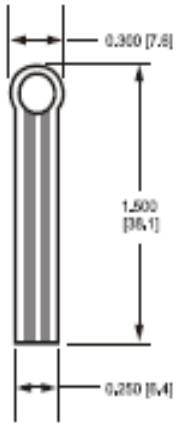


Figure 5:
Part No. 400 (0.2" Circle)

Active Area: 0.2" [5.0] diameter

Nominal Thickness: 0.012" [0.30 mm]

Material Build:

Semiconductive layer
0.004" [0.10] PES
Spacer adhesive
0.002" [0.05] Acrylic
Conductive layer
0.004" [0.10] PES
Rear adhesive
0.002" [0.05] Acrylic
Connector options
a. No connector
b. Solder Tabs (not shown)
c. AMP Female connector

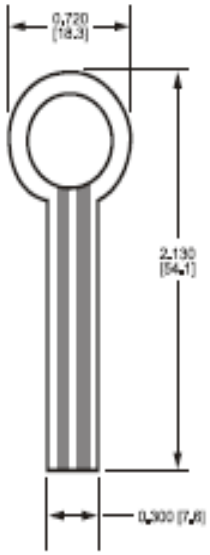


Figure 6:
Part No. 402 (0.5" Circle)

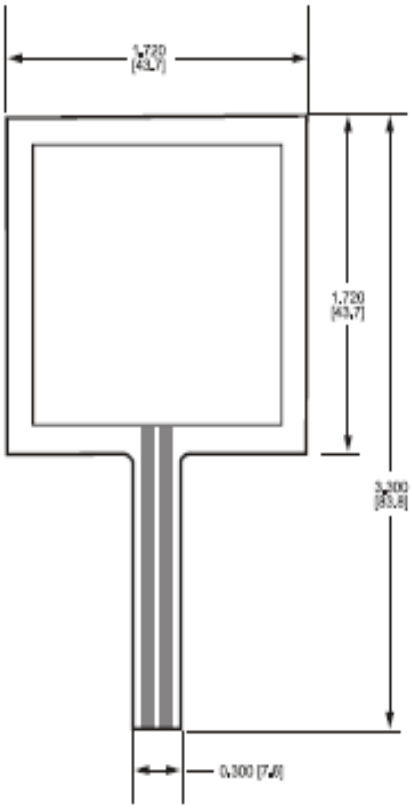
Active Area: 0.5" [12.7] diameter

Nominal thickness: 0.018" [0.46 mm]

Material Build:

Semiconductive Layer
0.005" [0.13] Ultem
Spacer Adhesive
0.006" [0.15] Acrylic
Conductive Layer
0.005" [0.13] Ultem
Rear Adhesive
0.002" [0.05] Acrylic
Connector
a. No connector
b. Solder Tabs (not shown)
c. AMP Female connector

Dimensions in brackets: millimeters • Dimensional Tolerance: ± 0.016" [0.4] • Thickness Tolerance: ± 10%



Active Area: 1.5" [38.1] x 1.5" [38.1]

Nominal thickness: 0.018" [0.46 mm]

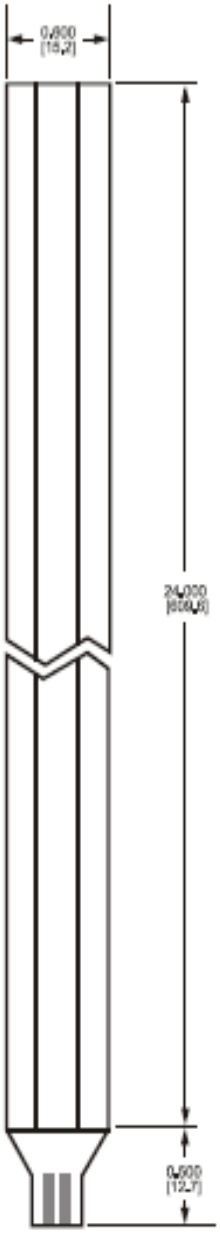
Material Build:

Semiconductive Layer
0.005" [0.13] Ultem
Spacer Adhesive
0.006" [0.15] Acrylic
Conductive Layer
0.005" [0.13] Ultem
Rear Adhesive
0.002" [0.05] Acrylic
Connectors

a. No connector
b. Solder Tabs (not shown)
c. AMP Female connector

Figure 7:
Part No. 406 (1.5" Square)

