The new „Highspeed Multimedia Modem” for the QO-100 NB transponder and the QO-100 Multimedia Beacon

Kurt Moraw, DJ0ABR, Matthias Bopp, DD1US

May 26th 2022, Software Rev. 0.87, Documentation Rev. 2.03

The best source with the latest updates can be found here: https://wiki.amsat-dl.org/doku.php?id=en:hsmodem:start

Introduction:

Overview

QO-100 is in very successful operation. It covers almost half of the earth's surface and thanks to its very stable propagation conditions it offers excellent radio communication and this 24h a day and 7 days a week. So far it has been used mainly in CW/SSB and DATV. To stimulate the use of other digital modes, DJØABR has developed the “Highspeed Multimedia Modem” software.

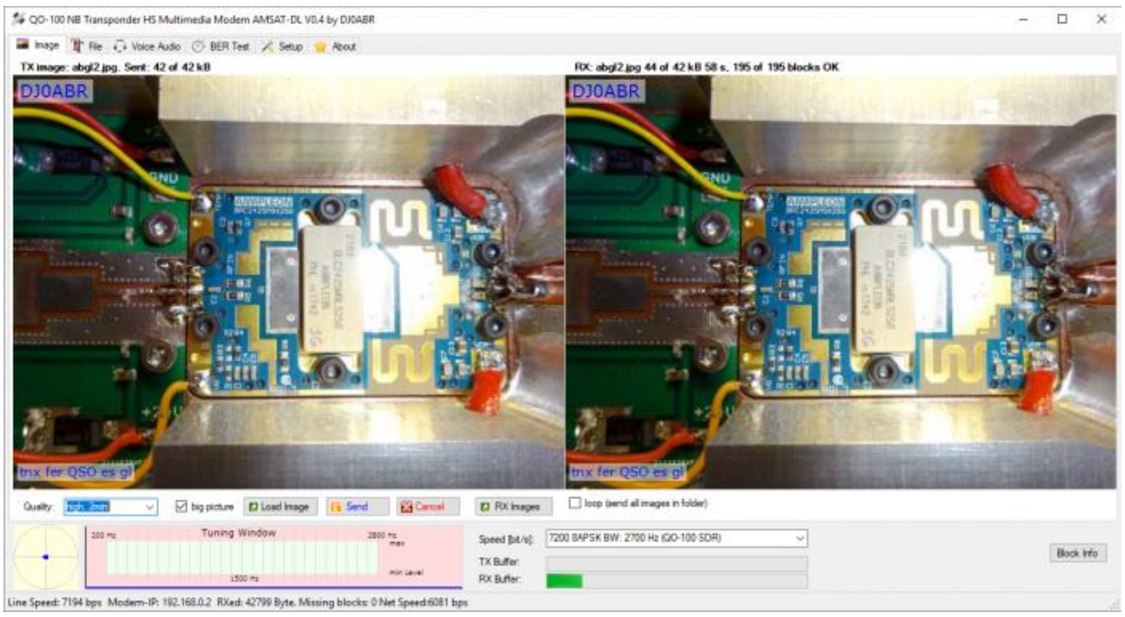
The multimedia high-speed modem is used for fast digital data transmission in a max. 2.7 kHz wide SSB channel. It was developed primarily for the QO-100 satellite, but can also be used on other bands. Which data can be transferred?

- Images (automatic scaling in several quality levels)
- Text files
- HTML (web) pages
- Binary files or any file
- Speech, digitized with two selectable codecs

the op-mode RTTY is also supported.

Transfer Speed

When using an amateur radio transceiver: up to 6000 bit / s
When using SDR solutions (SDR console): up to 7200 bit / s
Detailed information can be found in the Amsat Journal 4Q2020 or on the Internet (see links in the sidebar)



TX image: abgl2.jpg. Sent: 42 of 42 kB

RX: abgl2.jpg 44 of 42 kB 58 s, 195 of 195 blocks OK

Speed [bit/s]: 7200 8APSK BW: 2700 Hz (QO-100 SDR)

TX Buffer:

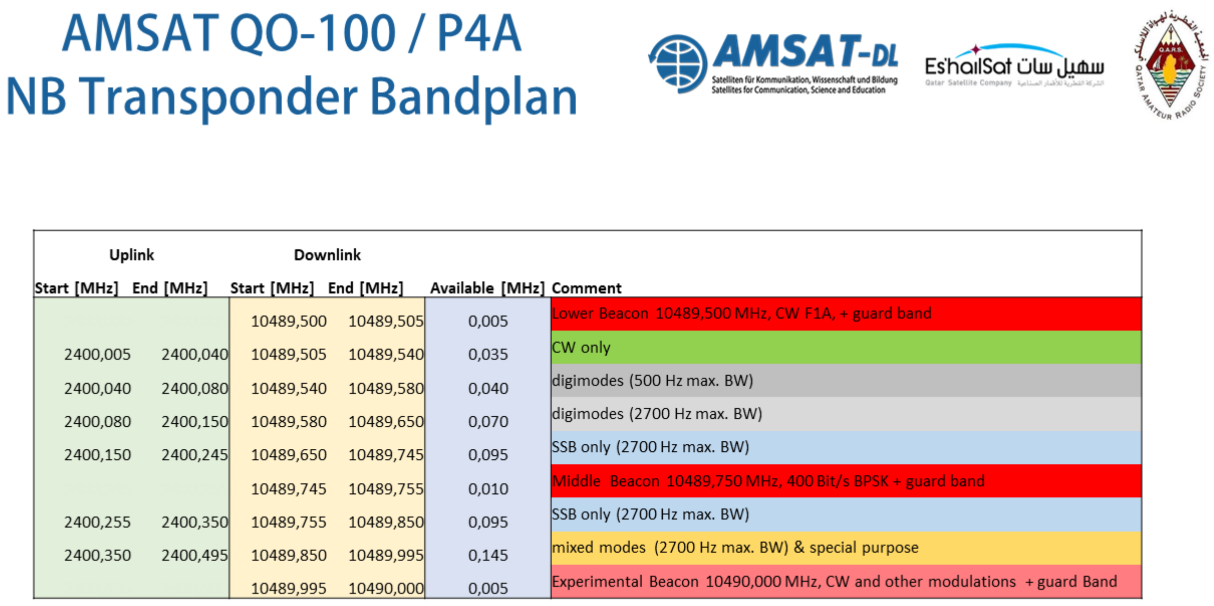
RX Buffer:

Line Speed: 7194 bps Modem-IP: 192.168.0.2 RXed: 42799 Byte, Missing blocks: 0 Net Speed:6081 bps

Technical specifications and theoretical limits

Quote Amsat:

```
* Not stronger than the Beacon
* No FM mode
* Nomodulation exceeding max. 2700 Hz BW
* No digital FM modes like C4FM, DSTAR...
* No transmission below the Lower Beacon
* No transmission above the Upper Beacon
* Excessive signals might trigger LEILA warnings
* Full-Duplex operation is mandatory
* Remote operation over Internet is undesirable
```



AMSAT QO-100 / P4A
NB Transponder Bandplan

Uplink		Downlink			
Start [MHz]	End [MHz]	Start [MHz]	End [MHz]	Available [MHz]	Comment
		10489,500	10489,505	0,005	Lower Beacon 10489,500 MHz, CW F1A, + guard band
2400,005	2400,040	10489,505	10489,540	0,035	CW only
2400,040	2400,080	10489,540	10489,580	0,040	digimodes (500 Hz max. BW)
2400,080	2400,150	10489,580	10489,650	0,070	digimodes (2700 Hz max. BW)
2400,150	2400,245	10489,650	10489,745	0,095	SSB only (2700 Hz max. BW)
		10489,745	10489,755	0,010	Middle Beacon 10489,750 MHz, 400 Bit/s BPSK + guard band
2400,255	2400,350	10489,755	10489,850	0,095	SSB only (2700 Hz max. BW)
2400,350	2400,495	10489,850	10489,995	0,145	mixed modes (2700 Hz max. BW) & special purpose
		10489,995	10490,000	0,005	Experimental Beacon 10490,000 MHz, CW and other modulations + guard Band

HS-Modem Implementation

There are two limit values to be observed, which are prescribed for operation above QO-100: the maximum bandwidth is 2.7kHz and the maximum signal must not exceed the beacon level. These two values result in a window for the maximum achievable transmission speed of digital data.

According to Shannon-Hartley's law, the bandwidth determines the maximum number of symbols per unit of time. Shannon does not initially define how many bits are packed into a symbol.

Since the symbol rate has this physical limit, in order to increase the transmission speed, we have to increase the number of bits/symbol. There are many ways to do this. Using phase modulation, for example, you can change the phase more or less. Each phase position then corresponds to one bit. At this point the second QO-100 rule "no signal stronger than the beacon" kicks in. This rule specifies basically a maximum signal to noise ratio. Noise leads to the points in the constellation diagram being blown up into "clouds".

If too many phase states are used, it will eventually become impossible for the receiver to reliably distinguish individual symbols as noise increases (Fig. 2a-2c).

Since practically no interference, but only Additive White Gaussian Noise (AWGN) influences the signal when transmitting via QO-100, the following formula applies to the maximum bit rate:

max. bit rate [bps] = $B * \log(1+S/N) / \log(2)$,

with B=bandwidth and S/N=signal to noise ratio. We use a maximum permissible bandwidth of 2700 Hz and an S/N of 10, which corresponds to a signal which is +10 dB above noise. This results in a theoretically maximum achievable Bit rate of 9340 bps (bits per second). Of course, one could also expect a higher SNR, but in satellite operation we want to work with low power and stations with small parabolic dishes.

A detailed explanation of these relationships can be found there.

Current transmission methods

In order to set the goals for the new modem, we first look at the common modulation types that are regularly seen in the digital band segment of QO-100:

RTTY:

RTTY is one of the oldest and best-known transmission methods for text. Instead of the usual ASCII code, a 5-bit Baudot code is transmitted by frequency shift keying. A logical “0” corresponds to the frequency 2125 Hz and a “1” to a frequency of 2295 Hz. At the usual bit rate of 45.45 bps, one bit takes 22 ms. During this time, almost 50 oscillations of the signal are transmitted, so the symbol rate is approx. 0.02 S/s. RTTY is extremely inefficient by today's standards, but easy to decode by the simplest means. PSK31 offers a significant improvement, but this shortwave mode is practically never heard on QO-100.

SSTV:

Classic SSTV is used for analogue image transmission. Brightness and colour values are assigned to different AF frequencies and transmitted together with a synchronisation signal. This method originally dates from the 1950s and is a real old-timer, but is still widely used on QO-100 and even the ISS. There are a number of similar coding schemes, on QO-100 you can usually hear “Scottie” or “Martin”. The image resolution varies, on average it is about 350 × 240 pixels. Depending on the method, the transmission time is e.g. 110 seconds. Analogue transmission is difficult to convert into digital speeds. An estimation with “Martin-1” would result in a corresponding bit rate of approx. 6800 bps. Unfortunately, analogue SSTV is very error-prone, since every slightest fluctuation in the received signal has an immediate effect on the picture content.

KGSTV:

This program, developed by JJØOBZ, is used for digital image transmission and is frequently found on QO-100. Pictures with a resolution of 320 x 240 pixels are transferred in blocks of 16 ×16 pixels. There are two methods to choose from: MSK at 1200 bps and 4FSK at 2400 bps. If the receiver was unable to decode individual blocks, it requests these again by transmitting an error list until the picture is complete. Due to the low level of interference on QO-100, however, the pictures are often complete on the first pass.

FreeDV:

The motivation for developing this mode of operation for digital voice transmission was that the digital voice systems (like D-Star and DMR) uses a proprietary codec (AMBE). This CODEC has to be bought as hardware as the software version is practically unaffordable for radio amateurs. Therefore, the free and open source Codec-2 was developed, on which FreeDV is based. On QO-100, work is mostly done in FreeDV-2020. This system was revised for QO-100. It uses an OFDM system with a bit rate of max. 2400 bps (2400A mode) if all OFDM carriers can be used for transmission. Of course, the good signal-to-noise ratio of QO-100 must be exploited to keep the error rate low.

All these systems were primarily developed for communication on short wave, only FreeDV got some adaptations to QO-100. Accordingly, they are of course designed for the high noise level on short wave and a significant part of the valuable bandwidth has to be used for error correction.

The target of the new high-speed modem is to supplement those currently common operating modes.

Technical data of the high-speed modem

The theoretical limit of 9340 bps at a given bandwidth and S/N may have to be corrected to take into account the limited capabilities of the hardware used, especially the transceiver. For example, ICOM transceivers in DATA mode have a fixed Tx bandwidth of 2400 Hz. This results in a maximum bit rate of 8300 bps for an S/N = 10dB. Older transceivers have different bandwidths. Especially near the filter edges the linearity and phase response start to drop off, which is why an additional reserve distance is necessary (excess bandwidth). Further adjustments to the filter curve are made with a digital equalizer. An attempt was made to determine the achievable limits for QPSK and 8APSK modulation by measurements with several different transceivers.

This project is still very young. Especially the modulation procedures will surely be followed by some more steps. For a stable starting point, two modulation types were initially implemented:

- QPSK with a bit rate from 3000 bps to 4800 bps (nominal 4410 bps)
- 8APSK with a bit rate from 5500 bps to 7200 bps (nominally 6000 bps when using an ICOM IC-9700 or 7200 bps when using an SDR such as Pluto or LimeSDR)
- Error correction: Reed Solomon Code [3]
- Full duplex (reception of own transmission possible, important on QO-100)
- Full duplex QSOs in split operation
- Transmission of any data such as images, text, HTML pages, entire web presences, binary data
- Transmission of file names and file size as well as CRC secured transfer of files via QO-100
- Automatic scaling of images according to the desired transmission time
- Automatic ZIP compression
- Digital voice transmission using CODEC-2 or OPUS

Initial tests have been conducted by DJØABR, DH5RAE, DL1EV and DL3MX. Alfred, DL3MX, had set up a very small system with a 40 cm parabolic dish on a tripod. The following results were achieved:

Operating mode	0-error SNR	0-error power	Minimum SNR	Minimum performance
QPSK-4410	+13dB	500mW	+11dB	300mW
8APSK-6000	+19dB	1,25W	+17dB	800mW

The performance data refer to a transmitter which reaches beacon levels at 5 W in SSB mode. In these measurements, the average BPSK-400 beacon was received with an SNR of approx. +25 dB (fluctuating between +24 to +28 dB). Since an error rate of (almost) 0 is already achieved with very small dishes, procedures involving the request of new data blocks are deemed to not be necessary. In case of an error, images are simply sent again, which is no problem with short transmission times of 10 to 30 seconds.

In further work to improve data rate and SNR, attention must always be paid to the widest possible range of use with a large number of radios.

Transceiver requirements

In order to be able to transmit high bit rates, good linearity and a perfect phase response are required. Devices with integrated sound cards, which have been available from all well-known manufacturers for years, work optimally. A connection via the normal microphone socket of older transceivers will only work at the lower transmission rates because of the internal speech filters. SDR transceivers like ADALM Pluto or LimeSDR work very well.

Filters, speech processors, noise blankers (NB) or noise reduction functions (NR) must must always be switched off.

The constellation diagram can be used to assess the reception quality. A too weak, noisy signal can be recognized by diffusely distributed pixels. However, if it stretches the pixels, the phase information is disturbed, e.g. by phase jitter in the received signal. On QO-100 this can be caused by the use of an impure GPS reference in the transmitter or receiver. This is also the reason why an OCXO is recommended instead of GPS for DVB-S2 operation (which is also QPSK modulated). The effects mentioned are shown in the following constellation diagrams using the example of a QPSK-4410 signal.

