Panasonic GridEYE Breakout Board and GUI

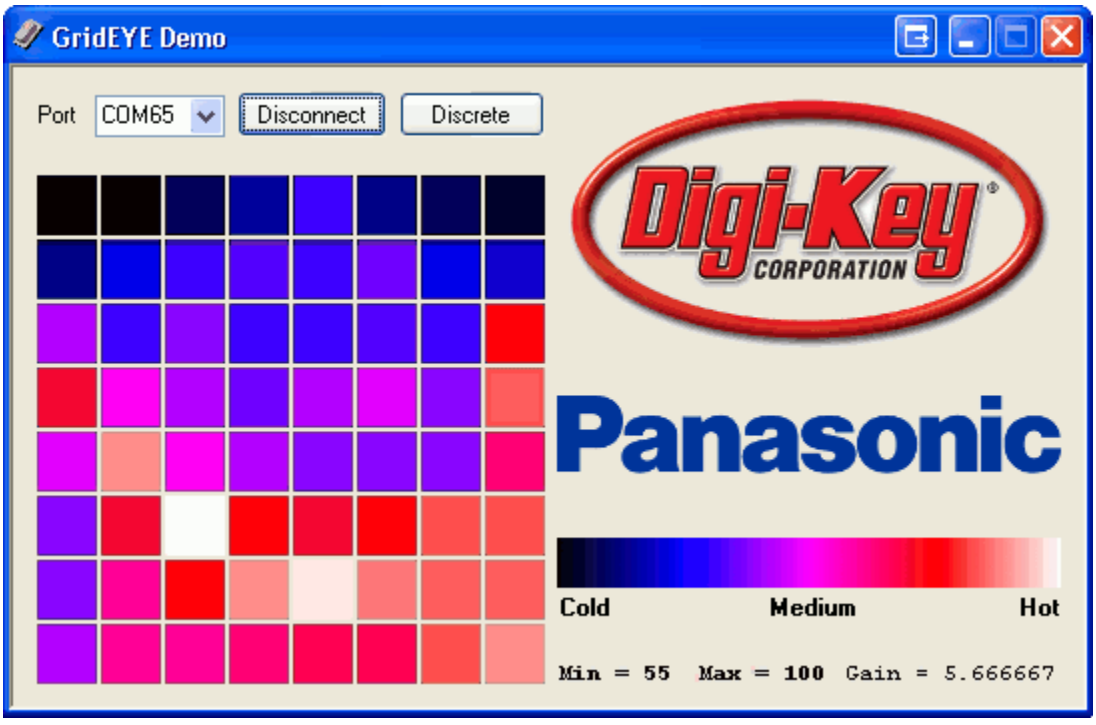
Update: The evaluation/development kit is now available from Digi-Key. These kits are fully assembled and programmed, and only require a mini USB cable to operate with the GUI below.

Click here to purchase the assembled and programmed board shown below.

Click here for a mini USB cable.

Click here to download the GUI.

Click here to download the FTDI USB Virtual COM Port (VCP) driver.



Panasonic GridEYE Sensor (3.3V high gain version used)

Basic Datasheet Link

Full Datasheet Link **<-- New!**

Typical Applications

• High performance home appliance (Microwave oven and air conditioner)
• Energy savings in office (Air-conditioning and lighting controls)