Dimensions in brackets: millimeters • Dimensional Tolerance: ± 0.016" [0.4] • Thickness Tolerance: ± 10%



Active Area: 24" [609.6] x 0.25" [6.3]

Nominal thickness: 0.135" [0.34 mm]

Material Build:

Semiconductive Layer
0.004" [0.10] PES
Spacer Adhesive
0.0035" [0.089] Acrylic
Conductive Layer
0.004" [0.10] PES
Rear Adhesive
0.002" [0.05] Acrylic
Connectors

a. No connector
b. Solder Tabs (not shown)
c. AMP Female connector

Figure 8
Part No. 408 (24" Trimmable Strip)

Dimensions in brackets: millimeters
Dimensional Tolerance: ± 0.016" [0.4]
Thickness Tolerance: ± 10%

FSR®
Force Sensing Resistors•

General FSR Characteristics

These are typical parameters. The FSR is a custom device and can be made for use outside these characteristics. Consult Sales Engineering with your specific requirements.

Simple FSR Devices and Arrays

PARAMETER	VALUE	NOTES
Size Range	Max = 20" x 24" (51 x 61 cm) Min = 0.2" x 0.2" (0.5 x 0.5 cm)	Any shape
Device thickness	0.008" to 0.050" (0.20 to 1.25 mm)	Dependent on materials
Force Sensitivity Range	< 100 g to > 10 kg	Dependent on mechanics
Pressure Sensitivity Range	< 1.5 psi to > 150 psi (< 0.1 kg/cm^2 to > 10 kg/cm^2)	Dependent on mechanics
Part-to-Part Force Repeatability	± 15% to ± 25% of established nominal resistance	With a repeatable actuation system
Single Part Force Repeatability	± 2% to ± 5% of established nominal resistance	With a repeatable actuation system
Force Resolution	Better than 0.5% full scale	
Break Force (Turn-on Force)	20 g to 100 g (0.7 oz to 3.5 oz)	Dependent on mechanics and FSR build
Stand-Off Resistance	> 1MΩ	Unloaded, unbent
Switch Characteristic	Essentially zero travel	
Device Rise Time	1-2 msec (mechanical)	
Lifetime	> 10 million actuations	
Temperature Range	-30°C to +70°C	Dependent on materials
Maximum Current	I mA/cm^2 of applied force	
Sensitivity to Noise/Vibration	Not significantly affected	
EMI / ESD	Passive device	
Lead Attachment	Standard flex circuit techniques	

For Linear pots

PARAMETER	VALUE	NOTES
Positional Resolution	0.003" to 0.02" (0.075 to 0.5 mm)	Dependent on actuator size
Positional Accuracy	Better than ± 1% of full length	

FSR terminology is defined on pages 14 and 15 of this guide.

FSR®
Force Sensing Resistors™

Glossary of Terms

Active Area	The area of an FSR device that responds to normal force with a decrease in resistance.
Actuator	The object which contacts the sensor surface and applies force to FSRs.
Applied Force	The force applied by the actuator on the active area of the sensor.
Array	Any grouping or matrix of FSR sensors which can be individually actuated and measured.
Break Force	The minimum force required, with a specific actuator size, to cause the onset of the FSR response.
Cross-talk	Measurement noise or inaccuracies of a sensor as a result of the actuation of another sensor on the same substrate. See also false triggering.
Driff	The change in resistance with time under a constant (static) load. Also called resistance drift.
Durometer	The measure of the hardness of rubber.
EMI	Electromagnetic interference.
ESD	Electrostatic discharge.
False triggering	The unwanted actuation of a FSR device from unexpected stimuli; e.g., bending or cross-talk.
Fixed Resistor	The printed resistor on linear potentiometers that is used to measure position.
Footprint	Surface area and force distribution of the actuator in contact with the sensor surface.
Force Resolution	The smallest measurable difference in force.
FSR™	Force Sensing Resistors®. A polymer thick film device with exhibits a decrease in resistance with an increase in force applied normal to the device surface.
Graphic Overlay	A printed substrate that covers the FSR. Usually used for esthetics and protection.
Housed Female	A stitched on AMP connector with a receptacle (female) ending. A black plastic housing protects the contacts. Suitable for removable ribbon cable connector and header pin attachment.
Hysteresis	In a dynamic measurement, the difference between instantaneous force measurements at a given force for an increasing load versus a decreasing load.
Interdigitating Electrodes	The conductor grid. An interweaving pattern of linearly offset conductor traces used to achieve electrical contact. This grid is shunted by the semiconductor layer to give the FSR response.
Lead Out or Busing System	The method of electrically accessing each individual sensor.
Lexan®	Polycarbonate. A substrate used for graphic overlays and labels. Available in a variety of surface textures.