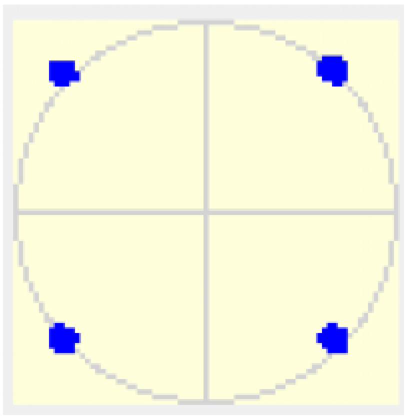


Figure 2a: Optimum constellation diagram for SNR >40 dB

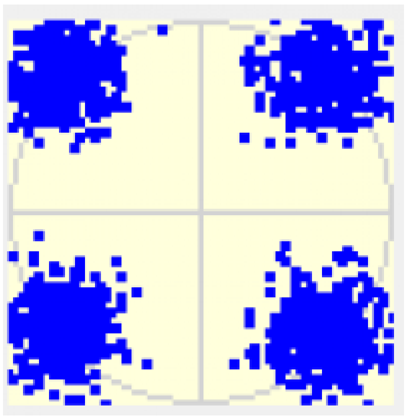


Figure 2b: Constellation diagram via QO-100 at SNR 15 dB

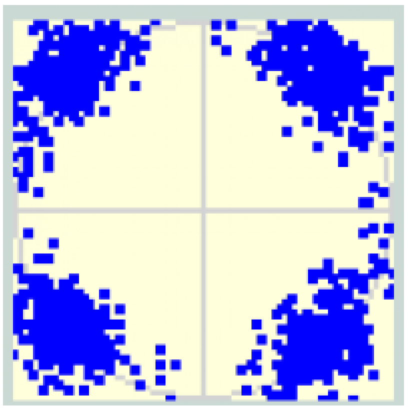


Figure 2c: Constellation diagram with jitter of the LNB reference frequency

Examples of transmission times (8APSK-6000)

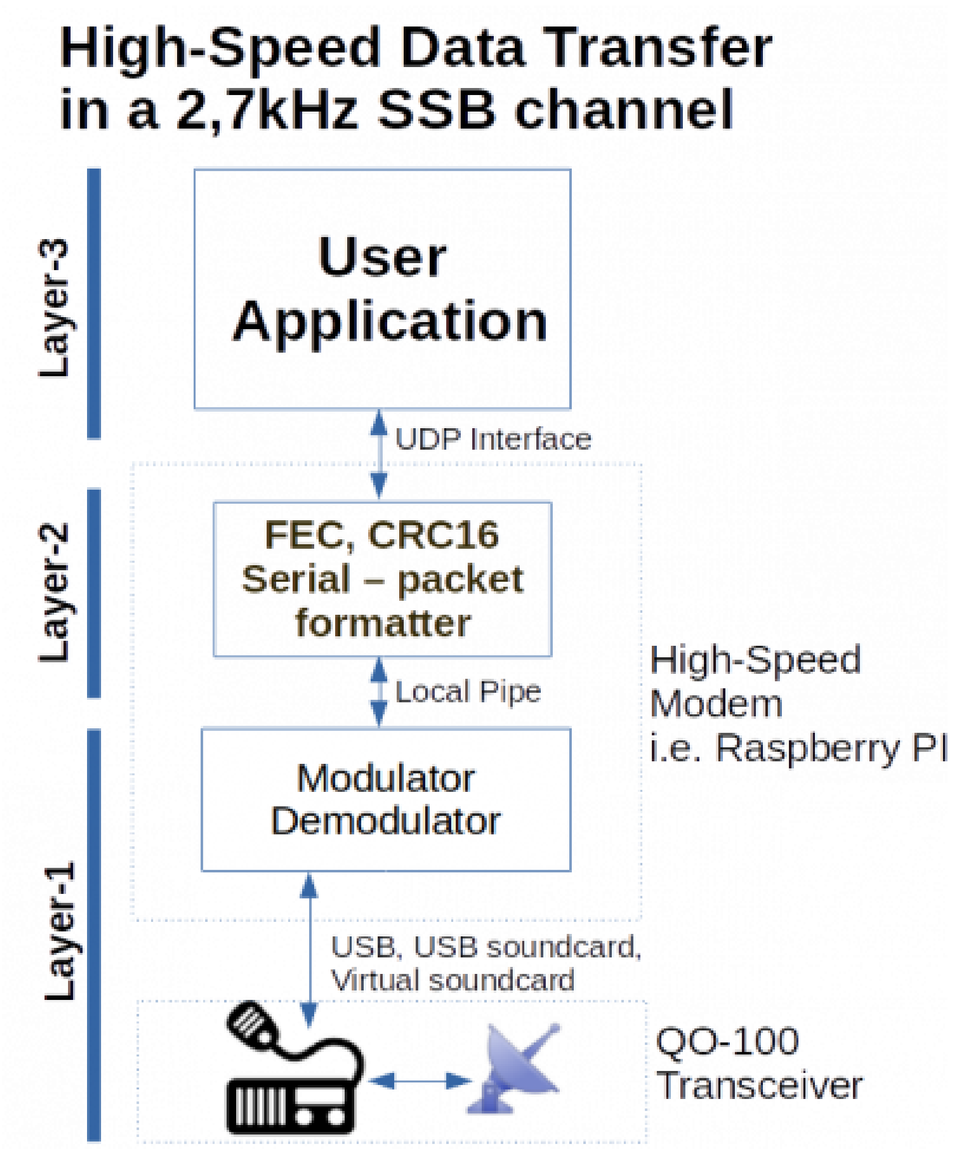
- Image with 320 × 240 pixels (asused for SSTV,KGSTV): approx. 8 seconds
- Image with 640 × 480 pixels (low quality): approx. 33 seconds
- Image with 640 × 480 pixels (medium quality): approx. 55 seconds
- HTML web page with 14 kB: approx. 7 seconds

Especially with texts or HTML pages, net speeds of more than 16 kbps can be achieved through automatic compression. A pure text station presentation can thus be sent in just 2 to 3 seconds, a detailed presentation with an image of the QSL card in about 30 to 40 seconds. Due to these short transmission times, block repetitions are unnecessary in case of errors. If necessary, simply send the file again.

For very large files (the maximum is fixed at 200 kB) HS Multimedia Modem can make additions on the basis of the block number and thus produce an error-free file from two faulty transmissions, unless the same block was faulty by chance.

Structure of the Highspeed Multimedia Modem

Modulator and demodulator were developed in GNU-Radio and then, because of better portability, implemented with Liquid-SDR. Together with the radio equipment they form the (faulty) physical transmission medium.



UDP ports are used as interfaces to the user program. This makes it easier for other developers to use the modem, offers independence from devices and operating systems and provides network capability.

Installation

The Highspeed Multimedia Modem in practice

The Highspeed Multimedia Modem is a system for fast transmission of data in a normal 2.7 kHz voice channel via QO-100. Compared to conventional systems, the transmission speed is increased by a factor of 5 to 10. This modem is suitable for the transmission of images, any files, texts, HTML web files and program files up to complete web presences as well as for digitized speech.

For a station presentation, for example, you can send the image of your QSL card together with cleanly formatted text and the receiver will automatically open the web browser to display the received presentation.

The high-speed modem works full duplex, i.e. transmission and reception run simultaneously. This allows you to check your own transmission, as requested by the satellite operator. A full duplex QSO is possible in split mode (TX and RX frequency offset by e.g. 5 kHz). This allows true intercom in digital speech.

Required equipment

- QO-100 radio system
- PC with Windows (Version 10, probably executable from V7, but not tested)
- PC with Linux (all distributions)
- Single-board computer, Raspberry Pi 4 and Odroid N2, N2+, C2 and C4 (Raspberry Pi 3B+ only with remote network operation)
- Optional for remote network operation: any PC (Windows, Linux) for the user interface.

Installation Windows

The Windows-SETUP Program is located here: HSmodem V0.87 Windows

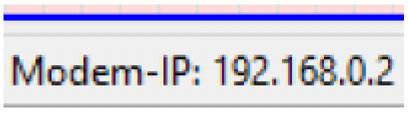
Download the current Windows version “hsmodem_Setup.exe” from the link above and install it. NO changes are made to Windows. The setup program only copies the files to the hard drive and creates the links in the start menu.

The user interface can be found in the start menu under “Amsat” - “hsmodem” and can be started. Windows 10: After the first start, Windows 10 shows two dialogs to obtain permission for network access. You have to confirm both, otherwise the program won't run.

DLLs: if a message about missing DLLs appears when the program starts (msvcp140.dll and others), the Microsoft package: vc_redist.x86.exe must be installed. Please only use the original from Microsoft, this is located here:
https://support.microsoft.com/de-de/help/2977003/the-latest-supported-visual-c-downloads
and the short link to this page:
https://t1p.de/b68y

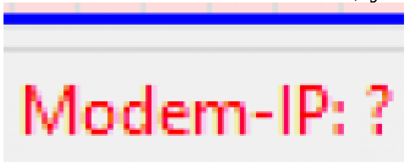
ATTENTION: ALWAYS select the x86 version (vc_redist.x86.exe) even if you have a 64bit PC!

Successful installation and functioning network access can be recognized in the status line by the network address of the modem being displayed (here an example):



Modem-IP: 192.168.0.2

if the network access failed, you can see:



Modem-IP: ?

Installation LINUX

from Github download this file only:
hsmodem.sh
and save it to any directory on the Linux PC (e.g. the home directory).

Example to load the file:

```
wget https://raw.githubusercontent.com/amsat-dl/QO-100-
Modem/main/hsmodem/hsmodem.sh
```

Make the file executable:

```
chmod 755 hsmodem.sh
```

and start the file

```
./hsmodem.sh
```

The system will then be configured and hsmodem will be created from the sources. Once the script has run through (and without errors), hsmodem is ready to use. For more details see on Github README.md

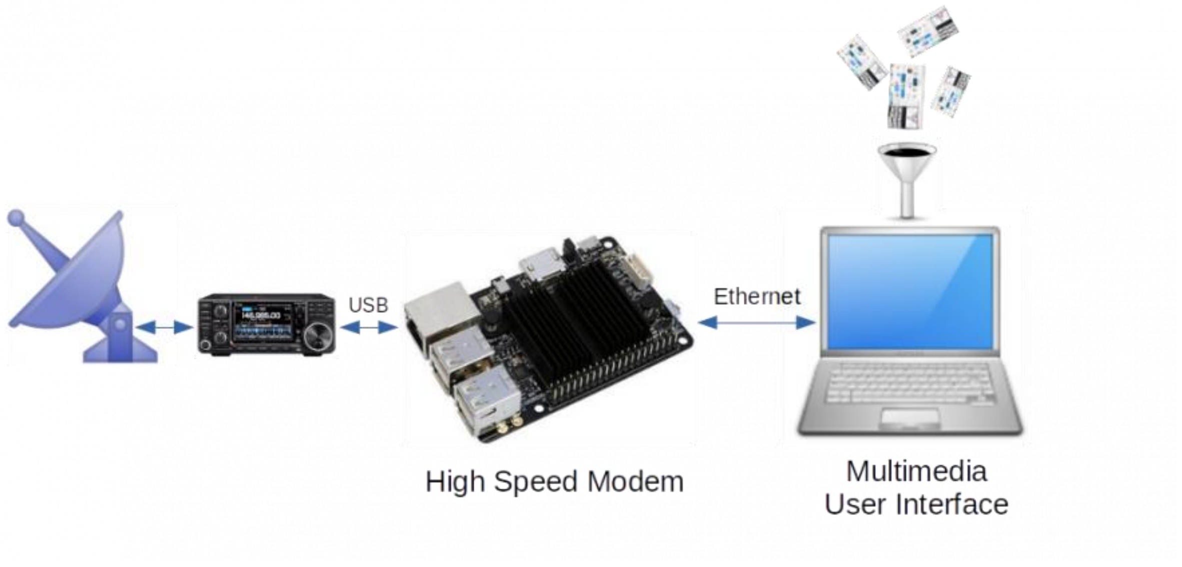
HSmodem requires a reasonably recent Linux in the 64bit version. The scripts are written for Debian/Ubuntu/Mint and derivatives. Other systems like Suse, Fedora may need adjustments in the names of the libraries to be installed.

Note:

GLIBC version 2.27 or newer must be installed (Thanks for the hint Martin, DM4IM)

Connections

Transceiver Connections



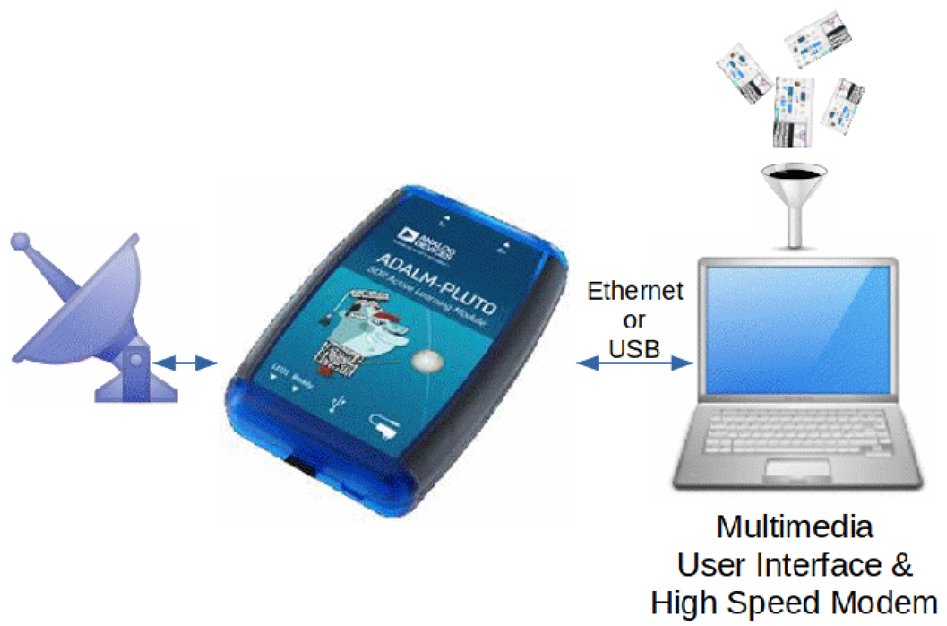
The modem transmits and receives in the LF range (maximum from 150 Hz to 2.85 kHz) via a sound card connected to a suitable transceiver. Many modern transceivers already have a built-in sound card and a USB port, e.g. most ICOM transceivers. In this case you simply connect the modem and transceiver with a USB cable. In figure 4 the modem runs on an Odroid SBC. But it can also run on the PC itself.

In this case you simply connect the modem and transceiver with a USB cable. In the picture the modem is running on an Odroid SBC. It can also run on the PC itself, in which case the single board computer fills and the transceiver is directly connected to the PC via USB.

If the transceiver has no built-in USB card, you can use a USB sound stick or other sound card. USB sound cards with line-in input are ideal. USB sticks that only have a microphone input are less ideal and need a careful setting because they are very sensitive.

Important: for optimal performance of the Highspeed Multimedia Modem the signal must be transmitted with good linearity and clean phase response. Using the normal microphone and loudspeaker connection you will only be able to work with a significantly reduced data rate. Well suited are transceivers with integrated sound card and SDR transmitter/receiver.

SDR transceiver and connection via virtual sound card:

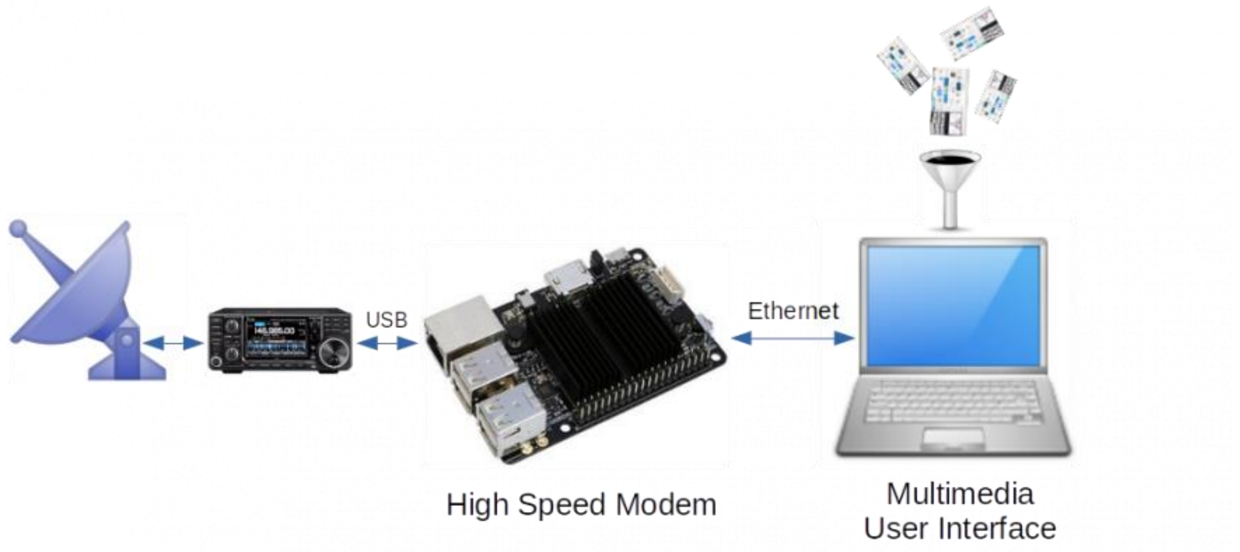


The use of an SDR transceiver is particularly elegant. This receives the IF signal from the LNB and transmits directly on the uplink frequency. It is usually connected to the PC by means of a USB cable. In the case of the ADALM Pluto, this can alternatively be done via an Ethernet connection.

