

XBee for NewBees (DOS-Tool zum Testen von XBee's im API-Modus)

Benötigt werden: FTDI-USB-Seriell-Adapter / ftd2xx.dll
ggf: C-Compiler: z.B. Dev-C++

Während die Arbeit mit den XBee's im AT-Modus fast ohne eigenes Zutun funktioniert, ist die Latte für den Einstieg in den API-Modus höher gehängt.

Um den Anfang und das Debuggen nicht allzu frustrierend werden zu lassen, habe ich mich entschlossen, zunächst ohne AVR ausschließlich am PC unter Dev-C++ zu arbeiten.

Damit habe ich die grundlegenden Routinen erstellt und getestet - und nebenbei ist ein kleines Tool entstanden, mit dem man XBee's im API-Modus konfigurieren und ihre Arbeitsweise erforschen kann.

Das Programm läuft in einem schlichten DOS-Fenster. Die Eingabe erfolgt über die DOS-Befehlszeile.

Der PC ist über einen FTDI USB-Seriell-Adapter mit einer der XBee's verbunden.

Die Software greift über die ftd2xx.dll von FTDI auf den Adapter zu (die Datei "cdm 2.06.00 whql certified\cdm 2.06.00 whql certified\i386.zip" muss ggf. von der Herstellersite ftdichip.com geladen werden).

Getestet habe ich in einem kleinen Netzwerk mit 1 Koordinator und 2 Routern (Modem XB24-ZB, Firmware 21xx bzw. 23xx), die zugehörige Dokumentation 90000976_F.pdf ist erschöpfende Pflichtlektüre.

Nach dem Laden der jeweiligen Firmware auf die XBee's (in ihrer Default-Einstellung) wurden alle weiteren Schritte aus der Befehlszeile des Tools ausgeführt.

Folgende Szenarien habe ich durchgespielt und ihre Tücken erforscht:

- beliebige Konfiguration des lokalen und der remote-Devices mittels AT-Kommandos, z.B. Setzen/Auslesen von Systemvariablen und Portpins, Auslesen von ADC-Werten ...
- ein Router wird als Enddevice in den sleep-Modus geschickt - und wieder als Router zurückgeholt,
- ein Enddevice läuft im sleep-Modus und liefert beim Aufwachen IO-Daten an ein anderen Router.

Mit den Erfahrungen aus der Arbeit mit diesem Tool sollte der Einsatz eines Microcontollers zur Steuerung des Netzes deutlich leichter fallen.

Dazu können Teile des C-Programmcodes in angepasster Form für den AVR genutzt werden.

Und genau das war das eigentliche Ziel dieser Übung.

Viel Spaß beim Funken,

Michael S.

Für die Arbeit im API-Modus ist weder das X-CTU noch ein Terminal-Programm eine echte Hilfestellung, da die Nachrichten in eine fest definierte Form (Frames) gebracht werden und mit Längenangaben und einer Prüfsumme versehen werden müssen.

Manuell ist die Zusammenstellung dieser Frames extrem umständlich.

Um aber auch im API-Modus mit wenig Aufwand ein Netzwerk konfigurieren zu können, habe ich mir ein kleines Tool zusammengehäkelt, das sozusagen als 'Frame-Builder' Daten in die notwendigen API-Frame-Formate zwingt.

Bzw. eingehende Daten in ihrer Frame-Struktur wiedergibt.

Achtung: die Oberfläche des Programms hat den Charme eines Shell-Scripts.

Die Ausgabe erfolgt in einem einfachen DOS-Fenster.

Nach der Eingabe eines Commands wird das Fenster mit den Meldungen einfach gelöscht und mit den neuen Daten neu aufgebaut.

Bei AT-Commands ist keine Guard Time wie im AT-Modus zu beachten.

Auch darf kein '+++' gesendet werden.

Das 'AT' des AT-Commands ist zu ersetzen durch '++' (für ein lokales AT-Command) oder '###' für ein Remote AT-Command !

Die Eingabeoptionen am Prompt sind :

XX + AT-Command gemäß der Xbee Command Reference Table

wobei XX anstelle von 'AT' für '++' oder '###' steht.

'++' sendet ein lokales AT-Command (Frame 0x08) an das am Modem angeschlossene Radio

'###' sendet ein remote AT-Command (Frame 0x17) an ein entferntes Radio

Beispiel: ++dh0013a200 setzt die Destination-High-Adresse auf dem lokalen device

###dh0013a200 setzt die Destination-High-Adresse auf dem remote device

Die Parameter hinter dem AT-Command werden hexadezimal ohne '0x' erwartet und vor dem Senden in binäre Form umgewandelt - solange die Zeichenfolge aus gültigen hexadezimalen Ziffern besteht.

Ausnahme ist nur ATNI. Hier werden alle folgenden Zeichen als String gesendet.

#x ein Target (Zielstation) für die nachfolgenden Frames wählen (x = Nr. des Targets)

+ nach eingehenden Daten suchen
(Das ist eine Krücke, da im Eingabemodus im Hintergrund empfangene Bytes nicht erkannt werden)

- löscht das DOS-Fenster

beenden (alternativ Ctrl-C)

Alle anderen Eingaben wird als "transparente" Daten via Frame 0x10 versandt.

Nach jeder Eingabe (außer '+') wird der Bildschirm gelöscht.

