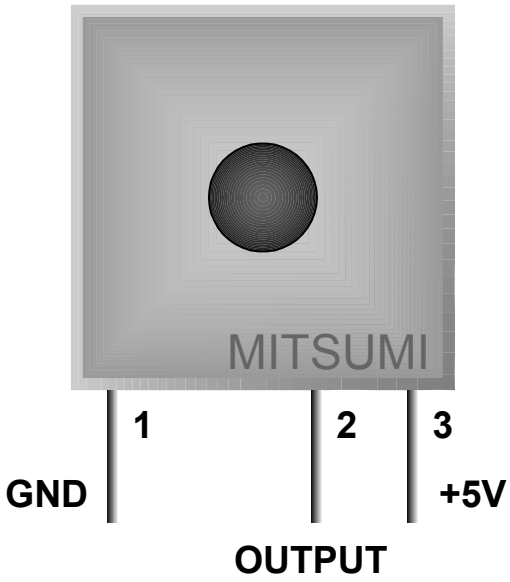
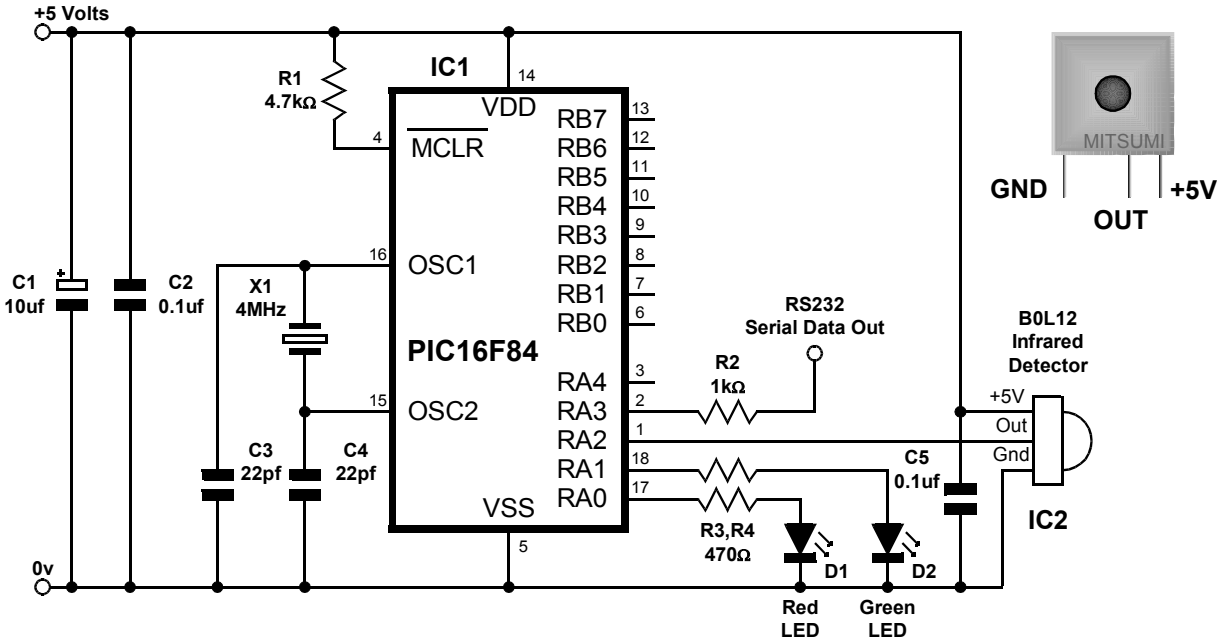
B0L12 Infrared Sensor.



Centre Frequency Approx 38KHz.

Application.