On the PC you now need a suitable software, very widespread is the freeware “SDR-Console” by Simon Brown. The Highspeed Modem can be connected to the used SDR software like any other digital mode of operation (fldigi, freeDV etc.) by means of a virtual sound card (virtual audio cable VAC).

Since this topic depends strongly on the operating system and the software used, it is planned to write a separate documentation for this.

Single-Board-Computer:



If the modem is running on a Raspberry Pi or Odroid, you will usually run the user interface on another PC in your home network. Only a C4 or N2 Odroid has enough computing power to run the modem and user interface simultaneously. You connect the single-board computer to the home network with an Ethernet cable.

The IP address is assigned automatically (DHCP). The recognition of the modem in the home network is also automatic. The user does not need to worry about the network. Once everything is connected, the single-board computer is switched on.

A sufficiently strong power supply and good cables should be used. Since one often loses several 100 mV of the supply voltage when using cheap USB cables, one should not save money here.

Virtual sound cable

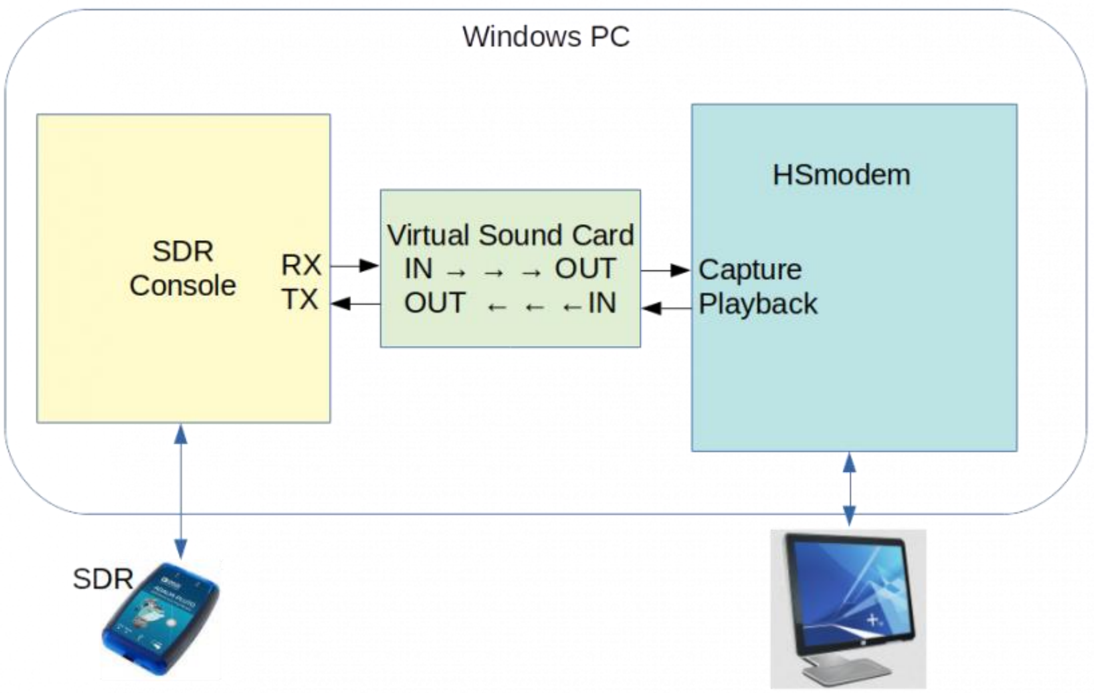
if two programs access sound cards and you want to connect them, you need a virtual sound cable. The best example of this is the connection of the SDR console with other programs (HSmodem, SSTV, RTTY …)

The **SDR console** delivers the received and demodulated NF signal to a sound card, where you can e.g. connect to a **loudspeaker** .

HSmodem expects the NF signal from the **microphone input** of a sound card.

How do you connect the output of the SDR console to the input of the HSmodem? This is done with the help of a virtual sound cable (VAC). A VAC couples an audio output with an audio input. If you not only want to receive but also to send, you need a separate VAC for each direction.

This is only about Windows (of course there is something like that for Linux too):



Receiving way: SDR (e.g. Pluto) → SDR console → VAC → HSmodem → monitor

We tested the virtual sound cable from vac.muzychenko.net, which works well, it can be purchased from Autor for little money.

Another VAC is available from vb-audio.com. In the simplest version, this also offers only one channel, but can be expanded to a higher version through a donation (donationware).

The free versions of the various providers only ever have one audio channel. Since we need two channels for sending and receiving, either the commercial version or two free versions from different manufacturers are used.

There are certainly other providers, but they have not been tested in this context.

More information on configuration: see chapter **SDR console**

USER INTERFACE GUI

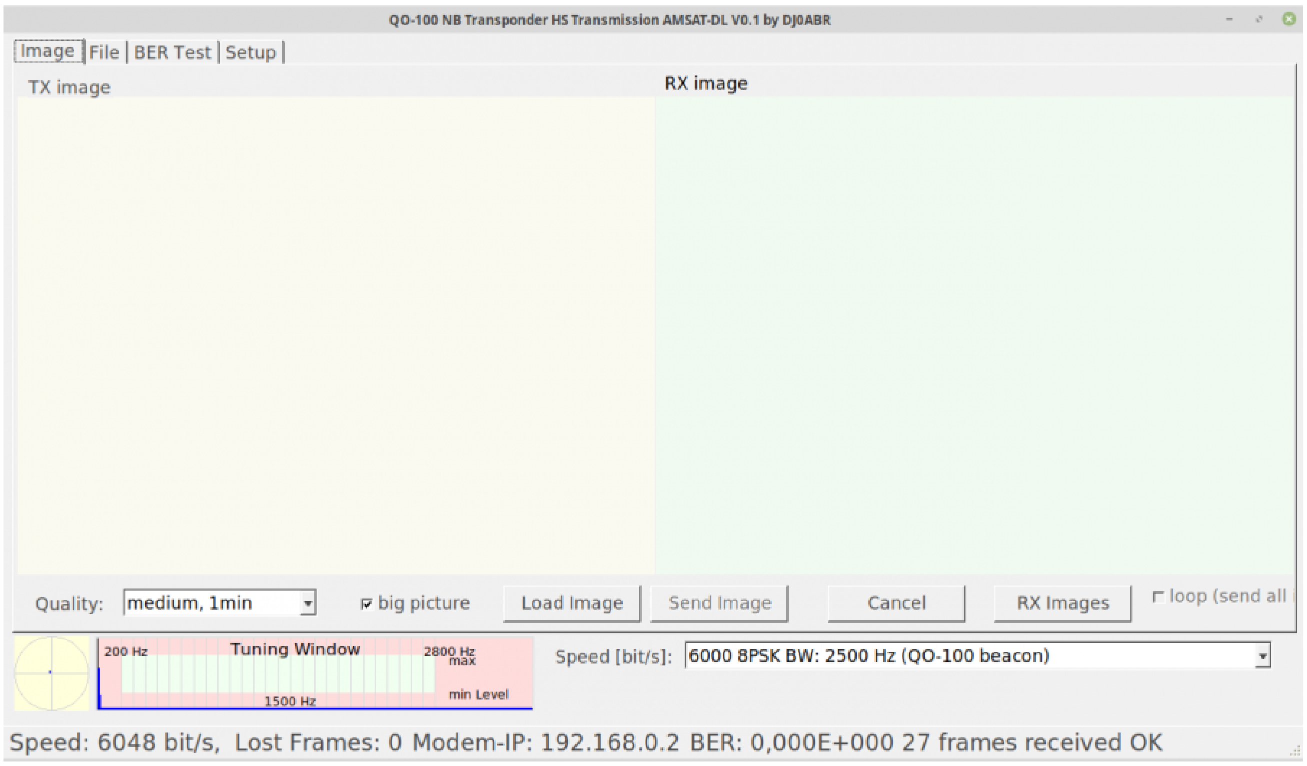
User Interface / Operation

The user interface will run on any computer (Windows PC, Linux PC, single-board computer, probably also Mac-OS, which has not been tested yet). After starting the program, the modem is automatically found in the home network, no matter whether it is running on the same or another computer. The program is called "oscardata.exe" and is located on the SD card in the directory: /home/odroid/modem or /home/pi/modem or as a download on this page [5].

Windows: the program is installed as described and started from the menu as usual.

Linux: the file runs under the Linux package: mono-complete from version 6.12.0. In the preconfigured images this is already installed. To start the program, you open a console and go to /home/odroid/modem or /home/pi/modem and start there: "mono oscardata.exe".

After starting oscardata.exe you will see the user interface (Fig. 7), for the time being still without content.



Quality: medium, 1min

☑ big picture

☐ loop (send all

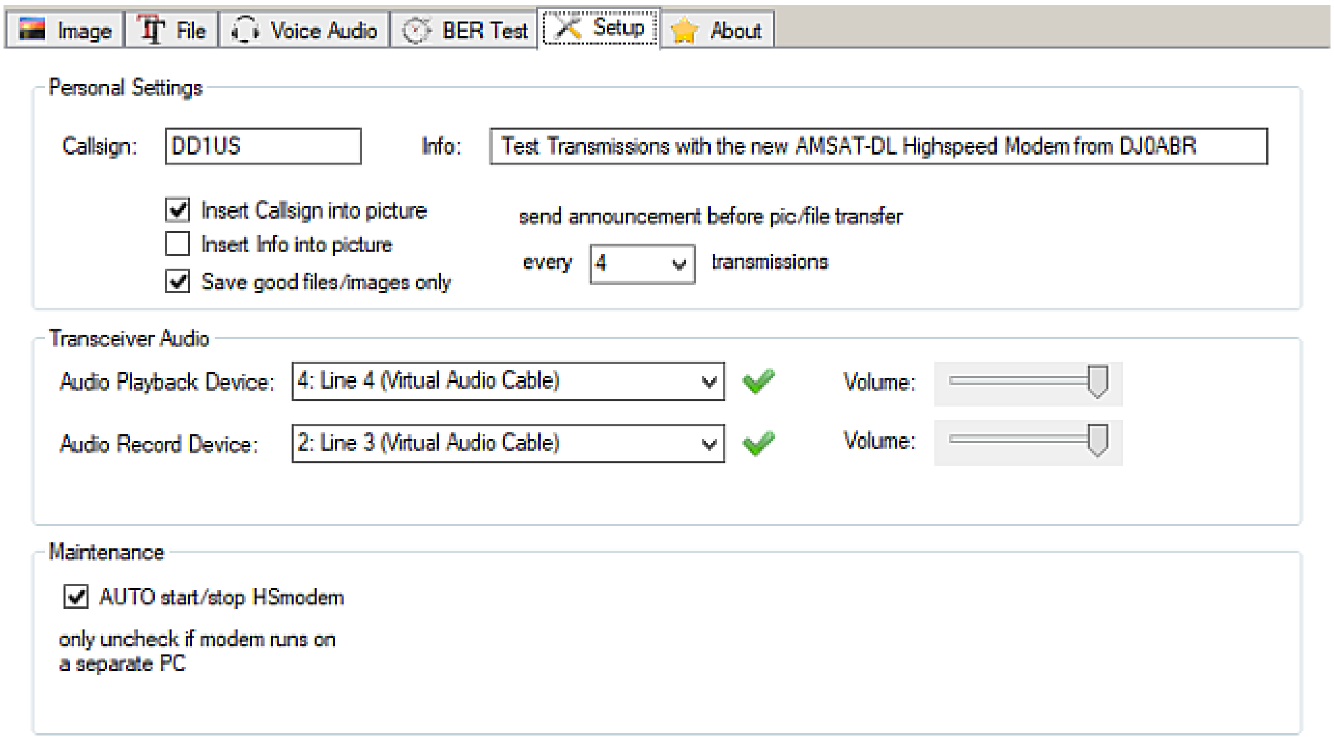
Speed [bit/s]: 6000 8PSK BW: 2500 Hz (QO-100 beacon)

Speed: 6048 bit/s, Lost Frames: 0 Modem-IP: 192.168.0.2 BER: 0,000E+000 27 frames received OK

The IP address of the modem appears in the status line. If not, there is no network connection yet. As soon as the transceiver is switched on you can see noise in the constellation diagram on the lower left.

Setup:

In the menu item “Setup” the necessary basic settings are defined.



Personal Settings

Callsign: DD1US

Info: Test Transmissions with the new AMSAT-DL Highspeed Modem from DJ0ABR

[✓] Insert Callsign into picture

[ ] Insert Info into picture

[✓] Save good files/images only

send announcement before pic/file transfer

every 4 transmissions

Transceiver Audio

Audio Playback Device: 4: Line 4 (Virtual Audio Cable) ✓ Volume:

Audio Record Device: 2: Line 3 (Virtual Audio Cable) ✓ Volume:

Maintenance

[✓] AUTO start/stop HSmodem

only uncheck if modem runs on a separate PC

- `Personal Settings:`

First you enter your own callsign, the info text is optional. This text and the callsign can be inserted into a transmitted picture. Whether the type of modulation and data rate used is transmitted before each transmission by means of an artificial voice or, as in the example (Fig. 8), before every fourth transmission can be freely defined.

- `Transceiver Audio:`

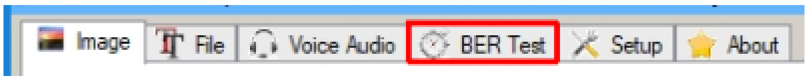
Here you define the audio sources for the transmitter (Audio Playback Device) or the audio sink for the receiver (Audio Record Device). These can be the microphone and the loudspeaker of the PC. In the example two “virtual audio cables” are used to connect the Highspeed Modem with the SDR-Console. The volume settings can be optimized later.

- `Maintenance:`

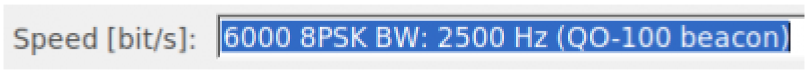
If the modem is running on the same PC as the user interface, a tick must be placed here. If a Raspberry Pi or Odroid computer is used for the modem and another computer is used for the user interface, you should uncheck this option.

Send and receive test data simultaneously:

In the “BER Test” window, test data can be sent and received to verify the correct functioning of your own system.



Different bandwidths and speeds are available (Fig. 10). For operation on QO-100, 4410 bps QPSK and 6000 bps 8APSK modes are recommended. For the first tests, 3000 QPSK should be selected, since this is where the requirements on the radio system are lowest.

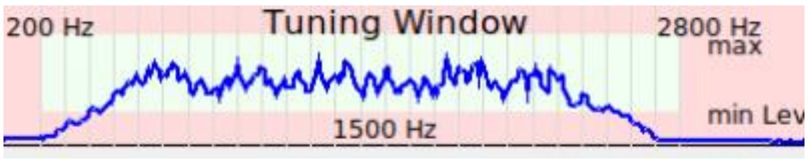


Speed [bit/s]: 6000 8PSK BW: 2500 Hz (QO-100 beacon)

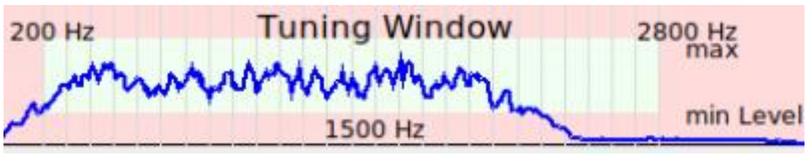
Now click the “START” button and the modem will send test data. Switch the transceiver manually to transmit (there is no automatic PTT yet) and check the output power. Your received own signal must always be below beacon level. When using an SDR-TRX and the SDR-Console you can use their VOX function. There are two possibilities to adjust the transmitter::

1. the level of the sound card and
2. the setting of the transmission power.