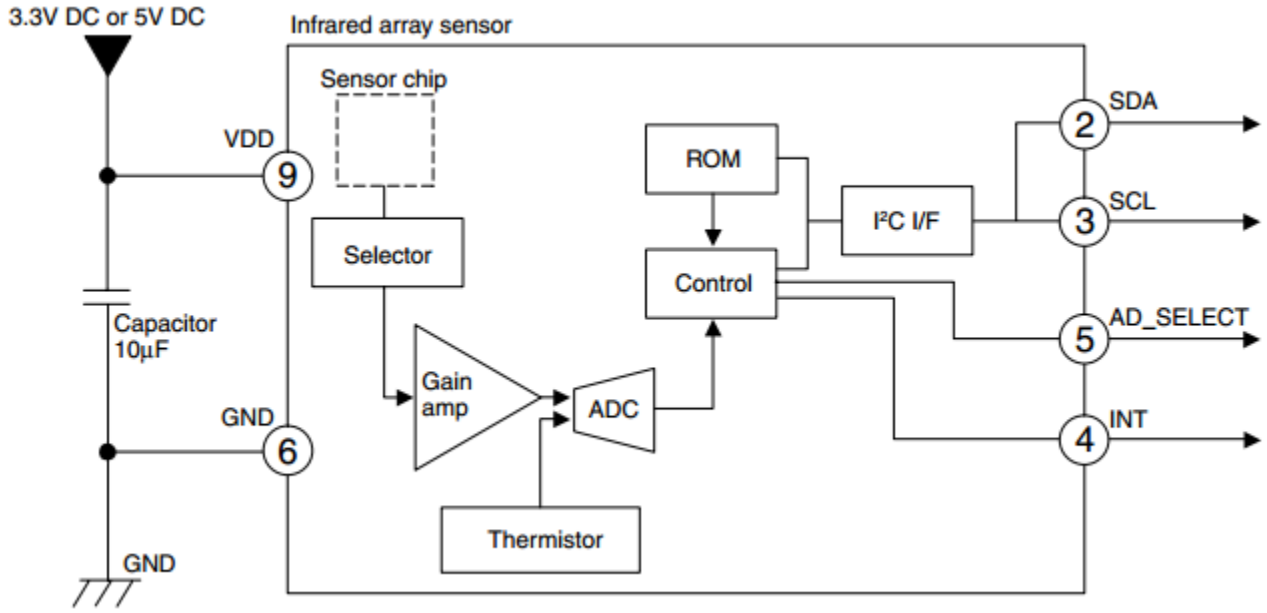
• Digital signage
• Automatic door and elevator

Functional Description

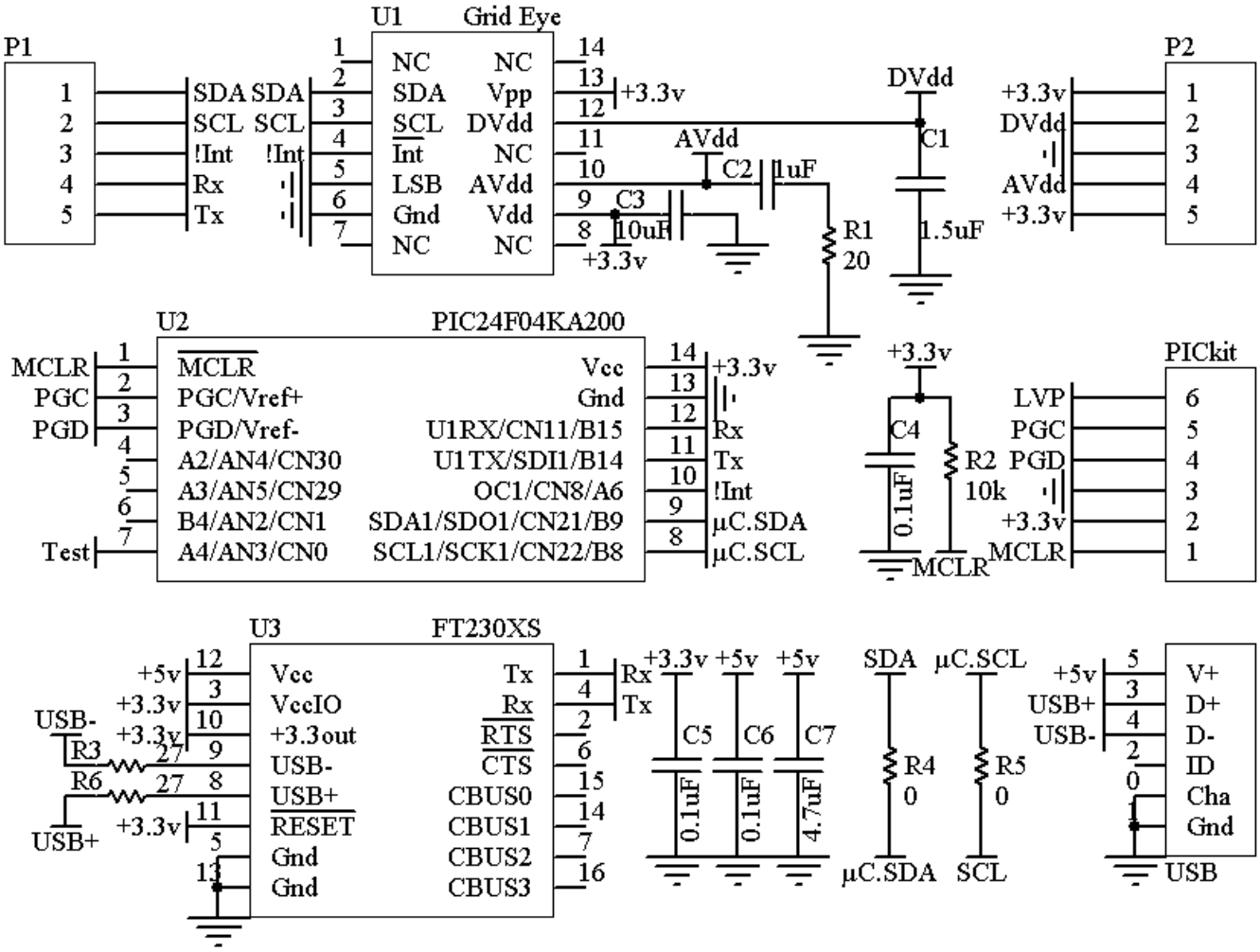
The Grid-EYE is a high precision infrared array sensor based on advanced MEMS technology. It is capable of measuring temperatures across a grid of 8x8 (64 pixels) up to ten times per second over the following temperature ranges:

Temperature range of measured object:
High gain: +32°F to +176°F (0°C to 80°C)
Low gain: –4°F to +212°F (–20°C to 100°C)

It is available in 3.3V and 5V versions, and high and low gain at each voltage. The sensor is packaged in a 8mm x 11.6mm x 4.3mm tall SMD reflow mountable can package.



Breakout Board Schematics



Breakout PCB

The PCB measures 0.8" x 0.9". It was fabricated by OSH Park.

Gerber files are available upon request.

Communication

The I²C communication routine between the Grid-EYE and PIC24 operates inside a 100mS timer interrupt:

```
// Thermistor reading
I2C Start
I2C Send Address + Write
I2C Send Byte 0x0E
I2C Stop
Delay 15 S
I2C Start
I2C Send Address + Read
I2C Read Thermistor Byte 1
I2C Send Ack
I2C Read Thermistor Byte 2
I2C Send NoAck
```

```
I2C Stop
// End of Thermistor reading

Delay 800 S

I2C Start
I2C Send Address + Write
I2C Send Byte 0x80 // Set up to read temperature Array
I2C Stop
Delay 15 S

Loop 4 times:
{
  I2C Start
  I2C Send Address + Read

  Loop 31 times:
  {
    I2C Read Pixel Byte
    I2C Send Ack
    Delay 300 uS
  }

  I2C Read Pixel Byte
  I2C Send NoAck
  I2C Stop

  Delay 2 mS
}
```

The serial packet from the PIC24 to GUI, via the UART, through the FTDI bridge and the COM Port, is as follows:

```
'*' \
'*'  >-- Packet start designator
'*' /
Thermistor High Byte
Thermistor Low Byte
64 High Byte, Low Byte pairs
```

Software

Embedded

The GridEYE sensor is connected via I2C to a PIC24F04KA200 microcontroller, which acts as the host. The PIC24 reads the thermistor on the GridEYE, followed by each of the 64 pixels, and passes the data out the UART interface, through the FTDI IC, to the Windows GUI.

```
#include <p24F04KA200.h>

#define COMM_QUEUE_SIZE 150
```

```
#define UART1_TX(x) ( U1TXREG = x )
#define ADDRESS 104

_FOSCSEL( FNOSC_FRCPLL & IESO_OFF )
_FOSC( POSCMOD_NONE & OSCIOFNC_ON & POSCFREQ_HS & FCKSM_CSDCMD )
_FWDT( WINDIS_OFF & FWDTEN_OFF )
_FPOR( BOREN_BOR0 & PWRTEN_OFF & BORV_18V & MCLRE_ON )

void I2C_start()
{
    _SEN = 1;
    Nop();
    while(_SEN);
}

void I2C_stop()
{
    _PEN = 1;
    Nop();
    while(_PEN);
}

void I2C_restart()
{
    _RSEN = 1;
    Nop();
    while(_RSEN);
}

void I2C_send_ack()
{
    _ACKDT = 0;
    _ACKEN = 1;
    Nop();
    while (_ACKEN);
}

void I2C_send_nack()
{
    _ACKDT = 1;
    _ACKEN = 1;
    Nop();
    while (_ACKEN);
}

char I2C_send_address(unsigned char addr, char read)
{
    I2C1TRN = ((addr << 1) | (read != 0));
    Nop();
    while (_TRSTAT || _TBF);
    return _ACKSTAT;
}
```

