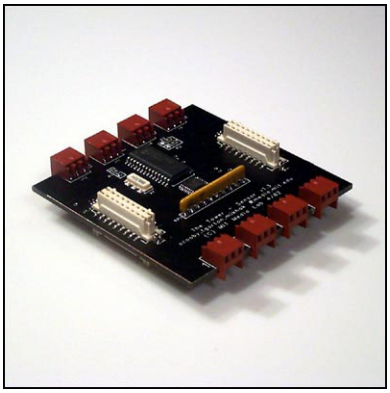
Sensor Layer

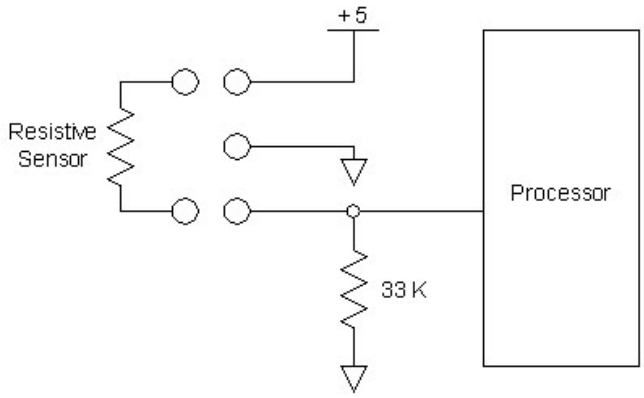


Description

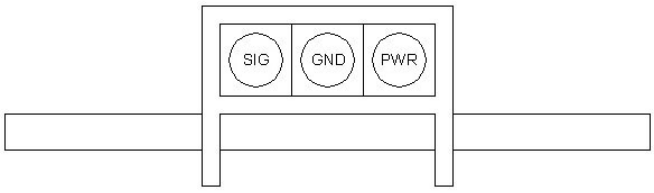
The Sensor layer has ports for eight resistive sensor inputs. By the use of a voltage divider, measurement values are converted by a 12-bit A/D converter. It is also possible to directly access power, ground, and the analog voltage-sensing lines directly from the ports as different sensing needs require.

Hardware Detail

The sensor ports are designed to allow for either resistive, or direct voltage-level readings. A schematic of the 3-terminal port is shown below:



When looking at the port from the board edge, the pin configuration is as follows:



To make a resistive measurement, the sensor is connected between the outer two pins, as shown in the diagram above. In conjunction with an on-board pull-down resistor, this creates a voltage divider, which is then read by a 12-bit A/D conversion chip, also located on the layer.

For direct voltage measurements, a ground-referenced voltage can be input into the SIG socket on the connector, the value of which will be returned normalized by the value of the power supply rail, a nominal 5 volts when using regulated wall power, however a bit higher when batteries are being used. It is important to note that this method of voltage measurement is not differential, and is therefore limited to the range between power and ground, and may not be ideal for some applications. When high precision voltage measurements are needed, we recommend using the High-Speed Sensor Layer.

Layer Code

The include file for the sensor layer contains one function, which is used to return a 12-bit sensor value. *(This function is written for the PIC Foundation. The include files used with other foundations differ slightly, due to the presence of local variables.)*

The **sensor** function is called with one argument, the sensor port number to read, which is a value between 1 and 8. The function first sends two values to the Sensor layer, a "0" indicating that a Sensor reading is being requested (as opposed to an address command), and then the sensor number to be read, minus one, since the layer firmware is actually expecting a number between 0 and 7. After the request is sent to perform the conversion, the result is read out of the layer's transmit buffer in the form of two bytes. Those two bytes are reassembled into a single result by shifting the high byte to the left by 8 bits, and then performing a logical or with the low-byte result. Actually three bytes are read out, but the first is ignored, as it is known to be a "2", representing the number of arguments to follow.

```
to sensor :n
    i2c-start
    i2c-write-byte $0a
    i2c-write-byte 2
    i2c-write-byte 0
    i2c-write-byte (:n - 1)
    i2c-stop
    i2c-start
    i2c-write-byte $0b
    ignore i2c-read-byte 1
    seti2c-byte (lsh i2c-read-byte 1 8)
    seti2c-byte i2c-byte or i2c-read-byte 0
    i2c-stop
    output i2c-byte
end
```

Examples of Use

Using the Sensor layer is as simple as calling the "sensor" function, with a single argument representing the sensor port to be read. A simple example of how to print a sensor value to the console is shown below:

```
print sensor 1
> 1527
```

This code will first call the "sensor" function with an argument of "1". The result of that operation, a value between 0 and 4095, will then be passed to the **print** function, which then prints the value to the console. In the above sample, the value happened to be 1527.

If you want to do a conditional test of a sensor value, and depending on the value, do different things, a sample of how to do so is given here:

```
ifelse ((sensor 1) > 1000)
    [set red-led led-port]
    [set green-led led-port]
```

In this example, the value of the sensor in port 1 is read, and if its result is greater than 1000, a red LED is turned on. If the result is found to be less than 1000, a green LED is turned on.