Details of how to set this on the IC-9700 are given in a later chapter. For reception, it is important that the input level of the sound card is set so that the spectrum display is in the green range.

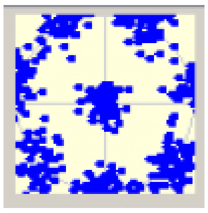


There can sometimes be a small offset between the transmit and receive frequencies. Even with GPS stabilised systems there can be a deviation of up to ±50 Hz depending on the system. The RIT function of the transceiver corrects any deviations to place the spectrum in the middle of the green area. This Figure shows a wrong setting, the frequency is set too low here.

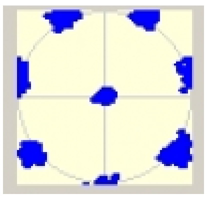


The modem has a capture range of approx. ±200 Hz, but the transceiver's receive filters are usually not as wide and would weaken the signal at the edges if they were set incorrectly. An exact setting is also possible with the constellation diagram.

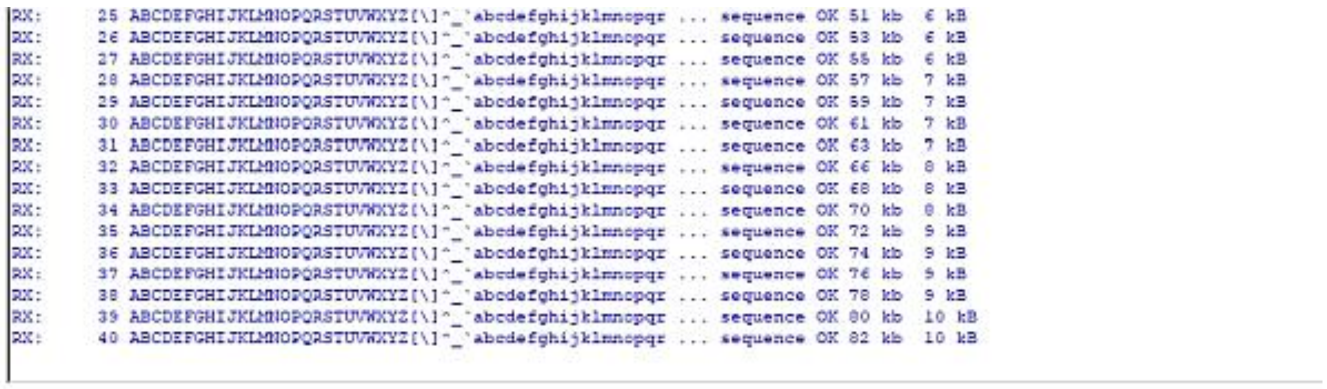
The points should be as sharp as possible. Here are two examples with 6000 bps 8APSK:



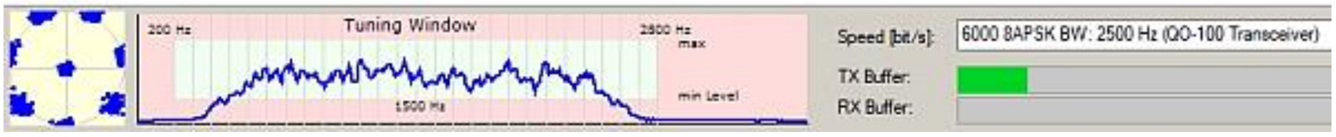
Wrong setting, the signal is very noisy, the error rate is high



Correct setting with low error rate As soon as the level and frequency are set correctly, the test data runs through the window.



```
RX:      25 ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqr ... sequence OK 51 kb  6 kB
RX:      26 ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqr ... sequence OK 53 kb  6 kB
RX:      27 ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqr ... sequence OK 55 kb  6 kB
RX:      28 ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqr ... sequence OK 57 kb  7 kB
RX:      29 ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqr ... sequence OK 59 kb  7 kB
RX:      30 ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqr ... sequence OK 61 kb  7 kB
RX:      31 ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqr ... sequence OK 63 kb  7 kB
RX:      32 ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqr ... sequence OK 66 kb  8 kB
RX:      33 ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqr ... sequence OK 68 kb  8 kB
RX:      34 ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqr ... sequence OK 70 kb  8 kB
RX:      35 ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqr ... sequence OK 72 kb  9 kB
RX:      36 ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqr ... sequence OK 74 kb  9 kB
RX:      37 ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqr ... sequence OK 76 kb  9 kB
RX:      38 ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqr ... sequence OK 78 kb  9 kB
RX:      39 ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqr ... sequence OK 80 kb  10 kB
RX:      40 ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqr ... sequence OK 82 kb  10 kB
```

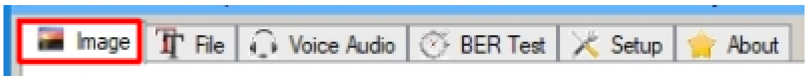


Speed [bit/s]: 6000 8APSK BW: 2500 Hz (QO-100 Transceiver)
TX Buffer:
RX Buffer:

If all adjustments are properly done, “sequence OK” is always displayed. Error messages (frame lost) should occur very rarely. Of course, you can receive test data from other stations as well.

Send and receive images:

If the test data has been received correctly as explained in the previous chapter, you can now switch to the menu item “Image” and try to receive and transmit images:



Quality:

Four resolutions (detail, sharpness) can be selected. This setting determines the transmission time, which is given as a rough estimate. With the setting “medium, 1min”, very sharp pictures can already be sent. Higher „Quality“ settings are only required for extreme requirements.

Big Picture:

The images are automatically scaled regardless of the image format. Almost any picture type and resolution can be sent. The scaling depends on the setting “big picture”: not activated: 320 × 240 pixels, activated: 640 × 480 pixels.

Load Image:

Click on “Load Image” to select an image to be sent. The storage location for images to be sent can be selected via the embedded file browser..

Send:

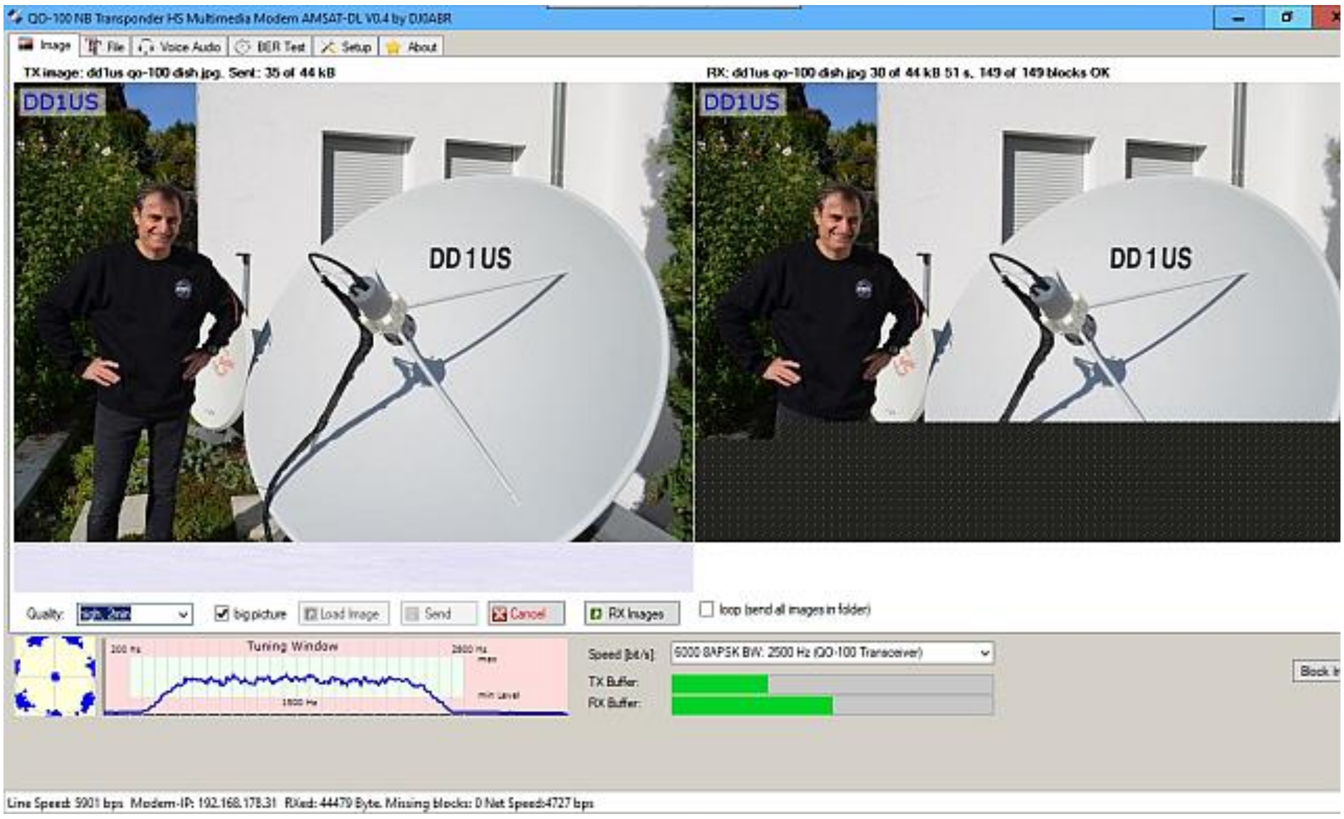
Click “Send” to send the selected image.

Rx Images:

The storage location for received images can be accessed by clicking on “RX Images”.

Loop (send all images in folder):

When this is enabled, all images in the same folder are sent in an endless loop.


QO-100 NB Transponder HS Multimedia Modem AMSAT-DL V0.4 by DJ0ABR

TX image: dd1us qo-100 dish jpg. Sent: 35 of 44 kB

RX: dd1us qo-100 dish jpg 30 of 44 kB 51 s, 149 of 149 blocks OK

Quality: [illegible]

☑ big picture

☐ loop (send all images in folder)

Speed [bit/s]: 6000 8APSK BW: 2500 Hz (QO-100 Transceiver)

Line Speed: 5901 bps Modem-IP: 192.168.178.31 RXed: 44479 Byte. Missing blocks: 0 Net Speed:4727 bps

In the left part in this screenshot you can see the original picture, already scaled automatically.

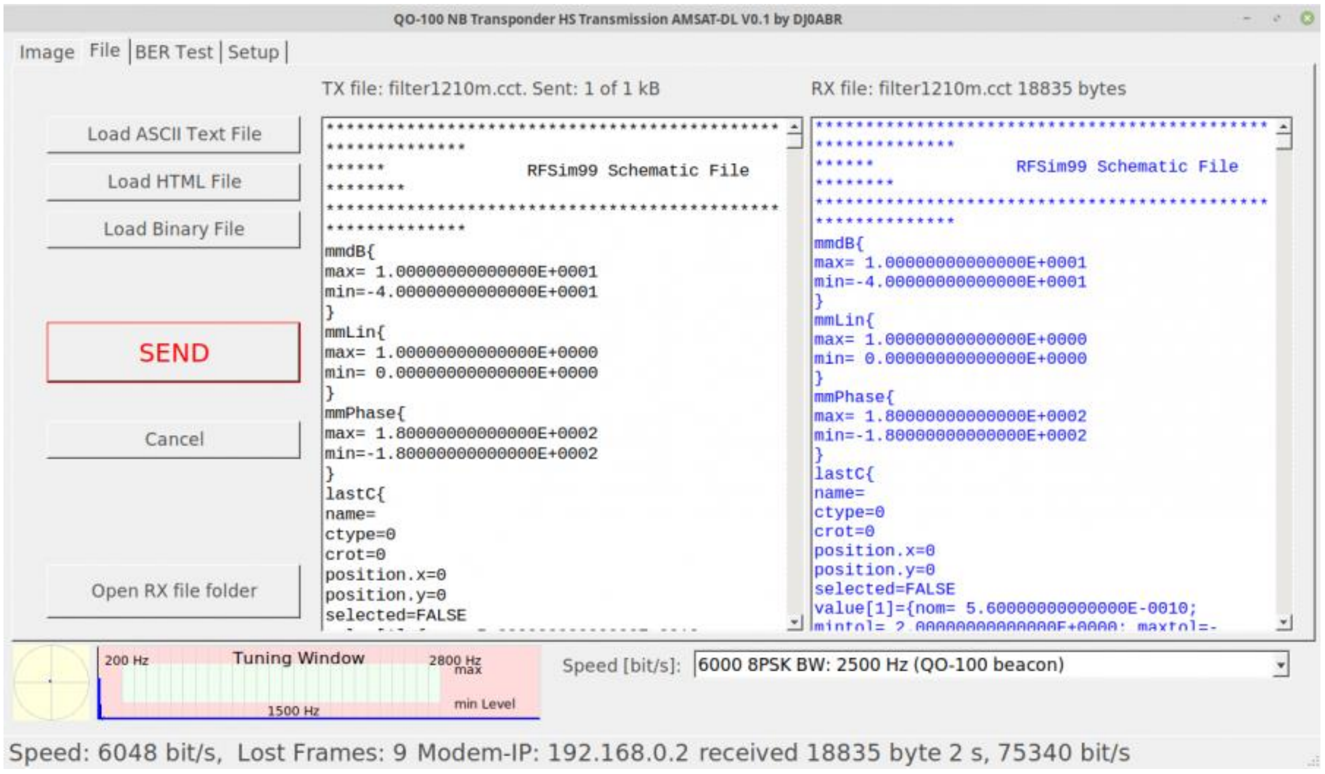
In the right part of the screenshot you can see the image as received via QO-100.

In the example above the transmission is approximately 2/3 complete.

Above the received image, the file name, received bytes, file size, transmission time, number of correctly received blocks and total number of received blocks are displayed. If reception is correct, it is a 100% copy of the left image.

Send and receive files:

The menu item “File” permits the transmission of ASCII, HTML and binary files.



TX file: filter1210m.cct. Sent: 1 of 1 kB

```
*********************************************
**************
******                  RFSim99 Schematic File
********
*********************************************
**************
mmdB{
max= 1.00000000000000E+0001
min=-4.00000000000000E+0001
}
mmLin{
max= 1.00000000000000E+0000
min= 0.00000000000000E+0000
}
mmPhase{
max= 1.80000000000000E+0002
min=-1.80000000000000E+0002
}
lastC{
name=
ctype=0
crot=0
position.x=0
position.y=0
selected=FALSE
```

RX file: filter1210m.cct 18835 bytes

```
*********************************************
**************
******                  RFSim99 Schematic File
********
*********************************************
**************
mmdB{
max= 1.00000000000000E+0001
min=-4.00000000000000E+0001
}
mmLin{
max= 1.00000000000000E+0000
min= 0.00000000000000E+0000
}
mmPhase{
max= 1.80000000000000E+0002
min=-1.80000000000000E+0002
}
lastC{
name=
ctype=0
crot=0
position.x=0
position.y=0
selected=FALSE
value[1]={nom= 5.60000000000000E-0010;
mintol= 2.00000000000000E+0000; maxtol=-
```

Speed [bit/s]: 6000 8PSK BW: 2500 Hz (QO-100 beacon)

Speed: 6048 bit/s, Lost Frames: 9 Modem-IP: 192.168.0.2 received 18835 byte 2 s, 75340 bit/s

Load ASCII Text File:

In this menu item you select a pure text file for transmission. It is shown in the left window.

SEND:

Sends the file. If you receive yourself, the received text file is displayed on the right. In the example, a text file describing the installation of the Highspeed Multimedia Modem on a Raspberry Pi was transmitted at an effective transmission speed of 8761 bps. Since text files can be compressed well, this high speed was achieved. Compression is automatically applied to all files except image files..

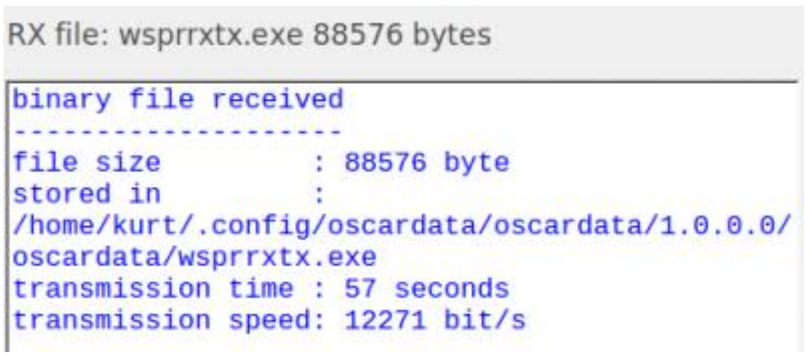
Load HTML File:

In this menu item you select an HTML file to be sent. HTML (web) files are transmitted in the same way as text files, except that a web browser opens automatically when received and displays the HTML pag.

