

Packet Radio via Soundkarte unter Linux

[Versionsgeschichte interaktiv durchsuchen](#)

[Visuell Wikitext](#)

Version vom 4. Mai 2011, 19:02 Uhr (Quelltext anzeigen)

[Oe5dxl](#) ([Diskussion](#) | [Beiträge](#))

[← Zum vorherigen Versionsunterschied](#)

Aktuelle Version vom 22. November 2019, 18:48 Uhr (Quelltext anzeigen)

[OE5HPM](#) ([Diskussion](#) | [Beiträge](#))

[\(→Der Source Code\)](#)

(15 dazwischenliegende Versionen von 3 Benutzern werden nicht angezeigt)

Zeile 1:	Zeile 1:
[[Kategorie:Packet-Radio und I-Gate]]	[[Kategorie:Packet-Radio und I-Gate]]
== Das Projekt ==	== Das Projekt ==
<div><div><div>Dieser (USB) Soundkartentreiber von OE5DXL befindet sich in der</div><div>Entwicklung und soll es ermöglichen mit 2 Kanälen mehrere Modems zugleich unter Linux zu initialisieren.
</div></div></div>	<div><div><div>Dieser (USB) Soundkartentreiber von OE5DXL soll es ermöglichen mit 2 Kanälen</div><div>(L und R der Soundkarte) mehrere Modems zugleich unter Linux zu initialisieren.
</div></div></div>
Als KISS Treiber sind bis zu 16 Modems von 1baud bis 28kbaud möglich. Der Equalizer ermöglicht einen Vollduplexbetrieb bei Verwendung eines getrennten Senders und Empfängers. 	Als KISS Treiber sind bis zu 16 Modems von 1baud bis 28kbaud möglich. Der Equalizer ermöglicht einen Vollduplexbetrieb bei Verwendung eines getrennten Senders und Empfängers.
<div><div><div>Weiterer Vorteil ist die Möglichkeit des ""Multibaud"" Digi, also mehrere Geschwindigkeiten FSK AFSK gemischt.</div></div></div>	<div><div><div>Weiterer Vorteil ist die Möglichkeit des ""Multibaud"" Digi, also mehrere Geschwindigkeiten FSK AFSK gemischt (bs pw. 1k2 2k4 4k8 9k6 auf einer QRG).</div></div></div>
In Stereo kann so theoretisch ein multibaud FSK AFSK KISS, als auch AXUDP AX.25 Modem betrieben werden.	In Stereo kann so theoretisch ein multibaud FSK AFSK KISS, als auch AXUDP AX.25 Modem betrieben werden.
	<div><div><div>[[Bild:Soundmodem-box.gif Soundmodem Schema]]</div></div></div>
==Der Source Code==	==Der Source Code==
<div><div><div>in Kürze</div></div></div>	<div><div><div>[https://github.com/oe5hpm/dxIAPRS] https://github.com/oe5hpm/dxIAPRS
</div></div></div>

		+	http://qitlab.oe5xbl.ampr.org/oe5hpm/dxIAPRS/ http://qitlab.oe5xbl.ampr.org/oe5hpm/dxIAPRS/
		+	==Der kompilierte Treiber==
		+	[[Media:soundmodem i386 linux.zip Soundmodem-bin]] - Der fertig kompilierte Soundmodem Treiber
-	==Der kompilierte Treiber==	+	[[Media:udpbox i386 linux bin.zip udpbox-bin]] - UDP Filter und RAW-Monitor Konverter und (neu) mit aprs-diql, Bake, User-Message-Receiver
-	Hier im ZIP der fertig kompilierte Soundmodem Treiber zum Download: [Media:soundmodem_linux_i386_static.zip Soundmodem-bin]]	+	
		+	[[Media:udpgate i386 linux bin.zip udpgate-bin]] - I-Gate, APRS-IS, APRS-Server mit Rangefilter, HTML-Statistik, Log
-	UDP Filter und RAW-Monitor Konverter	+	[[Media:udphub i386 linux bin.zip udphub-bin]] - axudp Hub zum HAMNET-PR-Login ohne IP Beschränkung
-	[[Media:udpbox_linux_i386_static.zip udpbox-bin]]		
		+	[[Media:udpflex i386 linux bin.zip udpflex-bin]] - Interface com-port (/dev/ttySxx) mit KISS oder RMNC bidirektional auf axudp
	==Starten bzw. Aufrufen des Treibers==		==Starten bzw. Aufrufen des Treibers==
Zeile 24:	mit also:	Zeile 29:	mit also:

```
aoss ./afskmodem -f 32000 -M 0 -c 0 -b
1200 -M 1 -c 0 -b 9600 -a -g
```

```
aoss ./afskmodem -f 32000 -M 0 -c 0 -b
1200 -M 1 -c 0 -b 9600 -a -g
```

