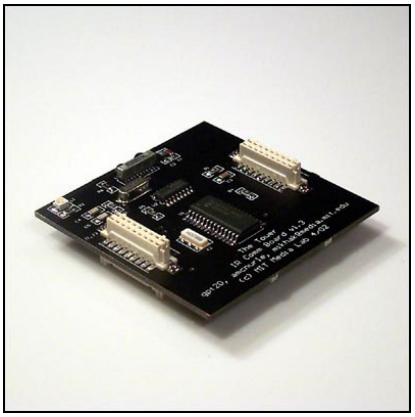
IR Layer



Description

The IR layer can be used for short-range, wireless communications. The hardware is fully IRDA compliant and thus can be used to communicate with any device speaking over this standard hardware protocol.

Hardware Detail

The IRDA emitter/detector pair is an Agilent HSDL-3610. IR data is can be sent and received at any baud rate from 2400 to 57600 bps through the UART port on the on-board PIC, and is translated to the necessary IR waveforms by an Agilent HSDL-7001. At full strength, signals can be transmitted up to about 8 feet. There are user-settable power settings, which allow anyone to reduce the strength of the signal transmission to either 2/3 or 1/3 of full power if desired, or turned off entirely, to avoid unintentional communication with other nearby compliant devices.

Layer Code

The include file for the IR layer contains five functions, one each for sending and receiving bytes, one to see if new data has been received, one for adjusting the transmission power, and a fifth for setting the baud rate. *(All of these functions are written for the PIC Foundation. The include files used with other foundations differ slightly, due to the presence of local variables.)*

The **new-ir?** function is used when you want to find out if a new data byte has been received over IR. It is almost always desired to check first before reading from the register, to avoid unintentional double-reads of the same data. The function takes no arguments, and will return either a “1” or a “0”, with a 1 signifying that new data is present, and a zero meaning otherwise. The function itself first sends a single value to the layer, a “0” indicating that it wants to perform the query. The result of “1” or “0” is then read back from the layer and output.

```
to new-ir?
    i2c-start
    i2c-write-byte $06
    i2c-write-byte 1
    i2c-write-byte 0
    i2c-stop
    i2c-start
    i2c-write-byte $07
    ignore i2c-read-byte 1
    seti2c-byte i2c-read-byte 0
    i2c-stop
    output i2c-byte
end
```

Once we know that new data is available, we can use the **get-ir** function to obtain the actual value that was received. The function takes no arguments, and sends a single value of "1" to the layer to indicate the desired operation. The result is then read back from the layer and output, again ignoring the first byte back, which is simply the number of bytes to follow, which in this case we know is one.

```
to get-ir
    i2c-start
    i2c-write-byte $06
    i2c-write-byte 1
    i2c-write-byte 1
    i2c-stop
    i2c-start
    i2c-write-byte $07
    ignore i2c-read-byte 1
    seti2c-byte i2c-read-byte 0
    i2c-stop
    output i2c-byte
end
```

The other key thing that we will probably want to do with the IR layer is to send data out. This can be done using the **put-ir** function. The function takes a single argument, the byte to be sent. Two values are sent to the layer, first a "2" indicating that data is to be sent out, and then the value itself. As soon as the layer receives the value, it will immediately be sent out over the IR channel.

```
to put-ir :n
    i2c-start
    i2c-write-byte $06
    i2c-write-byte 2
    i2c-write-byte 2
    i2c-write-byte :n
    i2c-stop
end
```

If we want to adjust the IR transmission power, we can use the **ir-setpower** function. The function takes a single argument, the desired power. The power setting can range from 0 to 3, with 0 corresponding to a disabled transmitter, and 3 setting the transmitter to full power in order to obtain maximum range. The function sends two values to the layer, a "3" to enter the power adjust mode, and then the actual power setting itself.

```
to ir-setpower :power
    i2c-start
    i2c-write-byte $06
    i2c-write-byte 2
    i2c-write-byte 3
    i2c-write-byte :power
    i2c-stop
end
```

Finally, the **ir-setbaud** function can be used to adjust the baud rate of the transmitter and receiver. The function takes a single argument, a value from 0 to 5 corresponding to the desired baud rate. The available baud rates are shown in the table below:

Control Value	Baud Rate
0	2400
1	4800
2	9600
3	19200
4	38400
5	57600

Two arguments are sent to the actual layer, a "4" signifying a baud rate set command, and then the desired value itself. At power-on, the layer will be operating at 38400 baud.

```
to ir-setbaud :baud
    i2c-start
    i2c-write-byte $06
    i2c-write-byte 2
    i2c-write-byte 4
    i2c-write-byte :baud
    i2c-stop
end
```

Examples of Use

The two most basic things to do with the IR layer, are sending and receiving bytes. Sending a byte is very simple:

```
put-ir 23
```

We've just sent the data byte 23 out over the IR channel. If we want do read that value with another Tower, we could do something like this:

```
print get-ir
> 23
```

This method works well, assuming that we know to read the byte on the receiver after we find out that it's been sent. But what if we don't know that? We probably want to make our receiver program wait until new data has been received, and then read it out. Remember the **new-ir?** function? We can use that to do just what we want to:

```
waituntil [new-ir?] print get-ir
> 23
```

In this case, we wait until the **new-ir?** function returns a true value of 1, meaning that new data has been received. As soon as that happens, we print out the result. We've now got a reliable way of sending and receiving data through the IR channel.

If we're operating in a room full of IR devices, and want to shorten the transmission range to ensure that we're only talking to the one we want, we can reduce the transmission power like this:

```
ir-setpower 1
```

We've just reduced the transmission power to 1/3 of it's full value. We can just as easily set it to any of the available power settings or turn it off entirely with a power argument of "0".

Finally, let's try changing the transmission baud rate to 19200 bps:

```
ir-setbaud 3
```

According to the table, an argument of "3" corresponds to 19200 bps, so we should now be able to communicate with any device operating at the same speed.