Load Binary File:

Analogous to the previous points, a binary file is selected here.

Of course, binary files are not displayed after reception. Instead, the right-hand window shows transmission statistics.



RX file: wsprrxtx.exe 88576 bytes

```
binary file received
--------------------
file size         : 88576 byte
stored in         :
/home/kurt/.config/oscardata/oscardata/1.0.0.0/
oscardata/wsprrxtx.exe
transmission time : 57 seconds
transmission speed: 12271 bit/s
```

Open RX file folder:

All received files can be accessed.

Digital voice transmission:

The high-speed multimedia modem would not deserve the name if it could not also transmit digital speech. Therefore, this possibility was also implemented in the menu item "Voice Audio"8).

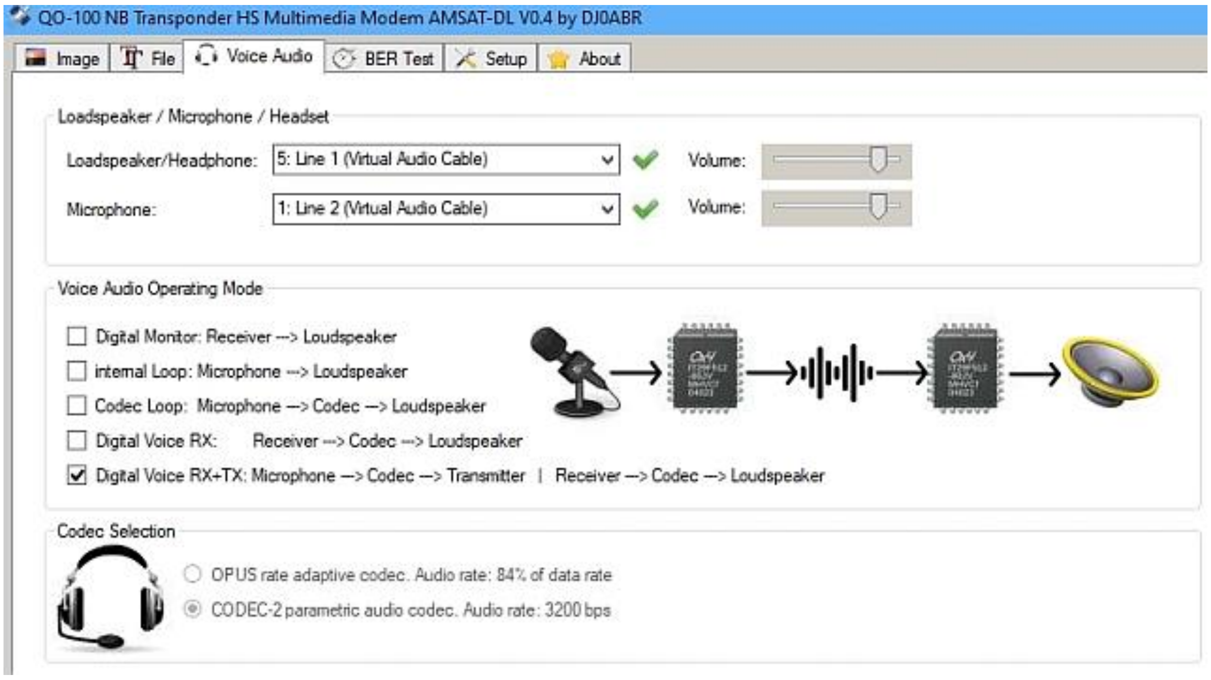

QO-100 NB Transponder HS Multimedia Modem AMSAT-DL V0.4 by DJ0ABR

Image | File | Voice Audio | BER Test | Setup | About

Loadspeaker / Microphone / Headset

Loadspeaker/Headphone: 5: Line 1 (Virtual Audio Cable) ✓ Volume:

Microphone: 1: Line 2 (Virtual Audio Cable) ✓ Volume:

Voice Audio Operating Mode

☐ Digital Monitor: Receiver ---> Loudspeaker

☐ internal Loop: Microphone ---> Loudspeaker

☐ Codec Loop: Microphone ---> Codec ---> Loudspeaker

☐ Digital Voice RX: Receiver ---> Codec ---> Loudspeaker

☑ Digital Voice RX+TX: Microphone ---> Codec ---> Transmitter | Receiver ---> Codec ---> Loudspeaker

Codec Selection

○ OPUS rate adaptive codec. Audio rate: 84% of data rate

● CODEC-2 parametric audio codec. Audio rate: 3200 bps

Loudspeaker / Microphone / Headset:

Here the inputs and outputs of the sound card or, as in the example (Fig. 18), virtual audio cables (Virtual Audio Cable VAC) are selected.

Voice Audio Operating Mode:

One of the five operating modes can be selected at a time:

- Digital Monitor: the reception signal is looped through directly to the loudspeaker. This mode is used to observe the frequency or to evaluate the received signal.
- Internal Loop: the microphone is connected directly to the loudspeaker. This allows the function of the sound card and the volume settings to be checked.
- Codec Loop: in this test mode the selected codec is additionally looped in. This allows you to check the actual quality of the audio transmission at the different bit rates without going on air.
- Digital Voice RX: in this mode you can listen to a running Highspeed Modem QSO
- Digital Voice RX+TX: this is the actual QSO mode
-

If you offset Tx and Rx frequency by at least 5 kHz, true intercom (full duplex) is possible.

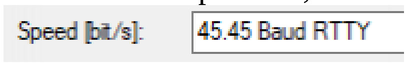
Codec Selection:

Here you can choose between two voice codecs. CODEC-2 works with a fixed audio rate of 3200 bps and is particularly suitable for low data rates. OPUS scales the audio rate in a ratio of 84 % to the data rate and offers a more natural speech quality at higher data rates (8APSK modes).

RTTY operation

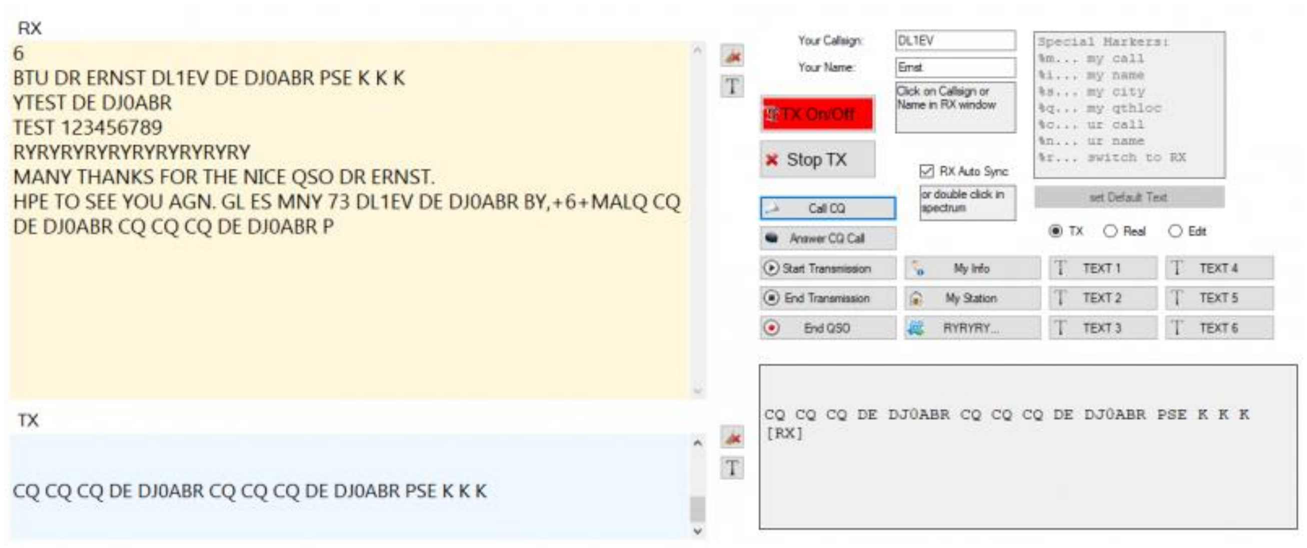
Even if RTTY is a slow operating mode, it is still very popular and has therefore been integrated into the HSmodem software.

To set RTTY operation, select the appropriate transmission speed:



Speed [bit/s]: 45.45 Baud RTTY

and then go to the RTTY window:



RX

```
6
BTU DR ERNST DL1EV DE DJ0ABR PSE K K K
YTEST DE DJ0ABR
TEST 123456789
RYRYRYRYRYRYRYRYRYRY
MANY THANKS FOR THE NICE QSO DR ERNST.
HPE TO SEE YOU AGN. GL ES MNY 73 DL1EV DE DJ0ABR BY,+6+MALQ CQ
DE DJ0ABR CQ CQ CQ DE DJ0ABR P
```

TX

```
CQ CQ CQ DE DJ0ABR CQ CQ CQ DE DJ0ABR PSE K K K
```

Your Callsign: DL1EV
Your Name: Ernst
Click on Callsign or Name in RX window

[✓] RX Auto Sync
or double click in spectrum

```
Special Markers:
%m... my call
%i... my name
%s... my city
%q... my qthloc
%c... ur call
%n... ur name
%r... switch to RX
```

[✓ TX] Real Edit

```
CQ CQ CQ DE DJ0ABR CQ CQ CQ DE DJ0ABR PSE K K K
[RX]
```

The yellow RX window shows all received characters. HSmodem-RTTY is full-duplex, i.e. Reception is always active, even during transmission. So you can hear your own transmission back via QO100.

The blue TX window is used for keyboard input, it is activated as soon as you are transmitting.

activate transmission:

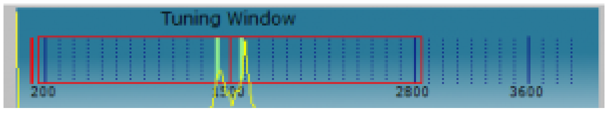
1. press the TX On / Off button, or
2. send a predefined line. At the beginning of each predefined transmission, the system automatically switches to TX.

Frequency tuning:

RTTY works with 2 frequencies, Mark and Space. These two frequencies show two peaks in the spectrum display. Reception is possible when these peaks are on the two green lines. There are two options:

1. Automatic tuning: activate **RX Auto Sync**
2. Double click in the spectrum window

the automatic SYNC needs approx. 1-2s for tuning. If several RTTY stations are transmitting close together, you have to switch it off and manually select the desired signal by double-clicking.



Data of the QSO partner:

To enter the callsign and name of the QSO partner in the corresponding input fields, click on the callsign or the name in the yellow RX window, the entry is then made automatically. Manual entry is of course also possible.

Text memory:

There are 14 memory buttons. 8 of them are preset with the usual texts, but can be changed.

Three functions can be selected: TX, Real and Edit.

- Edit … switches to the input mode. The text can be changed as required in the text field under the buttons. Special characters are available, see help text in the box on the screen.
- Real … display of the text as it is sent out. The special characters have been replaced with real text.
- TX … pressing a memory button sends the text memory.

Erase window content and fonts:

to do this, press one of the small buttons next to the RX or TX window.

Live data transmission: Streaming

Introduction

The high-speed modem can transfer files, i.e. images, texts, or any other files. Another operating mode has been added since version 0.80: data streaming, sending of external data streams.

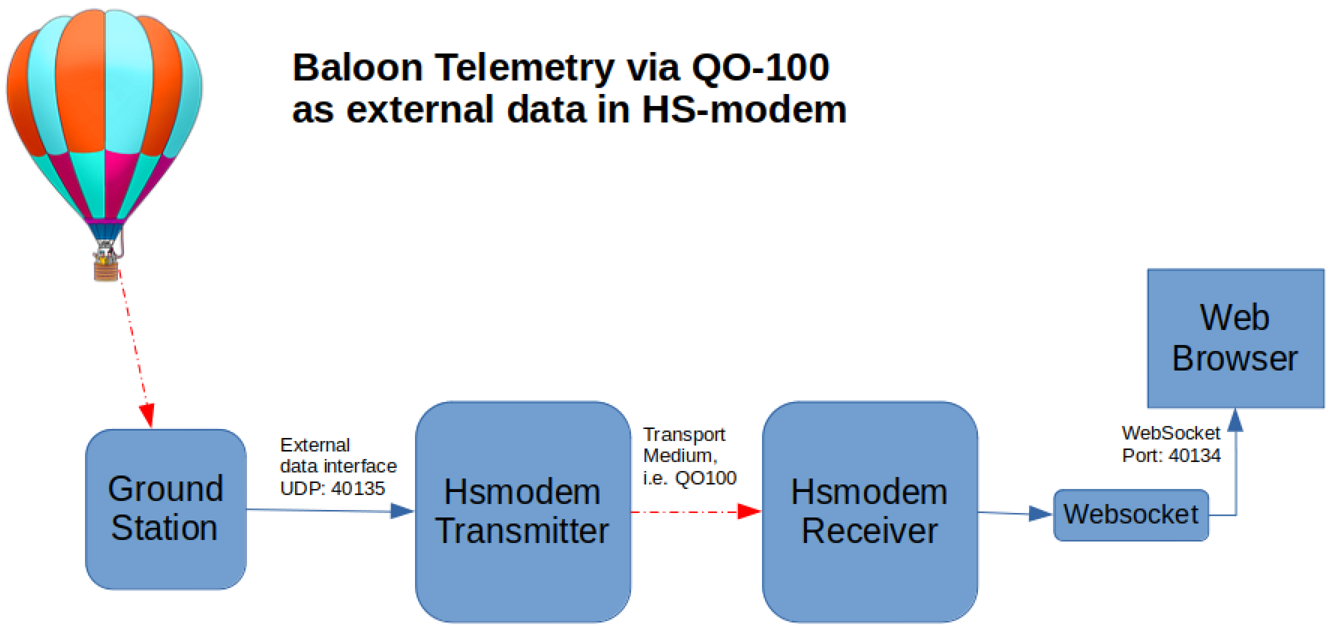
What are streams

Example: A temperature sensor of a weather station delivers a temperature value every few minutes, or a rev counter of a model car delivers the engine speed, an APRS station delivers continuously new APRS position data, an SDR generates a stream of spectrum / waterfall data. All of these are data streams.

This description is about data from any external sources which are continuously delivered and updated.

Practical example:

In a balloon experiment one would like to transfer the current balloon flight data via QO-100. The recipients should get this information clearly presented on a beautiful user interface.


Baloon Telemetry via QO-100
as external data in HS-modem
Ground Station
External data interface UDP: 40135
Hsmodem Transmitter
Transport Medium, i.e. QO100
Hsmodem Receiver
Websocket
WebSocket Port: 40134
Web Browser

To do this, the following things are required:

1. The information received from the balloon is packed by a small program (in 219 byte blocks).
2. This a data packet is sent to HS modem via an UDP network connection. Thanks to the network connection, the QO100 station can stand anywhere and does not have to be set up at the location of the balloon experiment.
3. HS-Modem modulates and sends this data at the selected speed.
4. Every radio amateur receiving this signal with HS modem will get this data
5. the RX HS modem recognizes the data stream and redirects it to a web socket (this is part of the HS modem)
6. In addition to the data stream, an HTML file is also sent to the recipients. As soon as they open the HTML file in a browser, they can view the current information from the balloon experiment, which is automatically updated by the websocket stream.