APRS mit Xastir KISS-Interface, PTT auf
ttyS0:

APRS mit Xastir KISS-Interface, PTT auf
ttyS0:

Zeile 33:

```
"interface control" "start"
```

Zeile 37:

```
"interface control" "start"
```

2-Frequenz-halbduplex-Digi mit 1200 /
1200+9600Bd xnet mit UDP und LPT **ptt**:

2-Frequenz-halbduplex-Digi mit 1200 /
1200+9600Bd xnet mit UDP und LPT **PTT**:
**
**

Bei UDP ist die Startreihenfolge egal, die
Programme können auch

Bei UDP ist die Startreihenfolge egal, die
Programme können auch auf
verschiedenen Rechnern laufen

auf verschiedenen Rechnern laufen

```
sudo nice -n -19 aoss ./afskmodem \
```

```
sudo nice -n -19 aoss ./afskmodem \
```

```
-p /dev/parport0 -f 44100 -c 2 -s 9 -l 256 -
b 6 -e 7 \
```

```
-p /dev/parport0 -f 44100 -c 2 -s 9 -l 256 -
b 6 -e 7 \
```

Zeile 42:

```
-M 1 -c 0 -b 1200 -H 40 -q 200 -U
127.0.0.1:9201/9211 -m 0 \
```

Zeile 45:

```
-M 1 -c 0 -b 1200 -H 40 -q 200 -U
127.0.0.1:9201/9211 -m 0 \
```

```
-M 2 -c 0 -b 9600 -a -g -q 200 -U
127.0.0.1:9202/9212 -m 0
```

```
-M 2 -c 0 -b 9600 -a -g -q 200 -U
127.0.0.1:9202/9212 -m 0
```

linuxsnet AUTOEXEC.NET

linuxsnet **(XNET)** AUTOEXEC.NET

```
attach ip0 axudp 1 1 l9200 d9210
127.0.0.1
```

```
attach ip0 axudp 1 1 l9200 d9210
127.0.0.1
```

```
attach ip1 axudp 2 1 l9201 d9211
127.0.0.1
```

```
attach ip1 axudp 2 1 l9201 d9211
127.0.0.1
```

Zeile 51:

Zeile 54:

xnet mit **kiss** und TTY ptt (**xnet** nach
dem Modem starten!)

XNET mit **KISS** und TTY ptt (**XNET** nach
dem Modem starten!)

```
aoss ./afskmodem \
```

```
aoss ./afskmodem \
```

```
-t /dev/ttyS0 -f 24000 -i /tmp
/soundmodem \
```

Zeile 66:

```
144.800MHz 1200Bd, 70cm 1200+9600Bd
xnet, aprsdigi, aprsd und udpbox:
```

– APRS **hoert** auf allen Userzugängen und sendet zum **igate**

– **senden** auf 144.800 nur APRS Messages

– **auf** dem 1200Bd 70cm Zugang normales PR + APRS

– 1. Modem sendet alle **ports** zu udpbox **por**
t 920x und hört auf **port** 921x:

– (auf langsamen Rechnern oder hoher CPU last hilft

– Priotität mit nice oder renice erhöhen)

```
sudo nice -n -19 aoss ./afskmodem \
```

```
-p /dev/parport0 -f 44100 -c 2 -s 9 -l 256 -
b 6 -e 7 \
```

