Configuring xbee modules through serial communication

Introduction

Configuring xbee modules allows you to see and change configured settings of the xbee. For example, you can see the serial address of a module (a unique way address given to each xbee for targeting when transmitting/receiving data) and change which module a module will send data to.

An xbee can be configured through serial communication from a computer to the xbee module.

Table Of Contents

Xbee configuration quick reference

A quick reference to the pin configurations and commands needed to configure xbee for communication between modules.

In order for the xbee modules to communicate they should be set to the same channel, same PAN ID, and same API mode. Depending on the API mode they may need to set corresponding MY and DH/DL values. The commands used to view or change these values can be seen below.

In API mode 0 (zero), the DH value should be 0 (zero) and the DL value should be the MY value of the xbee module you want to transmit data to.

In API mode 1 (one), along with your message, you may specify which xbee module to send the message to so it is not necessary to configure DL and MY values.

Here is how an example setup that allows xbee modules A and B to communicate.

Value	Xbee module A	Xbee module B
CH	C	C
ID	2	2
DH	0	0
DL	2	1
MY	1	2
AP	0	0

M3pi robot mbed pins

- Transmission (TX) pin - p28
- Recieve (RX) pin - p27
- Reset (rst) pin - p26

Application board mbed pins

- Transmission (TX) pin - p9
- Recieve (RX) pin - p10
- Reset (rst) pin - p30

Configuration commands

- ATCH - Channel. Set/Read the channel number used for transmitting and receiving data
- ATID - PAN ID. Set/Read the PAN (Personal Area Network) ID.
- ATDH - Destination Address High. Set/Read the upper 32 bits of the 64-bit destination address. When combined with DL, it defines the destination address used for transmission.
- ATDL - Destination Address Low. Set/Read the lower 32 bits of the 64-bit destination address. When combined with DH, DL defines the destination address used for transmission.
- ATMY - 16-bit Source Address. Set/Read the xbee module 16-bit source address.
- ATAP - API Enable. Disable/Enable API Mode.
- ATAC - Apply Changes. Explicitly apply changes to queued parameter value(s) and re-initialize module.
- ATWR - Write. Write parameter values to non-volatile memory so that parameter modifications

- persist through subsequent power-up or reset.

What you will need

If you are using a windows operating system you will need to download and install the mbed Windows serial port driver onto every mbed device. The drivers can be found at: Windows-serial-configuration. Instructions and more detailed information about serial communication can be found on the same page.

You will also need a terminal program, which lets you input data via keyboard through the USB serial port to be sent as serial data to the xbee and receive data back from the xbee, and you will need the mbed microcontroller to be plugged in via usb to the computer.

I recommend the terminal program “CoolTerm” which can be found at: http://freeware.the-meiers.org/. The program is the first one listed.

Configuring settings

After the driver is installed on your mbed device it should be able to be configured. Below a program is listed that can be used with the terminal program “CoolTerm” to configure your xbee. This program requires you to include “mbed.h”, an official library for the mbed. Creating a new example program in the online compiler will create an example program with the mbed library already included, so all you would need to do is modify the code. You could also create a new empty project and include the library by importing it from its webpage at: https://developer.mbed.org/users/mbed_official/code/mbed/.

```
// Official library for using mbed microcontroller
#include "mbed.h"

// Create objects to represent used accessories like led lights, button, xbee, a computer connected to robot through usb
Serial pc(USBTX, USBRX);
Serial xbee(p28, p27);
DigitalOut rst(p26);

int main()
{
    // xbee must be reset before use
    rst = 0;
    wait(0.5);
    rst = 1;
```

```
    wait(0.5);

    pc.printf("start serial communication\n");

    while (1) {
        // If terminal data available send through xbee
        if(pc.readable())
            xbee.putc(pc.getc());

        // If xbee gets data send to terminal
        while(xbee.readable())
            pc.printf("%c", xbee.getc());
    }
}
```