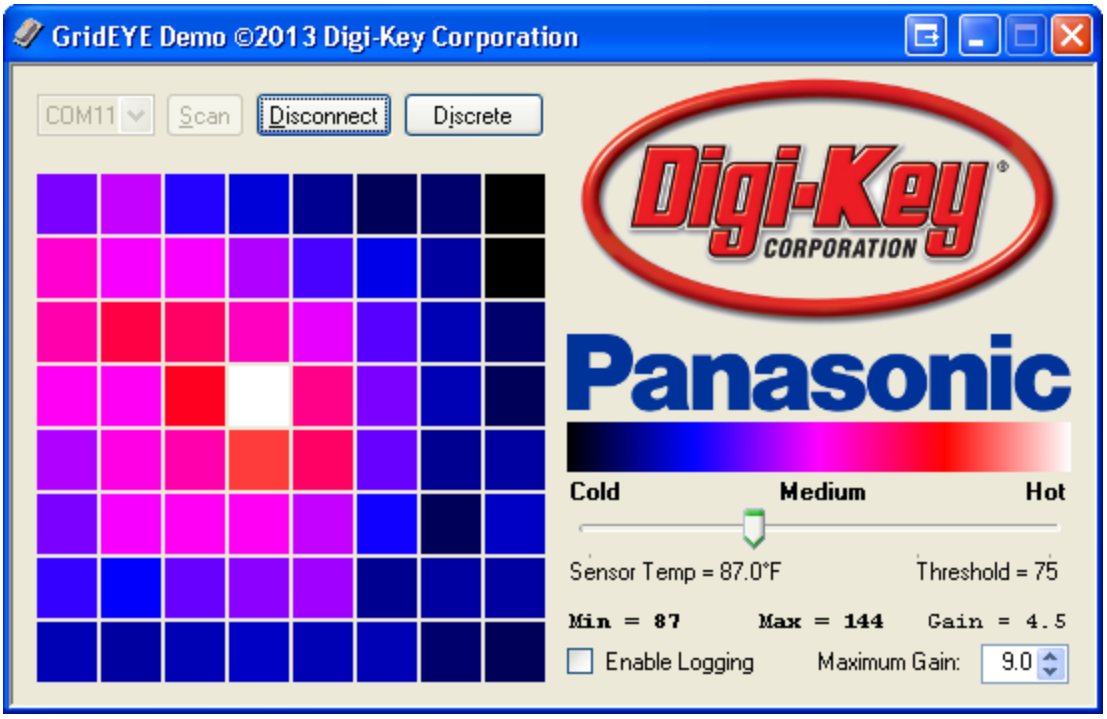
```
unsigned char I2C_send_byte(unsigned char b)
{
    I2C1TRN = b;
    Nop();
    while (_TRSTAT || _TBF);
    return _ACKSTAT;
}

unsigned char I2C_read_byte()
{
    unsigned char b;
    _RCEN = 1;
    Nop();
    while (_RCEN);
    b = I2C1RCV;
    return b;
}

unsigned char * I2C_read_bytes(unsigned char * bytes, int length)
{
    unsigned int i;
    for(i = 0; i < length; ++i)
    {
        bytes[i] = I2C_read_byte();
        if (i == length - 1)
        {
            I2C_send_nack();
        } else
        {
            I2C_send_ack();
        }
    }
    return bytes;
}

void delay_us(unsigned int delay)
{
    if( delay == 0 ) return;
    delay = delay << 1;
    while(--delay);
}

void delay_ms(unsigned int delay)
{
    while(delay--) delay_us(1000);
}

int main(  )
{
    // Hardware setup
    _RCDIV = 0;            // CLKDIV
    _CN8PUE = 1;        // Interrupt input pull-up enable
    _CN21PUE = 1;        // SDA input pull-up enable
```

```
    _CN22PUE = 1;        // SCL input pull-up enable
    _PCFG3 = 1;            // ADC pin to digital mode
    // I2C1 to GridEye
    I2C1BRG = 100;    // I2C1 Baud = 100 kHz @ 30 MIPS
    _I2CEN = 1;            // Enable I2C Module
    // UART1 to USB
    U1BRG = 8;            // UART1 BAUD = 115,200 bps when FCY = 30MHz
    _UARTEN = 1;        // Enable UART Module
    _UTXEN = 1;            // Enables UART TX hardware
    _U1RXIF = 0;        // Clear interrupt flag
    _U1RXIP = 4;        // Interrupt Priority
    _U1RXIE = 1;        // Enable interrupt

    // Timer 1 - Send data packet upon overflow
    T1CONbits.TCKPS = 3;    // Prescalar
    PR1 = 6240;                        // Target value
    _T1IF = 0;                        // Clear interrupt flag
    _T1IP = 5;                        // Interrupt priority
    _T1IE = 1;                        // Enable interrupt

    while(1)
    {
        Idle(); // Magic happens in timer and UART interrupts below.
    }
}

void __attribute__((__interrupt__, no_auto_psv)) _U1RXInterrupt(void)
{
    char data = 0;
    _U1RXIF = 0;                                                    //
Clear interrupt flag
    data = U1RXREG;                                                // Read
RX register
    if( data == '*') T1CONbits.TON = 1;        // Turns timer on
    if( data == '~') T1CONbits.TON = 0;        // Turns timer off
}

void __attribute__((__interrupt__, no_auto_psv)) _T1Interrupt(void)
{
    _T1IF = 0;
    int i,j;
    unsigned char data, checksum;
    UART1_TX('*');
    UART1_TX('*');
    UART1_TX('*');
    checksum = 0;
    // Thermistor reading
    I2C_start();
    I2C_send_address( ADDRESS, 0 );
    I2C_send_byte(14);
    I2C_stop();
    delay_us(15);
    I2C_start();
```

```
    I2C_send_address( ADDRESS, 1 );
    data = I2C_read_byte(); // Thermistor Byte 1
    I2C_send_ack();
    UART1_TX(data);
    checksum += data;
    data = I2C_read_byte(); // Thermistor Byte 1
    I2C_send_nack();
    UART1_TX(data);
    checksum += data;
    I2C_stop();
    // End of Thermistor reading
    delay_us(800);
    I2C_start();
    I2C_send_address( ADDRESS, 0 );
    I2C_send_byte(128); // Set up to read temperature Array
    I2C_stop();
    delay_us(15);
    for( i=0; i<4; i++ )
    {
        I2C_start();
        I2C_send_address( ADDRESS, 1 );
        for( j=0; j<31; j++ )
        {
            data = I2C_read_byte();
            I2C_send_ack();
            UART1_TX(data);
            checksum += data;
            delay_us(300); // 1100
        }
        data = I2C_read_byte();
        I2C_send_nack();
        I2C_stop();
        UART1_TX(data);
        checksum += data;
        delay_ms(2);
    }
    UART1_TX(checksum);
}
```