Zeile 80:

```
-M 2 -c 0 -b 9600 -a -g -q 200 -U
127.0.0.1:9202/9212 -m 2
```

– 2. **xnet** empfängt von udpbox und sendet direkt zum Modem:

```
linuxsnet AUTOEXEC.NET
```

```
attach ip0 axudp 1 1 l9300 d9210
127.0.0.1
```

Zeile 89:

```
po 3 baud 9600
```

– 3. udpbox empfängt vom Modem (Port 920x) in AXUDP **und aprsdigi** (9401)

```
-t /dev/ttyS0 -f 24000 -i /tmp
/soundmodem \
```

Zeile 69:

```
144.800MHz 1200Bd, 70cm 1200+9600Bd
xnet, aprsdigi, aprsd und udpbox:
```

+ APRS **hört** auf allen Userzugängen und sendet zum **IGATE.
**

+ **Senden** auf 144.800 nur APRS Messages. **
**

+ **Auf** dem 1200Bd 70cm Zugang normales PR + APRS. **
**

+ 1. Modem sendet alle **Ports** zu udpbox **Por**
t 920x und hört auf **Port** 921x: **
**

+ (auf langsamen Rechnern oder **bei** hoher CPU last hilft Priotität mit nice oder renice erhöhen)

```
sudo nice -n -19 aoss ./afskmodem \
```

```
-p /dev/parport0 -f 44100 -c 2 -s 9 -l 256 -
b 6 -e 7 \
```

Zeile 82:

```
-M 2 -c 0 -b 9600 -a -g -q 200 -U
127.0.0.1:9202/9212 -m 2
```

+ 2. **XNET** empfängt von udpbox und sendet direkt zum Modem: **
**

```
linuxsnet AUTOEXEC.NET
```

```
attach ip0 axudp 1 1 l9300 d9210
127.0.0.1
```

Zeile 91:

```
po 3 baud 9600
```

+ 3. udpbox empfängt vom Modem (Port 920x) in AXUDP (9401) **
**

-	sendet UI von allen Modem Funkports zu aprsdigi (Port 9400)	+	sendet alle UI zu aprsd auf 192.168.1.1: 9000
-	sendet alle UI zu aprsd auf 192.168.1.1: 9000	+	sendet nur "APRS Messages" (-f p58) zu Modem Funkport 1
-	sendet UI von aprsdigi von Port 9401 zu Modem Funkport 2	+	./udpbbox -R 0.0.0.0:9200 -m 192.168.1.1: 9000 -r 127.0.0.1:93:9300\
-	sendet nur "APRS Messages" (-f p58) zu Modem Funkport 1	+	-R 0.0.0.0:9201 -m 192.168.1.1: 9000 -r 127.0.0.1:93:9301\
-	./udpbbox -R 0.0.0.0:9200 -m 127.0.0.1: 9400 -m 192.168.1.1:9000 -r 127.0.0.1:93	+	-R 0.0.0.0:9202 -m 192.168.1.1: 9000 -r 127.0.0.1:93:9302\
-	-R 0.0.0.0:9201 -m 127.0.0.1: 9400 -m 192.168.1.1:9000 -r 127.0.0.1:93		
-	-R 0.0.0.0:9202 -m 127.0.0.1: 9400 -m 192.168.1.1:9000 -r 127.0.0.1:93		
	-M 0.0.0.0:9401 -r 127.0.0.1:9211 -f p58 -r 127.0.0.1:9210 -v		-M 0.0.0.0:9401 -r 127.0.0.1:9211 -f p58 -r 127.0.0.1:9210 -v
-	4. aprsdigi (konfiguration nur zum Test)	+	4. udpbox als aprs-digi (Beispiel)

-	./aprsdigi -v -D --flood WIDE --trace TRACE --notx --int udp:127.0.0.1/9400/16	+	-R empfangen axudp auf Port 9000
-	--tx --norx --int udp:127.0.0.1/9401/16: NOCALL-10,RELAY,WIDE,TRACE	+	
		+	-u bestätige und speichere User Messages an OE0AAA-12 in File /tmp/msg12.txt
		+	
		+	-f filtere für die überlastete 144.800 je nach Geschmack Data-Typen weg
		+	wie z.B. garnicht aprs-frames, thirdparty-messages, Status-Meldungen
		+	weq. Die Zahlen sind der Dezimalwert des 1. Bytes der Nutzdaten
		+	(siehe aprs Protokollbeschreibung APRS101.pdf)