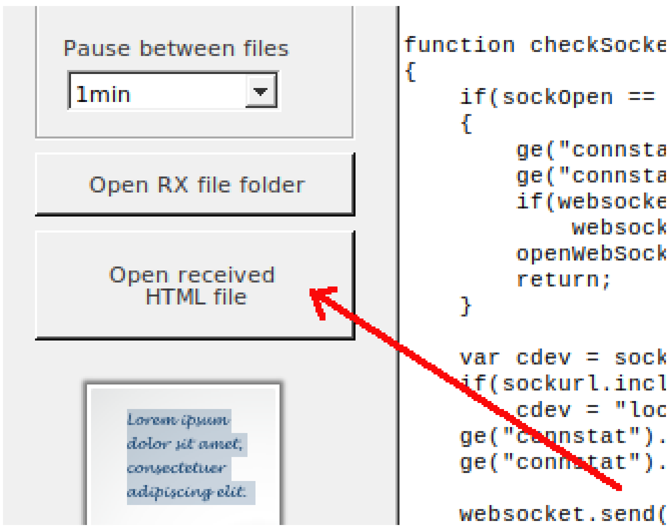
Usage RX - Streaming: Operation from the recipient's point of view

the transmission includes two parts:

1. the HTML web file for displaying the information
2. the data stream with the external information

the HTML file is required to display the data in a web browser. It has nothing to do with the internet. The HTML file is saved locally and called up by the browser, so this also works without an Internet connection.

Usually, the required HTML file is sent with the data stream, not very often, but it is sent every few minutes. HS-Modem recognizes HTML files automatically and saves them in the computer. As soon as you have received an HTML file, a new button will appear in the file window:



```
function checkSocke
{
    if(sockOpen ==
    {
        ge("connsta
        ge("connsta
        if(websocke
            websock
        openWebSock
        return;
    }

    var cdev = sock
    if(sockurl.incl
        cdev = "loc
    ge("connstat").
    ge("connstat").

    websocket.send(
```

Then click on this button to open the HTML file in a browser.

Further operation depends on the HTML application. In the case of the Amsat Multimedia Beacon, see: QO-100 Multimedia Beacon.

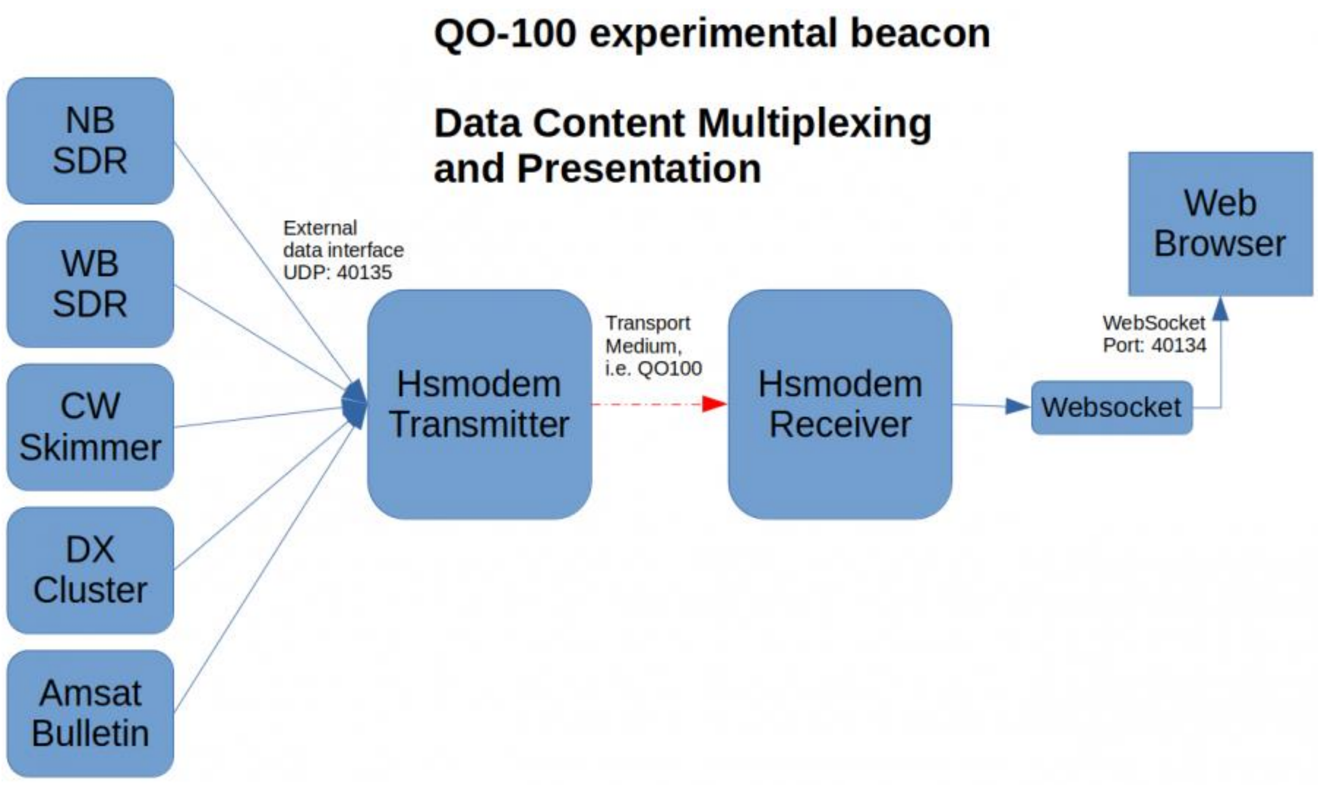
QO-100 Multimedia Beacon

This beacon can be received in experimental mode on 10489.580 MHz, at irregular times. A high-speed modem version 0.8 or newer is required.

The multimedia beacon uses the streaming mode of HS modem to transmit several data streams at the same time. The individual data streams are mixed in a time grid of 250ms, so that a simultaneous impression is created for the recipient.

At the current status 5 streams are transmitted:

1. Narrow band transponder spectrum and waterfall
2. Wide band transponder spectrum and waterfall
3. DX Cluster (all frequencies, selectable)
4. CW skimmer (QO-100)
5. Amsat Bulletin, breaking news


QO-100 experimental beacon
Data Content Multiplexing and Presentation
NB SDR
WB SDR
CW Skimmer
DX Cluster
Amsat Bulletin
External data interface UDP: 40135
Hsmodem Transmitter
Transport Medium, i.e. QO100
Hsmodem Receiver
Websocket
WebSocket Port: 40134
Web Browser

The data streams are fed into the multimedia HS modem beacon at the Amsat-DL ground station and sent to the QO-100. At the receiver, the data streams are separated automatically, and operation is very simple.

Receiving the QO-100 multimedia beacon

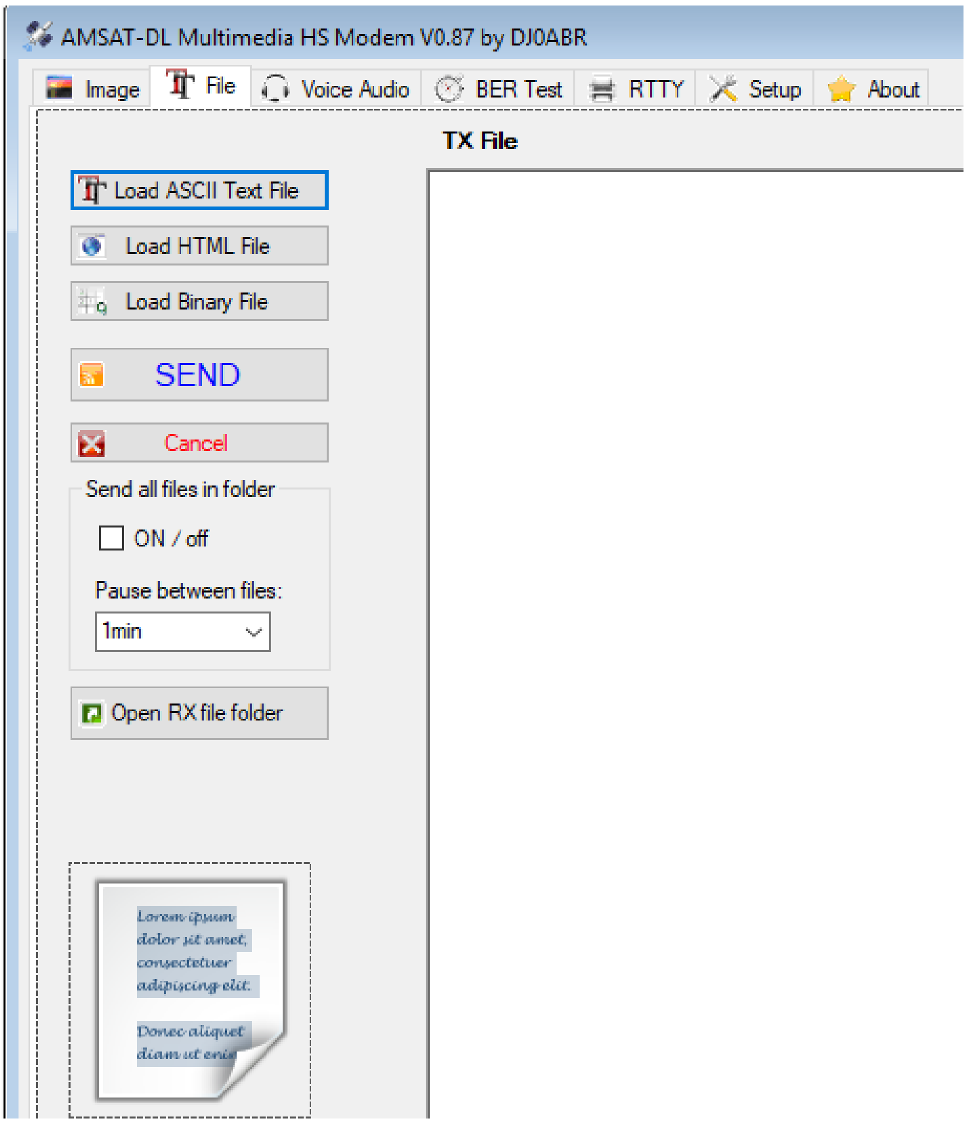
If you want to receive the beacon and see the information, do the following:

1. Tune the receiver with HS modem running to the beacon
2. Set the HS modem to 8APSK - 7200 bps
3. check that your RX filter bandwidth is at maximum (3.2kHz or even more), HSmodem has its own filters and does not need additional filters in the receiver.
4. Fine-tune the frequency while observing the 100Hz subcarrier until the synchronization is established.
5. Open the file QO100info.html and you will see the information transferred.

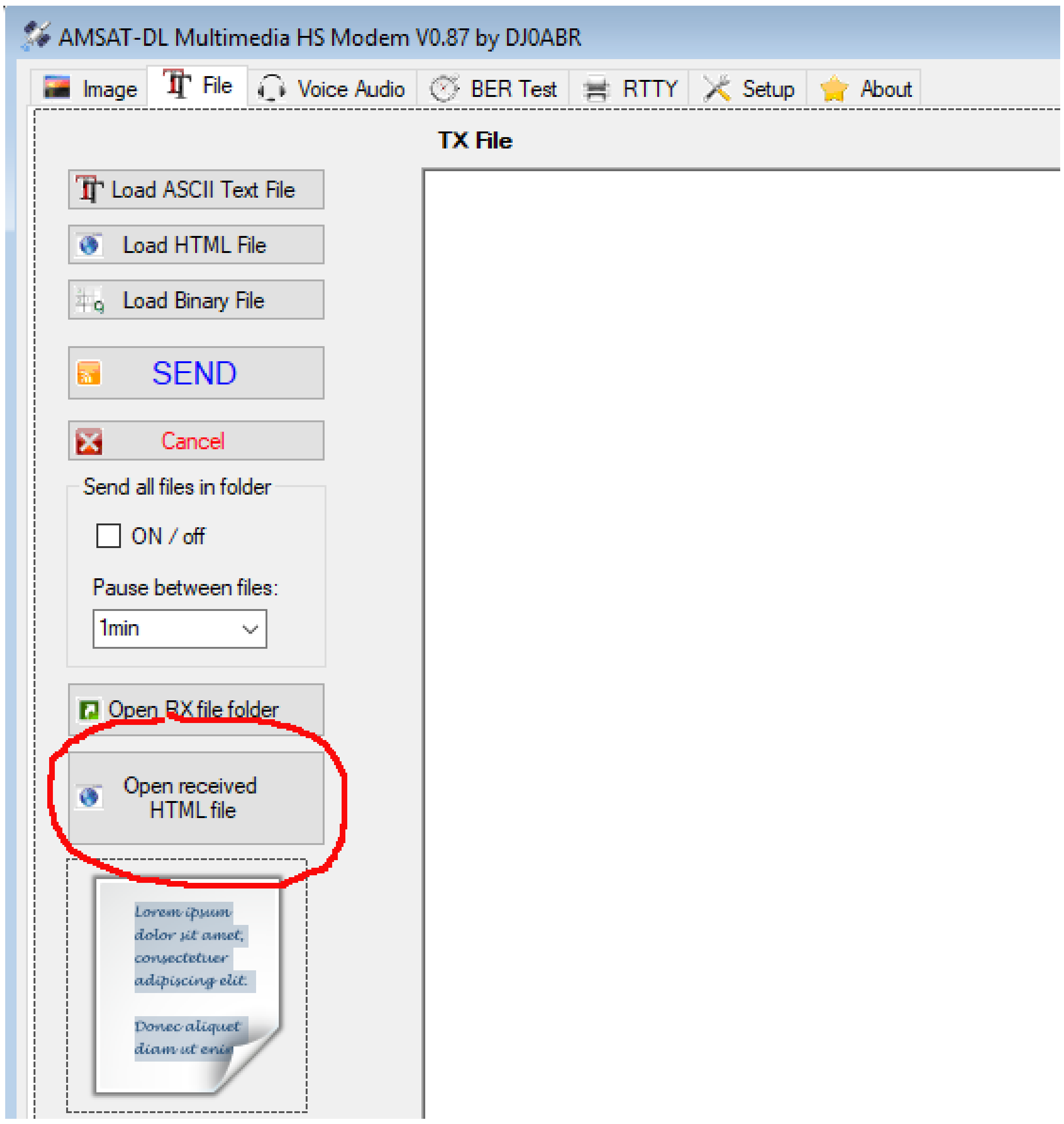
Open the file QO100info.html

this file is transmitted via the beacon every few minutes.

If the file QO100info.html has not yet been received, it looks like this:



As soon as the file QO100info.html has been received, an additional button is displayed:



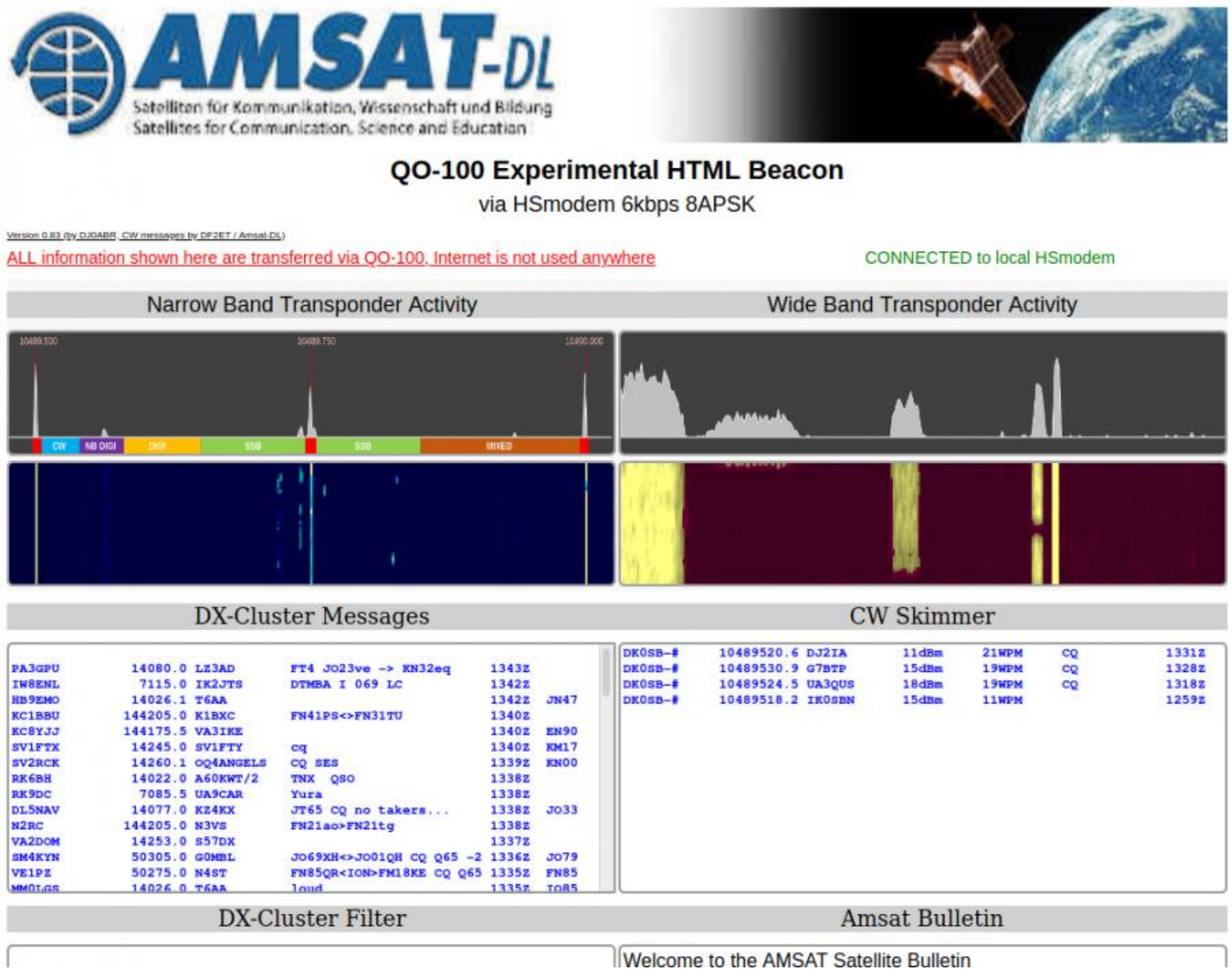
If you click on this marked button, the web browser is opened and the file QO100info.html is displayed.

if the file QO100info.html has already been received

Then it is already saved on the hard drive and you don't have to wait any longer. Press "open RX files". Here you can find QO100info.html and open it with a double click.