Melinex® A brand of polyester(PET). A substrate with lower temperature resistance than Ulterm® or PES, but with excellent flexibility and low cost. Similar to Mylar™.

Part or Device The FSR. Consists of the FSR semiconductive material, conductor, adhesives, graphics or overlays, and connectors.

PES Polyethersulfone. A transparent substrate with excellent temperature resistance, moderate chemical resistance, and good flexibility.

Pin Out The descriptions of a FSR's electrical access at the connector pad (tail).

Repeatability The ability to repeat, within a tolerance, a previous response characteristic.

Response Characteristic The relationship of force or pressure vs. resistance.

Saturation Pressure The pressure level beyond with the FSR response characteristic deviates from its inverse power law characteristic. Past the saturation pressure, increases in force yield little or no decrease in resistance.

Sensor Each area of the FSR device that is independently force sensitive (as in an array).

Solder-tabs Stitched on AMP connectors with tab endings. Suitable for direct PC board connection or for soldering to wires.

Space and Trace The widths of the gaps and fingers of the conductive grid; also called pitch.

Spacer Adhesive The adhesive used to laminate FSR devices tighter. Dictates stand-off.

Stand-off The gap or distance between the opposed polymer film layers when the sensor in unloaded and unbent.

Stand-off Resistance The FSR resistance when the device is unloaded and unbent.

Substrate Any base material on which the FSR semi-conductive or metallic polymers are printed. (For example, polyetherimide, polyethersulfone and polyester films).

Tail The region where the lead out or busing system terminates. Generally, the tail ends in a connector.

Ulterm® Polyetherimide (PEI). A yellow, semi-transparent substrate with excellent temperature and chemical resistance and limited flexibility.

FSR®
Force Sensing Resistors®

Suggested Electrical Interfaces
Basic FSRs

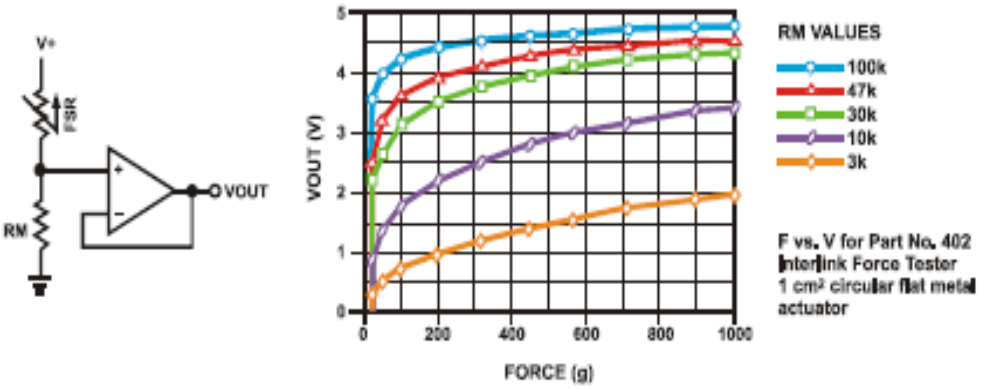


Figure 9
FSR Voltage Divider

FSR Voltage Divider

For a simple force-to-voltage conversion, the FSR device is tied to a measuring resistor in a voltage divider configuration. The output is described by the equation:

$$VOUT = (V+) / [1 + RFSR/RM].$$

In the shown configuration, the output voltage increases with increasing force. If RFSR and RM are swapped, the output swing will decrease with increasing force. These two output forms are mirror images about the line VOUT = (V+) / 2.

The measuring resistor, RM, is chosen to maximize the desired force sensitivity range and to limit current. The current through the FSR should be limited to less than 1 mA/square cm of applied force. Suggested op-amps for single sided supply designs are LM358 and LM324. FET input devices such as LF355 and TL082 are also good. The low bias currents of these op-amps reduce the error due to the source impedance of the voltage divider.

A family of FORCE vs. VOUT curves is shown on the graph above for a standard FSR in a voltage divider configuration with various RM resistors. A (V+) of +5V was used for these examples.