- +
- + **-x filtere Frames mit TCPIP oder NOCALL weg**
- +
- + **-p sende nur (soweit vom Absender richtig adressierte) Frames die**
- + **nach direkt gehört aussehen (first hop digi), sende aber den Rest**
- + **vom Pfad nach Protokoll modifiziert mit für weitere Hops**
- + **füge das digicall OE0AAA-11 zur korrekten Pfad aufzeichnung ein**
- + **anderfalls bliebe der digi unsichtbar.**
- + **Es werden alle Adressierungsarten akzeptiert einschliesslich der**
- + **effizientesten mit Destination-SSID. Damit kann bei (insbesondere von**
- + **Mobilstationen gesendeten) Frames 14 byte "WIDE1-1,WIDE2-2..." oder etwa**
- + **30% eingespart werden.**
- +
- + **-t filtere gleichbleibende Texte (sprich nervige Baken) 27min weg,**
- + **lasse aber (retryende) User Messages nach 28s durch**
- +
- + **-b sende Bake aus dem File aprsbeacon.txt alle 300s, aber ebenfalls**
- + **gefiltert, also wenn der Text nicht zb. durch neue Wetterdaten ersetzt**
- + **wurde, alle 30min**
- +

- + **-k** Filtere alles (ausser User-msq) ausserhalb Umkreis koordinate(grad) /radius(km)
- +
- + **-c** sende zu Monitorzwecken (nc -l -u -p 2000) den sendefertigen Inhalt
- + mit Linefeed an Rechner 192.168.1.24: 2000
- +
- + **-r** sende das gleiche(-e) zum Modem als axudp 127.0.0.1:9100
- +
- + **-v** sagt was es tut und warum auf dem standard output
- +
- + **./udpbbox -v -u OE0AAA-12:/tmp /msg12.txt -R 0.0.0.0:9000**
- + **-f d59,60,125,65-83,85-90,97-122 -x TCPIP,NOCALL -d OE0AAA-11**
- + **-p 5,6,7,8,9 -t 1680,28 -b 300: aprsbeacon.txt -k 48.2/-13.1/40**
- + **-c 192.168.1.24:2000 -e -r 127.0.0.1:9100
**
- +
- + **Man kann noch eine Kopie der ungefilterten rx Daten im monitor-format**
- + **an zb. aprsd für lgate senden (-m 127.0.0.1:9304)**
- + **Weitere parameter siehe -h**
- +
- + **Bakentext File Beispiel:**
- + **(Hier sollte man vorsichtig sein um keine Alarmsymbole zu erwischen**

- + aber Call und Koordinaten ausbessern)
- + Es wird nur die 1. Zeile des Files gesendet, das File kann aber jederzeit
- + zb. von einem Messwert Programm modifiziert werden.
- +
- + OE0AAA-11>TEST,TRACE2-2:!9000.00 N/18000.00E#PHG3750Test Digi
- +
- + dazu auf 266MHz Geode CPU mit billiq-USB-Sound"karte" optimierter Modemstart
- + (-e 50 leichtes rx Hochpassfilter wegen dumpfem nf-Ausgang beim Rx)
- +
- + aoss /home/tc/afskmodem -f 24000 -e 8 -t /dev/ttyS0\
- + -l 128 -b 1 -M 0 -U 127.0.0.1:9000 /9100 -m 0 -e 50 &
- +
- + Startreihenfolge egal, mit & am Ende der Kommandozeile laufen die Programme im Hintergrund
- +
- +
- + I-Gate mit udpgate
- +
- + Rx-Igate kompatibel nach:
- + http://wiki.ham.fi/APRS_iGate_properties#APRS-IS_connection_2
- +
- + ./udpgate -R 0.0.0.0:9000 -t 14580 -s OE0AAA-10 -n 10:netbeacon.txt\