Using QO100info.html

This is how the beacon QO100info.html is displayed in the browser:


AMSAT-DL
Satelliten für Kommunikation, Wissenschaft und Bildung
Satellites for Communication, Science and Education

QO-100 Experimental HTML Beacon

via HSmodem 6kbps 8APSK

Version 0.83 (by DJ0ABR, CW messages by DF2ET / Amsat-DL)

ALL information shown here are transferred via QO-100, Internet is not used anywhere

CONNECTED to local HSmodem

Narrow Band Transponder Activity

Wide Band Transponder Activity

10489.500
10489.750
10490.000
CW
NB DIGI
DIGI
SSB
SSB
MIXED

DX-Cluster Messages

```
PA3GPU          14080.0 LZ3AD       FT4 JO23ve -> KN32eq     1343Z
IW8ENL           7115.0 IK2JTS      DTMBA I 069 LC           1342Z
HB9EMO          14026.1 T6AA                                 1342Z  JN47
KC1BBU         144205.0 K1BXC       FN41PS<>FN31TU           1340Z
KC8YJJ         144175.5 VA3IKE                               1340Z  EN90
SV1FTX          14245.0 SV1FTY      cq                       1340Z  KM17
SV2RCK          14260.1 OQ4ANGELS   CQ SES                   1339Z  KN00
RK6BH           14022.0 A60KWT/2    TNX  QSO                 1338Z
RK9DC            7085.5 UA9CAR      Yura                     1338Z
DL5NAV          14077.0 KZ4KX       JT65 CQ no takers...     1338Z  JO33
N2RC           144205.0 N3VS        FN21ao>FN21tg            1338Z
VA2DOM          14253.0 S57DX                                1337Z
SM4KYN          50305.0 G0MBL       JO69XH<>JO01QH CQ Q65 -2 1336Z  JO79
VE1PZ           50275.0 N4ST        FN85QR<ION>FM18KE CQ Q65 1335Z  FN85
MM0LGS          14026.0 T6AA        loud                     1335Z  IO85
```

CW Skimmer

```
DK0SB-#     10489520.6 DJ2IA       11dBm     21WPM     CQ            1331Z
DK0SB-#     10489530.9 G7BTP       15dBm     19WPM     CQ            1328Z
DK0SB-#     10489524.5 UA3QUS      18dBm     19WPM     CQ            1318Z
DK0SB-#     10489518.2 IK0SBN      15dBm     11WPM                   1259Z
```

DX-Cluster Filter

Amsat Bulletin

Welcome to the AMSAT Satellite Bulletin

A few seconds after opening the file in the browser, a connection to the HS-Modem is established and “Connected to local HSModem” appears in green at the top right. If this message does not appear, HS-Modem is not running or may have to be restarted. With a Windows 10 PC you may be asked whether you want to release port 40134, which you also have to do so that the data can be forwarded to the browser.

The beacon can now be viewed, no operation is required, the information is constantly updated. Filters are available for the DX cluster messages that can be set as required.

TRX settings

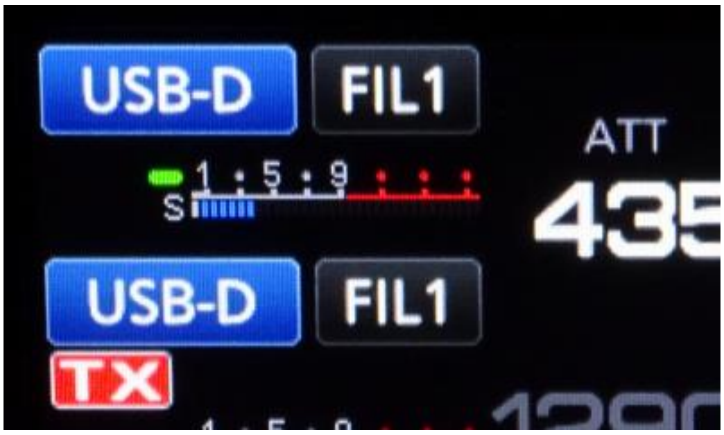
Transmit and receive adjustments using ICOM transceivers:

There are many transceivers, we are using ICOM devices. We do not know how it works with devices (with built-in USB sound cards) from other manufacturers and we are grateful for reviews.

The following transceivers were tested: IC-9700, IC-7100 and (Rx only) IC-7300 and IC-7610. The settings shown below apply to these four transceivers. IC-910 and IC-9100 also work, only the names of the menu items are slightly different on these devices. The IC-705 has not been tested, but it can be assumed that the HS modem will also work with it.

USB-D:

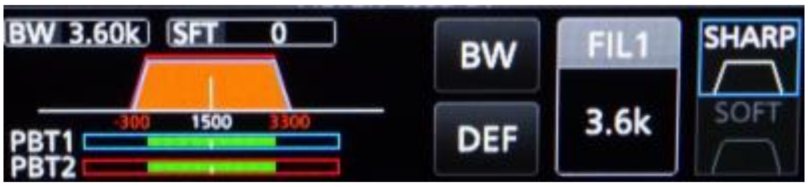
USB is usually set for analogue voice operation. For data operation, you must switch to USB-D. To do so, press [USB], and then enable DATA. This must be performed for both the transmitting and the receiving side. Then, you can view your own images which you are transmitting for checking purposes..



If you want to use analogue voice operation again, you have to switch back to normal USB..

Filter:

The reception filter is set to FIL1. Then check the bandwidth: press FIL1 for 2 s until the filter menu is displayed. The filter curve is set to SHARP. Then tap on BW and set to 3.6 k with the big knob.



Frequency:

The satellite band plan reserves the frequency range 10489.580 to 10489.650 MHz for “digital operation with a bandwidth of up to 2.7 kHz”. Alternatively, the “Mixed/Special” range above 10489.850 MHz can be used.

The display in the transceiver must of course always be offset according the transmit converter. Usually the decimal places below 1MHz match.

Level Settings:

Transmission level, output power:

The output power is influenced by both the sound card output level and the transceiver power setting. It is important that the sound card is not overdriven, as any overdriving will severely disturb the QPSK/8APSK signal.

Below please find the setting using the example of ICOM transceivers:

MENU button - SET - CONNECTORS - MOD INPUT:

- `Set everything to 50%`
- `DATA OFF MOD: MIC/ACC`
- `DATA MOD: USB`

With the setting “USB MOD LEVEL” you can adjust the level of the transmitted signal, about 40-50 % should be optimal.

Reception level:

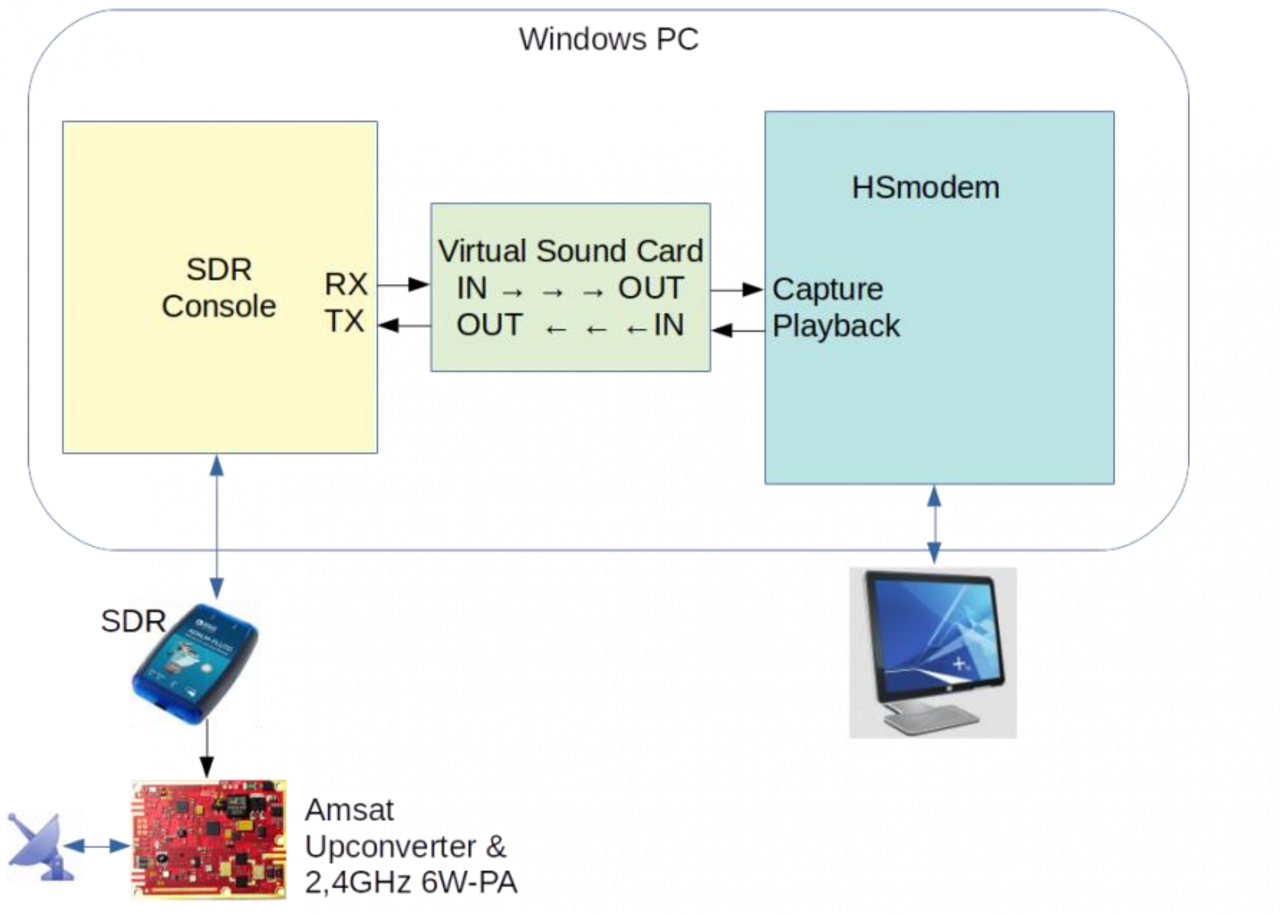
The reception level must be set so that when a signal is received, the spectrum is within the green range.

MENU button: SET – CONNECTORS - USB AF/IF OUTPUT:

- `OUTPUT SELECT: AF`
- `AF OUTPUT LEVEL: approx. 50%, depending on spectrum display`

SDR-Console

Operation with SDR console in the following typical configuration:



Hardware:

- Windows 10 PC
- SDR Adalm-Pluto (connected via Ethernet-USB adapter)
- TX: Amsat upconverter with 6W PA to a POTY antenna
- RX: LNB output directly to the Pluto RX input (via DC feed)

Software:

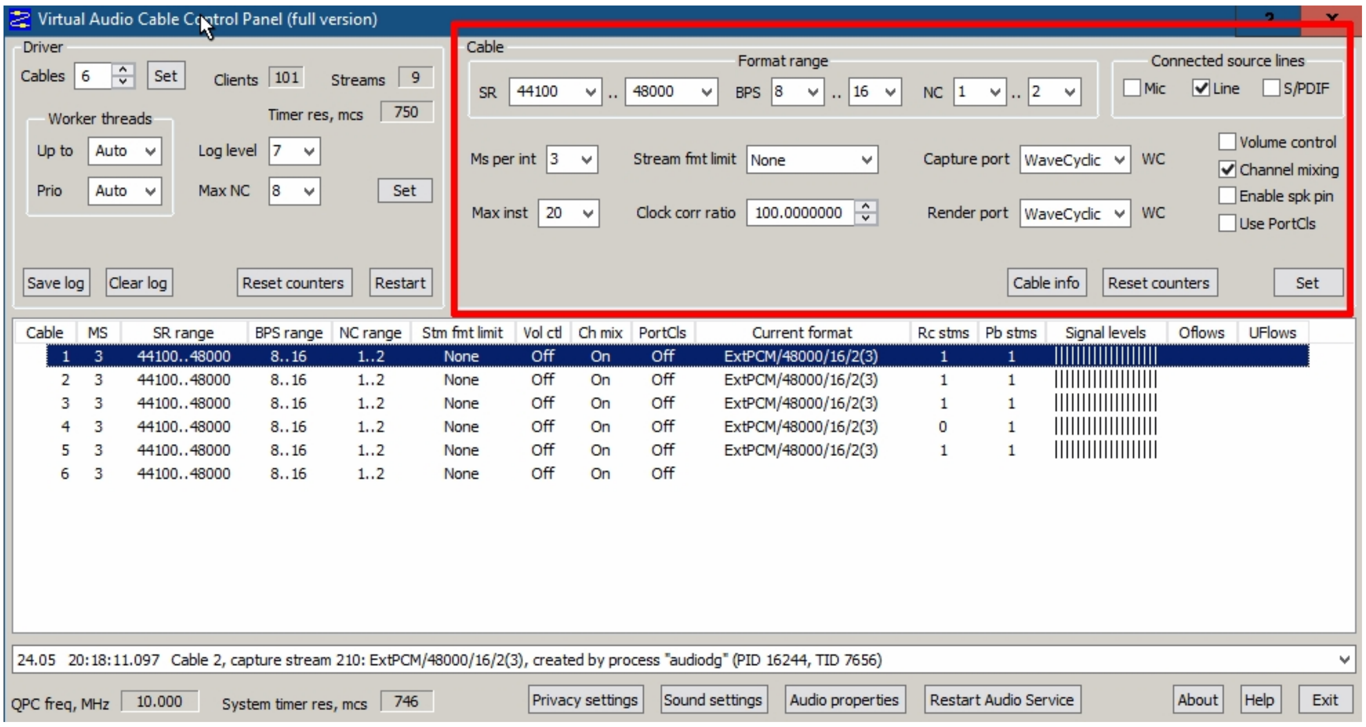
- SDR console (these tests were made with V3.0.23)
- HSmodem
- virtual audio cable from vac.muzychenko.net

for the individual settings, see the corresponding following chapters.

Audio connection SDR console <-> HSmodem

For this description the virtual audio cable (VAC) from vac.muzychenko.net was used.

