Networked Objects

Patrick Dwyer
Spring 2006
Week 5 - February 16th

XPort meet PIC

Our example code for this week calls a script on a remote server. The PIC establishes a connection to the remote server through the XPort, and then transmits an HTTP request. We can embed data in the request URL to interact with the remote script.

XPort Settings

The following are the important settings to configure on the XPort for this example:

```
*** Channel 1
Baudrate 9600, I/F Mode 4C, Flow 00
Port 10001
Remote IP Adr: --- none ---, Port 00000
Connect Mode : D4
Disconn Mode : 00
Flush   Mode : 00
```

xport_hello.pbp

The following code is also available on the website.

```
' We are using a 20Mhz oscillator
DEFINE OSC 20

' We communicate with the XPort using 9600 8N1 serial
true9600 con 84

' Our serial communication pins
tx var PORTC.6
rx var PORTC.7

' Our blinking pin for status messages
statusPin var PORTC.4
OUTPUT statusPin

' Used to read response from the XPort
inByte var byte

' Track whether or not we are connected to the remote server
connected var bit
connected = 0

' Turn on our LED so we know that the startup sequence is going
```

```
high statusPin

' Wait for the XPort to boot up
pause 5000

' Turn off the status LED while operating
LOW statusPin

' Just so we know that we're ready, we'll quickly blink the status LED
counter var byte
counter = 0
while counter < 4
    high statusPin
    pause 100
    low statusPin
    pause 100
    counter = counter + 1
wend

' In our main loop we need to accomplish the following:
'           1. Connect to the remote server
'           2. Send data to the remote server
'           3. Pause so we don't continually send data
' We use the xport_connect method to establish our connection,
' and the http_request method to send our data to a remote script
main:


    if connected = 1 then
      ' If we're already connected then we can send our message
        gosub http_request

        ' back off the server for a few seconds
        pause 4000
    else

        ' try and connect
        gosub xport_connect

    endif

goto main

xport_connect:

    ' Send our connection string to the XPort. This string contains
```

```
    ' the remote IP and the remote port. In this case we're connecting
    ' to digilutionary.com on port 80 (HTTP)
    serout2 tx, true9600, ["C82.165.251.49/80", 10]

    ' now we're connected
    connected = 1
return

http_request:

    ' light LED to indicate HTTP GET request in progress:
    high statusPin

      ' To send data to the server we need to create a valid HTTP request
      ' header. All of the data we transmit here will go to the remote
      ' server.
      ' The first line tells the server that we want to get a certain
      ' page, and includes the url parameters that we want to pass to
      ' our script.
      ' The second line tells the remote server what format the header
      ' we're sending is in.
      ' The third line tells the server that the domain we want to be
      ' talking too is digilutionary.com
      ' The next line identifies what type of device is requesting data.
      ' Finally we send two newline characters, which tells the server that
      ' our request header is finished, at which point it runs our remote
      ' script
    SEROUT2 TX, true9600, ["GET /classes/netobj/write.py?user=pnd201&msg=Hi"]
    serout2 tx, true9600, [" HTTP/1.1", 10]
    SEROUT2 tx, true9600, ["HOST: digilutionary.com", 10]
    SEROUT2 tx, true9600, ["User-Agent: PIC/XPort"]
    serout2 tx, true9600, [10, 10]


    ' wait for bytes from server:
    ' Our server sends a 0 to finish
    while inByte <> 0
        serin2 rx, true9600, [inByte]
    wend

    ' now we're disconnected:
    connected = 0

    ' turn off LED, since GET request is complete:
    low statusPin
return
```