- + **-q www.db0anf.de 14580 -p /etc/pass.txt -f "m/30" -l 6:udp.log -w 14501**
- +
- + **-t TCP Port für Connects mit Aprs-Gattern wie xastir oder weitere igates**
- +
- + **-s Call des Servers**
- +
- + **-n alle 10 Min Netzbake mit Server Position**
- + **File Inhalt (bitte richtige Koordinaten an den gleichen Spalten eingeben!)**
- + **grad minuten.minutenkommas, "/" und "&" ist das Aprs-Symbol**
- +
- + **!8959.00N/01300.20E&lgate Nordpol**
- +
- + **-q Connect zum APRS-IS Netz oder anderem udpgate (Befehl wiederholen dann werden die Server bei Linkausfall der Reihe nach mit 30s Pause versucht)**
- +
- + **-p password oder Filename mit Passwort damits in der Kommandozeile unsichtbar ist**
- +
- + **-f Filterparameter werden zum Server gesendet**
- +
- + **-l Loggt Connects, Frames (gute, gefilterte, duplikate) je nach loglevel**
- + **(das File wird nach jeder Zeile geschlossen und kann gekürzt /gelöscht werden)**

```
netstat -w www Port -t -l -n -e -p tcp
```

Startreihenfolge egal, mit & am Ende der Kommandozeile laufen

die Programme im Hintergrund

Dieses Projekt ist Open Source - Haftung, Verantwortung und Spaß übernimmt jeder selbst.

Aktuelle Version vom 22. November 2019, 18:48 Uhr

Inhaltsverzeichnis

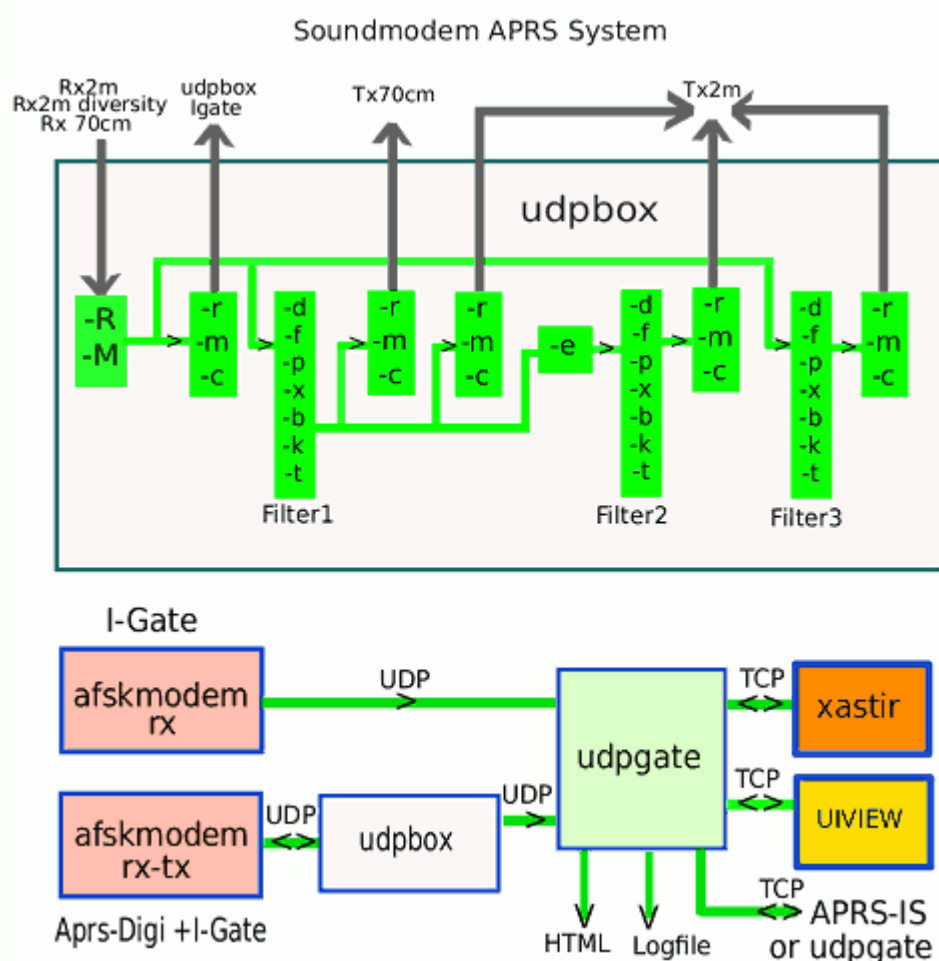
1	Das Projekt	11
2	Der Source Code	11
3	Der kompilierte Treiber	11
4	Starten bzw. Aufrufen des Treibers	12

