

Sending data from the cRIO to an Arduino

Sending data from the cRIO to an Arduino

Sometimes it is useful to use a coprocessor to handle operations on some sensors, lights, etc. A popular processor is the Arduino. This article shows sample code to send some data between the cRIO and an Arduino. Although it only sends data in one direction (from the cRIO to the Arduino), it serves as an example of how to do it.

This program sends one of two values (either 72 or 76) from the cRIO to either turn the LED (pin 13 on the Arduino) either on or off. The value is arbitrary and was just part of a larger sample program.

The cRIO program

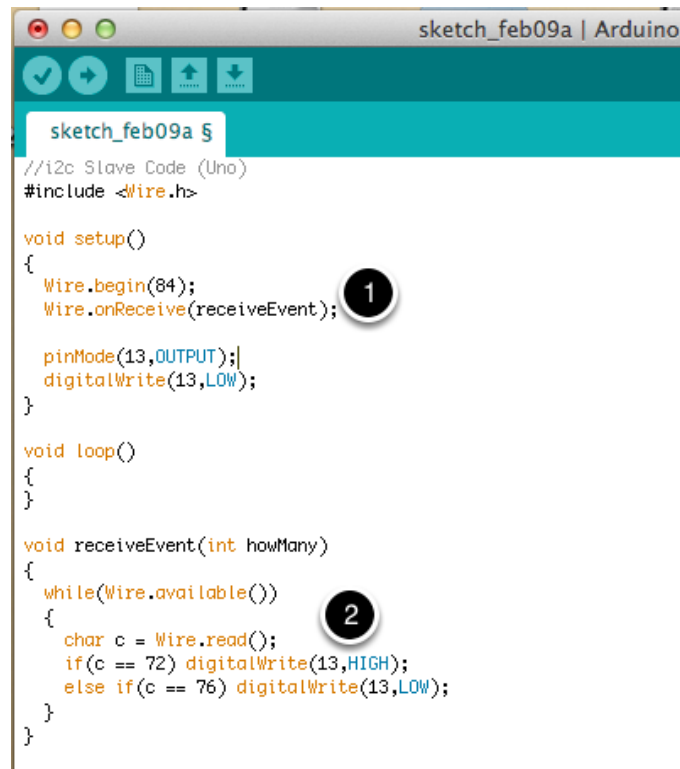
```
1 package edu.wpi.first.wpilibj.templates;
2
3 import edu.wpi.first.wpilibj.DigitalModule;
4 import edu.wpi.first.wpilibj.I2C;
5 import edu.wpi.first.wpilibj.SimpleRobot;
6 import edu.wpi.first.wpilibj.Timer;
7
8 public class RobotTemplate extends SimpleRobot {
9
10     I2C i2c;
11     byte[] toSend = new byte[1];
12
13     public void robotInit() {
14         DigitalModule module = DigitalModule.getInstance(2);
15         i2c = module.getI2C(168);
16     }
17
18     public void autonomous() {
19     }
20
21     public void operatorControl() {
22         boolean on = false;
23         System.out.println("Starting OperatorControl");
24         while (isOperatorControl()) {
25             if (on)
26                 toSend[0] = 76;
27             else
28                 toSend[0] = 72;
29             on = ! on;
30             i2c.transaction(toSend, 1, null, 0);
31             Timer.delay(.0005);
32         }
33     }
34
35     public void test() {
36     }
37 }
38
39 }
```

The I2C protocol has a master and slave processors or devices. The master controls the bus and either sends data to a slave or requests data from a slave processor. Slaves cannot initiate transactions on their own. Each slave processor (or device) has a unique address that the master processor uses to select it. In this example, the Arduino slave processor recognizes address 84. The steps are:

Sending data from the cRIO to an Arduino

1. Initialize the I2C connection on address 84. Because of differences between the implementation of the library for the cRIO and Arduino (the lower bit of the address selects either read or write) the cRIO uses address 168. 168 is the address 84 shifted by 1 bit (the read/write bit).
2. Use a byte array to fill with the data to send. In this case it's a single byte, either 76 or 72 that sets the light on or off on the Arduino.
3. Send the data to the Arduino without receiving any data. The parameters "toSend" and "1" specify a single byte from the "toSend" array. The second set of parameters (null and 0) would be a byte array and length if the master was expecting the slave to respond with some data.

The Arduino program



```
sketch_feb09a | Arduino IDE
sketch_feb09a $
//I2c Slave Code (Uno)
#include <Wire.h>

void setup()
{
  Wire.begin(84);
  Wire.onReceive(receiveEvent); 1
  pinMode(13,OUTPUT);
  digitalWrite(13,LOW);
}

void loop()
{
}

void receiveEvent(int howMany)
{
  while(Wire.available()) 2
  {
    char c = Wire.read();
    if(c == 72) digitalWrite(13,HIGH);
    else if(c == 76) digitalWrite(13,LOW);
  }
}
```

The Arduino program:

1. Sets up an interrupt handler to receive inbound requests as address 84
2. Reads the data turning the light on or off depending on the value.