Windows GUI

The GUI was developed in Visual Basic 2010 Express (free from Microsoft).



The GUI implements the following case-insensitive keyboard commands:

s: Scan serial ports when disconnected.

c: Connect to selected serial port when disconnected.

d: Disconnect from the serial port when connected.

i: Toggle between discrete and interpolated images.

The GUI also allows logging of all data received from the development board. This data is space delimited and each packet is appended with the calculated checksum and a time-stamp. The data looks like this:

Sample Logged Data

```
42 42 42 214 1 96 0 102 0 104 0 101 0 107 0 107 0 109 0 107 0 96 0 101 0
 96 0 101 0 102 0 112 0 116 0 113 0 95 0 100 0 95 0 102 0 105 0 119 0
120 0 123 0 93 0 98 0 98 0 100 0 108 0 124 0 132 0 129 0 94 0 97 0 95 0
106 0 114 0 124 0 150 0 143 0 100 0 96 0 101 0 102 0 109 0 120 0 128 0
133 0 94 0 98 0 100 0 100 0 107 0 105 0 120 0 123 0 91 0 99 0 96 0 96 0
104 0 107 0 109 0 113 0 188 188 Timestamp: 12:25:30.687
```

Additional Information

Panasonic Grid Eye Memory Registers

Panasonic Grid Eye Library for Atmel Software Framework