Das Projekt

Dieser (USB) Soundkartentreiber von OE5DXL soll es ermöglichen mit 2 Kanälen (L und R der Soundkarte) mehrere Modems zugleich unter Linux zu initialisieren.

Als KISS Treiber sind bis zu 16 Modems von 1baud bis 28kbaud möglich. Der Equalizer ermöglicht einen Vollduplexbetrieb bei Verwendung eines getrennten Senders und Empfängers.

Weiterer Vorteil ist die Möglichkeit des **"Multibaud"** Digi, also mehrere Geschwindigkeiten FSK AFSK gemischt (bspw. 1k2 2k4 4k8 9k6 auf einer QRG). In Stereo kann so theoretisch ein multibaud FSK AFSK KISS, als auch AXUDP AX.25 Modem betrieben werden.



Der Source Code

[1]<https://github.com/oe5hpm/dxIAPRS>

[2]<http://gitlab.oe5xbl.ampr.org/oe5hpm/dxIAPRS/>

Der kompilierte Treiber

Soundmodem-bin - Der fertig kompilierte Soundmodem Treiber

udpbox-bin - UDP Filter und RAW-Monitor Konverter und (neu) mit aprs-digi, Bake, User-Message-Receiver

udpgate-bin - I-Gate, APRS-IS, APRS-Server mit Rangefilter, HTML-Statistik, Log

udphub-bin - axudp Hub zum HAMNET-PR-Login ohne IP Beschränkung

udpflex-bin - Interface com-port (/dev/ttySxx) mit KISS oder RMNC bidirektional auf axudp

Starten bzw. Aufrufen des Treibers

mit oss testen 1200 + 9600 baud monitor (ohne kiss oder udp)

```
./afskmodem -f 32000 -M 0 -c 0 -b 1200 -M 1 -c 0 -b 9600 -a -g
```

mit alsa:

```
aoss ./afskmodem -f 32000 -M 0 -c 0 -b 1200 -M 1 -c 0 -b 9600 -a -g
```

APRS mit Xastir KISS-Interface, PTT auf ttyS0:

```
aoss ./afskmodem -i /tmp/soundmodem -t /dev/ttyS0 -f 32000 -M 0 -i
```

Xastir

```
"interface" > "interface control" > "add" "serial kiss tnc"  
"add" "tnc port" /tmp/soundmodem  
"interface control" "start"
```

2-Frequenz-halbduplex-Digi mit 1200 / 1200+9600Bd xnet mit UDP und LPT PTT:

Bei UDP ist die Startreihenfolge egal, die Programme können auch auf verschiedenen Rechnern laufen

```
sudo nice -n -19 aoss ./afskmodem \  
-p /dev/parport0 -f 44100 -c 2 -s 9 -l 256 -b 6 -e 7 \  
-C 0 -b 1 -r 300 -C 1 -b 2 \  
-M 0 -c 1 -b 1200 -q 200 -U 127.0.0.1:9200/9210 -m 0 \  
-M 1 -c 0 -b 1200 -H 40 -q 200 -U 127.0.0.1:9201/9211 -m 0 \  
-M 2 -c 0 -b 9600 -a -g -q 200 -U 127.0.0.1:9202/9212 -m 0
```

linuxsnet (XNET) AUTOEXEC.NET

```
attach ip0 axudp 1 1 l9200 d9210 127.0.0.1
attach ip1 axudp 2 1 l9201 d9211 127.0.0.1
attach ip2 axudp 3 1 l9202 d9212 127.0.0.1
po 1 baud 1200
po 2 baud 1200
po 3 baud 9600
```