In der ersten Zeile wird das erkannte Kommando ausgegeben, in der Folgezeile das versandte Frame in hexadezimaler Darstellung.

Es folgt dann die Anzahl der zurückgelieferten Bytes sowie die Daten in hexadezimaler Darstellung.

Vom Empfänger wird normalerweise eine Quittung zurückgegeben.

Das Programm wartet ca. 6 Sekunden auf eine Antwort.

Wird eine Antwort erkannt, dann erfolgt die Anzeige des/der Frames und es wird wieder in den Eingabemodus gewechselt.

Bei manchen Kommandos (etwa ATND oder bei Broadcasts) kann es einige Zeit dauern, bis die Rückmeldungen von allen Stationen eingehen.

Während das Programm im Eingabemodus läuft, kann nicht erkannt werden, dass weitere Daten empfangen wurden. Das hat zu der folgenden Hilfskonstruktion geführt:

Wird ein '+' eingegeben, dann prüft das Programm auf zwischenzeitlich empfangene Daten.

Die empfangenen Frames werden - soweit möglich - entsprechend ihrer Struktur formatiert ausgegeben.

Zur Kontrolle werden alle versandten bzw. empfangenen Frames byteweise auf dem Bildschirm ausgegeben.

Die Target-Adressen (also die Seriennummern aller beteiligter Devices, an die transparente Daten und remote AT-Commandes verschickt werden) müssen in der Datei 'xbee_adr.cfg' vereinbart werden.

Die Default-Adresse für Coordinator und Broadcast sind bereits im Programmcode definiert, der Coordinator wird als Target#0, ein Broadcast als Target#1 adressiert.

Es ist zu beachten, dass die Array-Größe für die Adressen an die Anzahl der vorhandenen Stationen angepasst wird (wobei noch eine 2 für Coordinator und Broadcast zu addieren sind).

Für 3 Stationen muss in der global.h also als Arraygröße #define DEVICES 5 vereinbart werden.

In der 'xbee_adr.cfg' müssen natürlich diese Einträge auch vorhanden sein.

Die Bytes werden immer im Format 'FF' mit genau zwei hexadezimalen Zeichen eingetragen !

Die Einträge in der Datei müssen durch die Adressen der eingesetzten XBee's ersetzt werden !!

Die Adressen der Stationen werden beim Programmstart aus der Datei 'xbee_adr.cfg' eingelesen.

Durch die Eingabe von # + Targetnummer kann als Ziel für alle nachfolgenden Übertragungen eine beliebige Station angewählt werden.

Vorgabe beim Programmstart ist #0 == Coordinator.

Zur Kontrolle der XBee's ist es sinnvoll, sie während der Testphase jeweils mit 3 LED's zu bestücken. Die zeigen dann (gegen gnd geschaltet) an:

- Pin.6 XBee hat Daten gerade empfangen, die Led leuchtet für 4 Sekunden (->ATRP),
genaugenommen sollte die Helligkeit der Led sogar abhängig sein von der Empfangsstärke.
- Pin.13 XBee im Sleep-Modus (Led off) oder aktiv (Led on),
- Pin.15 XBee hat (irgendwann einmal) ein Netz gefunden (Led blinkend) oder nicht (Led on).

Eine blinkende Led an Pin.15 sagt nicht zwingend aus, dass Coordinator und Router assoziiert sind.

So blinkt die Led am Router, auch wenn der Coordinator noch gar nicht eingeschaltet ist.

Die Led zeigt nur an, dass der Router/Enddevice irgendwann einmal assoziiert war.

Um ggf. eine neue Assoziation zu erzwingen, kann man den Router mit 'ATNR0' reseten.

Dann baut er eine neue Verbindung zum Coordinator auf.

Viel Freude entsteht, wenn man in einem funktionierenden Netz ein 'ATNR0' an den Coordinator sendet.

Ein ATNR1 dagegen ist ohne Risiken und Nebenwirkungen.

Warum ? Einfach mal ausprobieren.

Ach ja, als Fingerübung sollte auch getestet werden:

#1 als Target ein Broadcast wählen

##CB1 als Broadcast ein ATCB1 senden (falls Daten zu fehlen scheinen, mit '+' nachhelfen !)

Nun senden alle anderen Stationen das Frame 0x95 'Node Identification Indicator'.

Da die Frame_ID > 0 war, wird zusätzlich von allen Stationen noch ein 'Remote Command Response' via Frame 0x97 verschickt.

#0 als Target den Coordinator wählen

##CB1 Nun sendet ausschließlich der Coordinator sein Frame 0x95 / 0x97.

++CB1 Nur das lokale Radio sendet ein Frame 0x95 - was man aber nur am Blinken der Assoc.-Led der anderen Stationen erkennen kann.

Allerdings wird das AT-Command mit einem Frame 0x88 bestätigt.

##D03 Pin.0 als digitalen Eingang setzen

##IS den Pin-Status anfordern - im Datenbereich des Frames ist der Status codiert (10101).

##IRF00 automatische IO-Samples im 4-Sekundenabstand anfordern, die werden via Frame 0x92 versandt.

+ das '+' holt die eingegangenen Bytes auf den Schirm

##IRO das automatische IO-Sample wieder abschalten

Viel Spaß beim Funken,

Michael S.