Sony Remote control receiver, Using the Crownhill PICBASIC PLUS Compiler.



```
'Program SONY_RX.BAS

'This program reads signals from a Sony IR remote control,
'and displays them on a Serial LCD, connected to PORTA.3 at inverted 9600
Baud 8-N-1
'The infrared receiver module used is the MITSUMI B0L12.

'The infrared receiver module for this experiment should,
'be a type that is set for a 38 kHz carrier frequency.
'If another type is used. some reduction in range may be noticed.
'The output pin of the IR module connects to PORTA.2

'The remote control used, may be either a Sony manufactured unit,
'or one of the universal remotes that can be configured for Sony equipment.
'This is important since we are dealing with a specific signal protocol.

'With Sony's SIRCS specification, a start pulse is initially sent to
'indicate the beginning of a frame of data. This pulse is approx 2.5 msec in
length.
'Following this, are 7-bits of data, which represent the instruction being
sent.
'Then an additional 5-bit command byte, which signifies the target device
(TV, VCR, etc.).
'Data bits are sent with the least significant bit first.

      DEVICE 16F84
      REMARKS ON
' ** Setup the Crystal Frequency, in Mhz **
      DECLARE    XTAL  4

' ** Setup the RSOUT Parameters **
      DECLARE    RSOUT_PIN PortA.3      ' Rsout PortA.1
      DECLARE    SERIAL_BAUD 9600       ' Rsout Baud Rate ***
      DECLARE    RSOUT_MODE Inverted    ' Set Serial Mode to Inverted
      DECLARE    RSOUT_PACE 50          ' Delay between characters sent

' ** Declare the Variables **
      DIM Green_Led    AS PORTA.1 ' Led flashes with a valid IR signal
      DIM IR_Sense     AS PORTA.2 ' IR sensor is attached to this pin
      DIM ST           AS WORD    ' Header length, signal
      DIM IR_Word      AS ST      ' Double up the variable, to save ram
      DIM ID           AS BYTE    ' The sony bit length 600us = 0, 1200us = 1
      DIM IR_Data      AS BYTE    ' The data byte returned
      DIM IR_Dev       AS BYTE    ' The command byte returned
      DIM Sony_LP      AS BYTE    ' Temporary variable used for a loop
      DIM IR_Valid     AS BIT     ' Flag to indicate valid signal received

      RSOUT CLS                   ' Clear the Serial LCD
      RSOUT AT 1,1,"Sony Data=   "
      RSOUT AT 2,1,"Sony Command="

Main:
      LOW Green_Led                    ' Turn off the green LED
      GOSUB Sony_In                    ' Receive the remote control signal
      IF IR_Valid = 1 THEN
           HIGH Green_Led              ' Turn on the green LED
' Display the data Byte (7-bit code), and the command byte (5-bit code)
           RSOUT AT 1,13,@IR_Data,"  ",AT 2,13,@IR_Dev," "
```

```
ENDIF
  GOTO Main                          ' Loop Forever.
```

```
' The subroutine "SONY_IN", receives the signal from a Sony remote control,
' and returns with the 7-bit data byte in the variable "IR_DATA",
' and the 5-bit command byte in the variable "IR_Dev".

Sony_In:
      TRISA.4 = 1                              ' Set the sensor pin to input
      IR_Valid = 1                             ' Initialize the valid data flag
' Are we already in the middle of a pulse? If so, exit
      IF IR_Sense = 0 THEN IR_Valid = 0 : RETURN
     IR_Word = 0
      IR_Data = 0
      IR_Dev = 0     ' Clear the variables used within the subroutine
      ST = PULSIN IR_Sense,Low                 ' Measure the header length
' Verify a good start bit, should be approx 240-260, using a 4MHz Crystal
' If not valid then return with IR_VALID = 0
      IF St < 200 THEN IR_Valid = 0 : RETURN
      IF St > 270 THEN IR_Valid = 0 : RETURN
' Receive the 12 data bits (LSB first), and convert them into a 12-bit word,
' A high (1) should be approx 120, actual timing is 1200 us
' A low (0) should be approx 60 , actual timing is 600 us
' We split the difference and say that < 100 is a low, >= 100 is a high
' These values are for use with a 4mhz crystal
      FOR Sony_Lp = 0 TO 11                ' Do 12-bits
            ID = PULSIN IR_Sense,LOW       ' Receive the IR bit pulse
            If ID >= 100 THEN
                  SET IR_Word.0     ' If greater than 100 then received a 1
            ELSE
                  CLEAR IR_Word.0  ' If less than 100 we have received a 0
            ENDIF
            IR_Word = IR_Word << 1
      NEXT                                   ' Close the loop

' Split the 7-bit data byte, and the 5-bit command byte
      IR_Data = IR_Word & %01111111          ' Mask the first 7 "DATA" bits
' Move down and mask the last 5 "COMMAND" bits
      IR_Dev = %00011111 & (IR_Word >> 7)
      RETURN                                 ' Exit the subroutine
```