XNET mit KISS und TTY ptt (XNET nach dem Modem starten!)

```
aoss ./afskmodem \
-t /dev/ttyS0 -f 24000 -i /tmp/soundmodem \
-c 2 -s 9 -l 256 -b 6 -e 7 -C 0 -r 300 \
-M 0 -c 1 -b 1200 -q 200 -m 0 \
-M 1 -c 0 -b 1200 -H 40 -q 200 -m 0 \
-M 2 -c 0 -b 9600 -a -g -q 200 -m 0
```

linuxsnet AUTOEXEC.NET

```
attach sdev0 kiss 1 3 38400 /tmp/soundmodem
po 1 baud 1200
po 2 baud 1200
po 3 baud 9600
```

144.800MHz 1200Bd, 70cm 1200+9600Bd xnet, aprsdigi, aprsd und udpbox: APRS hört auf allen Userzugängen und sendet zum IGATE.

Senden auf 144.800 nur APRS Messages.

Auf dem 1200Bd 70cm Zugang normales PR + APRS.

1. Modem sendet alle Ports zu udpbox Port 920x und hört auf Port 921x:

(auf langsamen Rechnern oder bei hoher CPU last hilft Priotität mit nice oder renice erhöhen)

```
sudo nice -n -19 aoss ./afskmodem \
-p /dev/parport0 -f 44100 -c 2 -s 9 -l 256 -b 6 -e 7 \
-C 0 -b 1 -r 300 -C 1 -b 2 \
-M 0 -c 1 -b 1200 -q 200 -U 127.0.0.1:9200/9210 -m 2 \
-M 1 -c 0 -b 1200 -H 40 -q 200 -U 127.0.0.1:9201/9211 -m 2 \
-M 2 -c 0 -b 9600 -a -g -q 200 -U 127.0.0.1:9202/9212 -m 2
```

2. XNET empfängt von udpbox und sendet direkt zum Modem:

linuxsnet AUTOEXEC.NET

```
attach ip0 axudp 1 1 l9300 d9210 127.0.0.1
attach ip1 axudp 2 1 l9301 d9211 127.0.0.1
attach ip2 axudp 3 1 l9302 d9212 127.0.0.1
po 1 baud 1200
po 2 baud 1200
po 3 baud 9600
```

3. udpbox empfängt vom Modem (Port 920x) in AXUDP (9401)

sendet alle UI zu aprsd auf 192.168.1.1:9000

sendet nur "APRS Messages" (-f p58) zu Modem Funkport 1

```
./udpbox -R 0.0.0.0:9200 -m 192.168.1.1:9000 -r 127.0.0.1:93:9300\  
-R 0.0.0.0:9201 -m 192.168.1.1:9000 -r 127.0.0.1:93:9301\  
-R 0.0.0.0:9202 -m 192.168.1.1:9000 -r 127.0.0.1:93:9302\  
-M 0.0.0.0:9401 -r 127.0.0.1:9211 -f p58 -r 127.0.0.1:9210 -v
```

4. udpbox als aprs-digi (Beispiel)

-R empfangen axudp auf Port 9000

-u bestätige und speichere User Messages an OE0AAA-12 in File /tmp/msg12.txt

-f filtere für die überlastete 144.800 je nach Geschmack Data-Typen weg wie z.B. garnicht aprs-frames, thirdparty-messages, Status-Meldungen weg. Die Zahlen sind der Dezimalwert des 1. Bytes der Nutzdaten (siehe aprs Protokollbeschreibung APRS101.pdf)

-x filtere Frames mit TCPIP oder NOCALL weg

-p sende nur (soweit vom Absender richtig adressierte) Frames die nach direkt gehört aussehen (first hop digi), sende aber den Rest vom Pfad nach Protokoll modifiziert mit für weitere Hops füge das digicall OE0AAA-11 zur korrekten Pfad aufzeichnung ein anderfalls bliebe der digi unsichtbar. Es werden alle Adressierungsarten akzeptiert einschliesslich der effizientesten mit Destination-SSID. Damit kann bei (insbesondere von Mobilstationen gesendeten) Frames 14 byte "WIDE1-1,WIDE2-2..." oder etwa 30% eingespart werden.

-t filtere gleichbleibende Texte (sprich nervige Baken) 27min weg, lasse aber (retryende) User Messages nach 28s durch

-b sende Bake aus dem File aprsbeacon.txt alle 300s, aber ebenfalls gefiltert, also wenn der Text nicht zb. durch neue Wetterdaten ersetzt wurde, alle 30min

-k Filtere alles (ausser User-msg) ausserhalb Umkreis koordinate(grad)/radius(km)

-c sende zu Monitorzwecken (nc -l -u -p 2000) den sendefertigen Inhalt mit Linefeed an Rechner 192.168.1.24:2000

-r sende das gleiche(-e) zum Modem als axudp 127.0.0.1:9100

-v sagt was es tut und warum auf dem standard output

```
./udpbox -v -u OE0AAA-12:/tmp/msg12.txt -R 0.0.0.0:9000\  
-f d59,60,125,65-83,85-90,97-122 -x TCPIP,NOCALL -d OE0AAA-11\  
-p 5,6,7,8,9 -t 1680,28 -b 300:aprsbeacon.txt -k 48.2/-13.1/40\  
-c 192.168.1.24:2000 -e -r 127.0.0.1:9100
```

Man kann noch eine Kopie der ungefilterten rx Daten im monitor-format an zb. aprsd für lgate senden (-m 127.0.0.1:9304) Weitere parameter siehe -h

Bakentext File Beispiel: (Hier sollte man vorsichtig sein um keine Alarmsymbole zu erwischen aber Call und Koordinaten ausbessern) Es wird nur die 1. Zeile des Files gesendet, das File kann aber jederzeit zb. von einem Messwert Programm modifiziert werden.

```
0E0AAA-11>TEST,TRACE2-2:!9000.00N/18000.00E#PHG3750Test Digi
```

dazu auf 266MHz Geode CPU mit billig-USB-Sound"karte" optimierter Modemstart (-e 50 leichtes rx Hochpassfilter wegen dumpfem nf-Ausgang beim Rx)

```
aoss /home/tc/afskmodem -f 24000 -e 8 -t /dev/ttyS0\  
-l 128 -b 1 -M 0 -U 127.0.0.1:9000/9100 -m 0 -e 50 &
```

Startreihenfolge egal, mit & am Ende der Kommandozeile laufen die Programme im Hintergrund

I-Gate mit udpgate

Rx-Igate kompatibel nach: http://wiki.ham.fi/APRS_iGate_properties#APRS-IS_connection_2

```
./udpgate -R 0.0.0.0:9000 -t 14580 -s 0E0AAA-10 -n 10:netbeacon.txt\  
-g www.db0anf.de 14580 -p /etc/pass.txt -f "m/30" -l 6:udp.log -w 14501
```

-t TCP Port für Connects mit Aprs-Gaffern wie xastir oder weitere igates

-s Call des Servers

-n alle 10 Min Netzbake mit Server Position File Inhalt (bitte richtige Koordinaten an den gleichen Spalten eingeben!) grad minuten.minutenkommas, "/" und "&" ist das Aprs-Symbol

```
!8959.00N/01300.20E&Igate Nordpol
```

-g Connect zum APRS-IS Netz oder anderem udpgate (Befehl wiederholen dann werden die Server bei Linkausfall der Reihe nach mit 30s Pause versucht)

-p password oder Filename mit Passwort damits in der Kommandozeile unsichtbar ist

-f Filterparameter werden zum Server gesendet

-l Loggt Connects, Frames (gute, gefilterte, duplikate) je nach loglevel (das File wird nach jeder Zeile geschlossen und kann gekürzt/gelöscht werden)

-w www Port

www-Statistik-Beispiel

Dieses Projekt ist Open Source - Haftung, Verantwortung und Spaß übernimmt jeder selbst.