After installing the VAC cable you can open the Control Panel (Windows menu: “Virtual Audio Cable” - “Control Panel”)



Virtual Audio Cable Control Panel (full version)

Driver
Cables 6 Set
Clients 101 Streams 9
Timer res, mcs 750
Worker threads: Up to Auto, Prio Auto
Log level 7
Max NC 8 Set
Save log | Clear log | Reset counters | Restart

Cable
Format range: SR 44100 .. 48000, BPS 8 .. 16, NC 1 .. 2
Connected source lines: [ ] Mic [✓ Line] [ ] S/PDIF
Ms per int 3
Stream fmt limit None
Capture port WaveCyclic WC
[ ] Volume control
[✓ Channel mixing]
[ ] Enable spk pin
[ ] Use PortCls
Max inst 20
Clock corr ratio 100.0000000
Render port WaveCyclic WC
Cable info | Reset counters | Set

Cable	MS	SR range	BPS range	NC range	Stm fmt limit	Vol ctl	Ch mix	PortCls	Current format	Rc stms	Pb stms	Signal levels	Oflows	UFlows
1	3	44100..48000	8..16	1..2	None	Off	On	Off	ExtPCM/48000/16/2(3)	1	1	||||||||||||||||		
2	3	44100..48000	8..16	1..2	None	Off	On	Off	ExtPCM/48000/16/2(3)	1	1	||||||||||||||||		
3	3	44100..48000	8..16	1..2	None	Off	On	Off	ExtPCM/48000/16/2(3)	1	1	||||||||||||||||		
4	3	44100..48000	8..16	1..2	None	Off	On	Off	ExtPCM/48000/16/2(3)	0	1	||||||||||||||||		
5	3	44100..48000	8..16	1..2	None	Off	On	Off	ExtPCM/48000/16/2(3)	1	1	||||||||||||||||		
6	3	44100..48000	8..16	1..2	None	Off	On	Off						

```
24.05  20:18:11.097  Cable 2, capture stream 210: ExtPCM/48000/16/2(3), created by process "audiodg" (PID 16244, TID 7656)
```

QPC freq, MHz 10.000 System timer res, mcs 746
Privacy settings | Sound settings | Audio properties | Restart Audio Service | About | Help | Exit

You need at least two virtual cables, one for reception, the other for transmission.

In the example above 6 cables are used so that multiple receiver outputs of SDR console can be routed to different modems simultaneous.

For your first tests you may simply use only 2 cables.

The tested settings can be seen in the picture, you can simply accept them.

Cables 1 and 2 can be seen in the “Cable” column on the left. Under Windows they appear with the designation **Line-1** and **Line-2** .

In this example we use the Line-1 cable for transmission and the Line-2 cable for reception.

Continue in the chapter ***SDR console settings***.

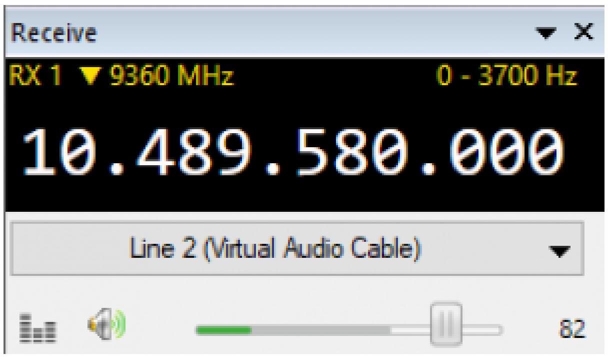
Make sure you are using the settings for Capture port and Render port as shown in the screenshot above otherwise you will get severe dropouts especially in your transmit audio.

Preparing the SDR console

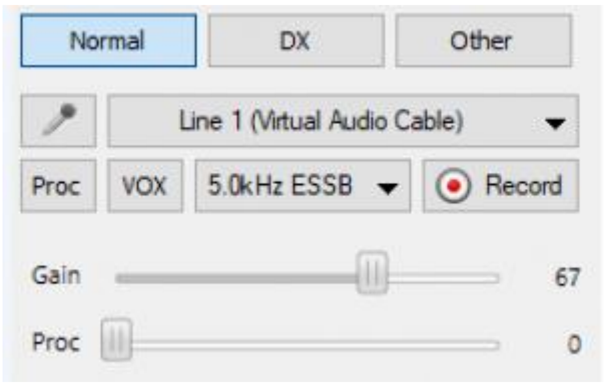
The SDR console is a comprehensive program with many settings. Hardly anyone knows them all by heart. Therefore all necessary settings are described here. If you have set everything correctly, a practically error-free operation with HSmodem will be possible. Apart from the filter settings, everything can be left as it is for normal SSB operation.

Audio and Virtual Audio Cable:

On the **receiving side** we select the audio cable **Line-2** :


Receive
RX 1 ▼ 9360 MHz 0 - 3700 Hz
10.489.580.000
Line 2 (Virtual Audio Cable)
82

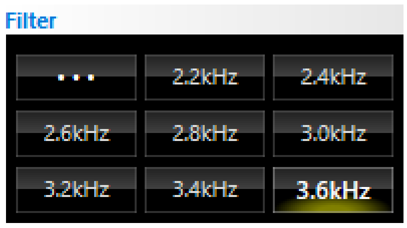
On the **transmission side** we select the audio cable **Line-1** :


Normal DX Other
Line 1 (Virtual Audio Cable)
Proc VOX 5.0kHz ESSB Record
Gain 67
Proc 0

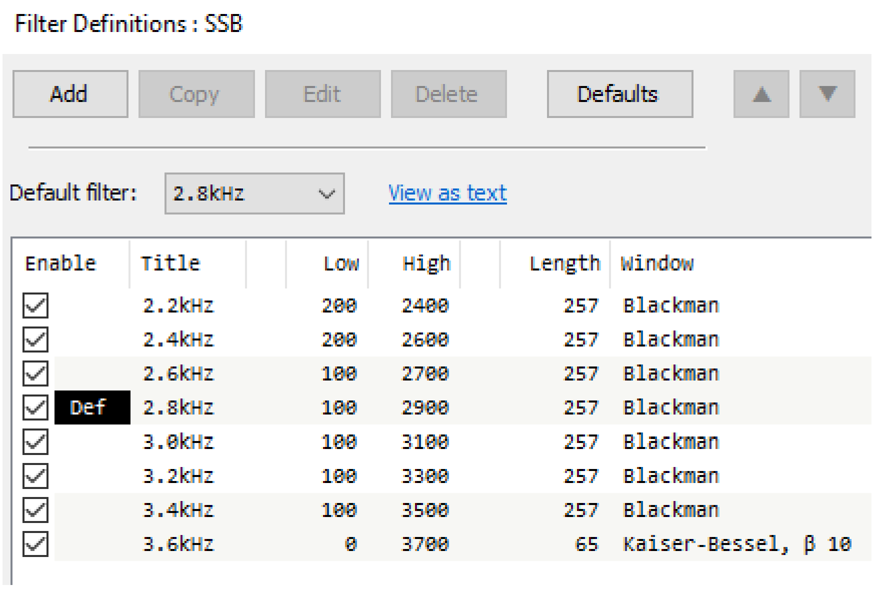
Filter settings for reception:

the minimum filter width must be 3 kHz, but HSmodem has its own filters. It is therefore best to choose the reception filter as wide as possible:

To set, click on the three points in the filter settings



The lowest filter (3.6 kHz) is used. **Here it is important to set the lower limit frequency to 0 Hz**, because HSmodem uses the entire range.


Filter Definitions : SSB

Add | Copy | Edit | Delete | Defaults

Default filter: 2.8kHz

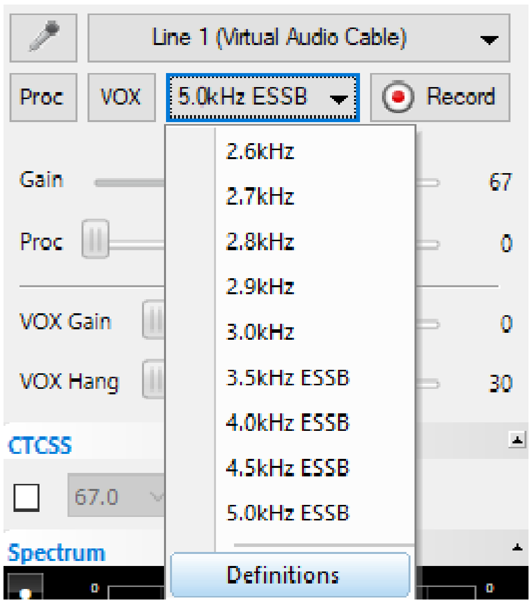
View as text

Enable		Title	Low	High	Length	Window
✓		2.2kHz	200	2400	257	Blackman
✓		2.4kHz	200	2600	257	Blackman
✓		2.6kHz	100	2700	257	Blackman
✓	Def	2.8kHz	100	2900	257	Blackman
✓		3.0kHz	100	3100	257	Blackman
✓		3.2kHz	100	3300	257	Blackman
✓		3.4kHz	100	3500	257	Blackman
✓		3.6kHz	0	3700	65	Kaiser-Bessel, β 10

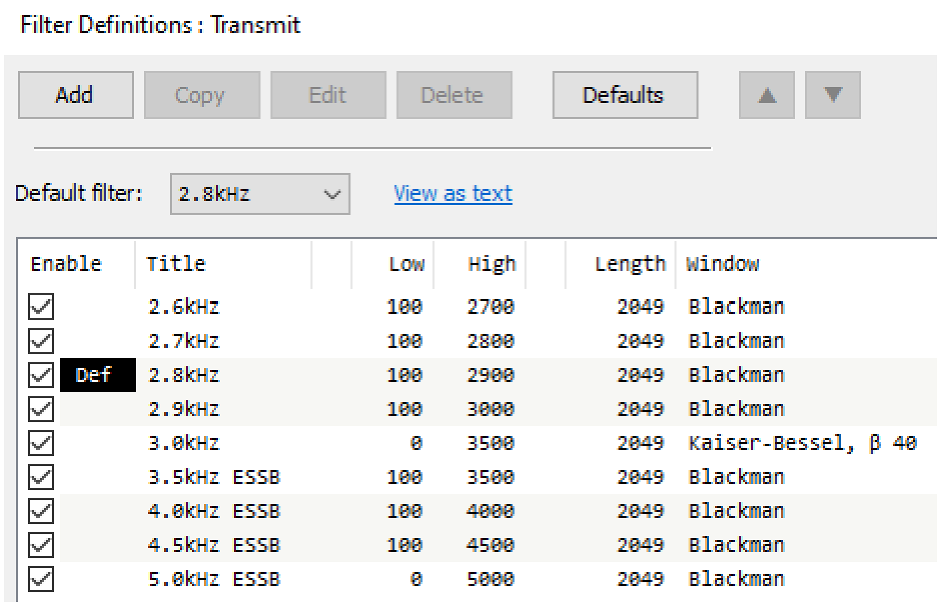
the filter is finally selected (highlighted in yellow) and is thus in operation.

Filter settings for sending:

HSmodem also has its own filter here, so we can choose the widest one, the 5kHz ESSB filter. First we have to configure it:



Here, too, the lower limit frequency “low” must be set to 0.



Filter Definitions : Transmit

Default filter: 2.8kHz

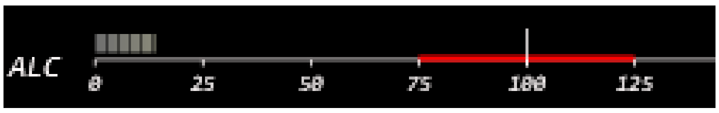
Enable		Title	Low	High	Length	Window
✓		2.6kHz	100	2700	2049	Blackman
✓		2.7kHz	100	2800	2049	Blackman
✓	Def	2.8kHz	100	2900	2049	Blackman
✓		2.9kHz	100	3000	2049	Blackman
✓		3.0kHz	0	3500	2049	Kaiser-Bessel, β 40
✓		3.5kHz ESSB	100	3500	2049	Blackman
✓		4.0kHz ESSB	100	4000	2049	Blackman
✓		4.5kHz ESSB	100	4500	2049	Blackman
✓		5.0kHz ESSB	0	5000	2049	Blackman

Volume settings for reception:

The reception volume must be set so that the signal is not overloaded. In our tests, a setting around 80% has proven itself.

Volume settings for sending:

also here you have to be careful not to oversteer. As an aid, the SDR console has various "Meter" displays:



The picture shows an ALC of approx. 25%, it shouldn't be much more. Values up to 50% are not a problem, above 100 overdrive will occur and impair the signal. Please adjust with the gain control. The output power is set with the "Drive" control, not with "Gain".

Turn off all audio processors / audio improvements

that is a very important point. If the audio signal is caused by any noise blankers, equalizers or the like. is influenced, the phase modulation is destroyed and no operation is possible.

Here is a checklist, all points must be checked:

at the sender:

- Normal (not DX or Other)
- Turn off proc
- Turn off CTCSS

at the recipient:

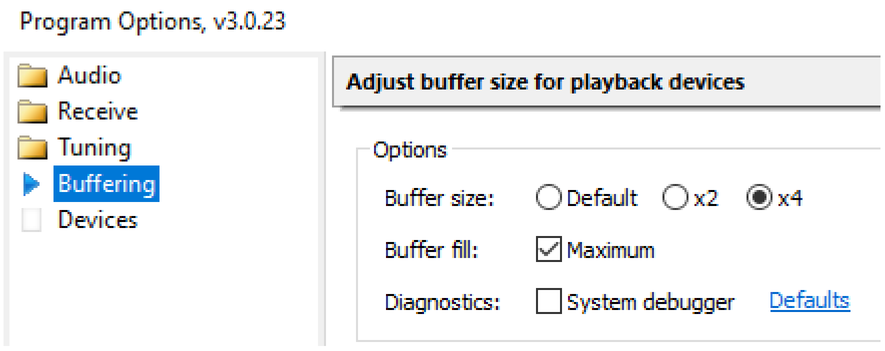
- RX filter: 100%
- CW: off
- Noise blanker: off
- Noise Reduction: off
- Notch: off
- Squelch: off

In addition, the equalizer must be switched off, this can be found in the menu **Receive** , button **Options** and there under **Audio Equalizer** . **Enable** must remain switched off here.

Receive buffer

These buffers are important in order to avoid short interruptions in audio transmission, which would lead to data errors.

Menu **Receive** , button **Options** and there for **Buffering** : Buffersize x4 and buffer fill: set maximum.


Program Options, v3.0.23

Audio
Receive
Tuning
Buffering
Devices

Adjust buffer size for playback devices

Options

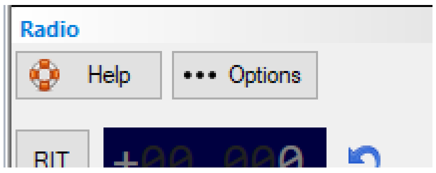
Buffer size: Default x2 [✓ x4]

Buffer fill: [✓ Maximum]

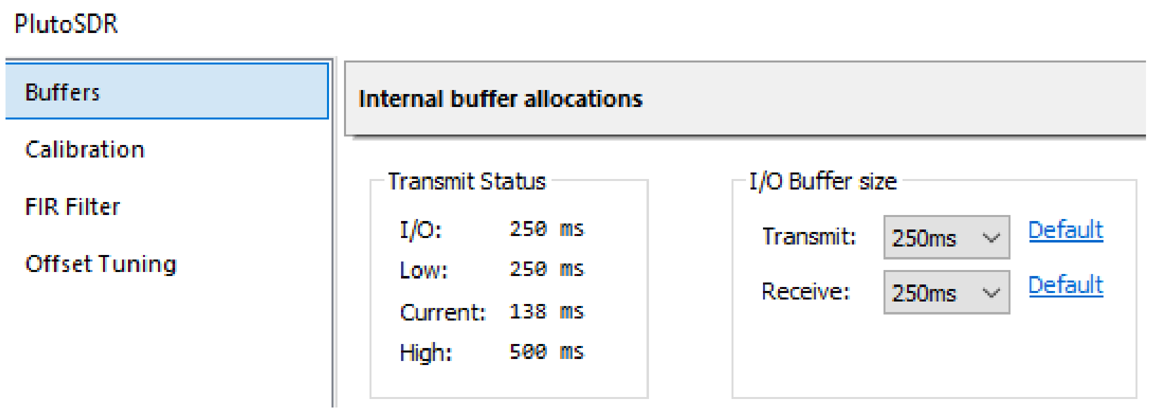
Diagnostics: [ ] System debugger

Defaults

In addition, the Pluto has its own buffer, which can be selected via *** Options on the receiving side:



Here you set transmit and receive to 250ms.


PlutoSDR

Internal buffer allocations

Transmit Status

```
I/O:      250 ms
Low:      250 ms
Current:  138 ms
High:     500 ms
```

I/O Buffer size

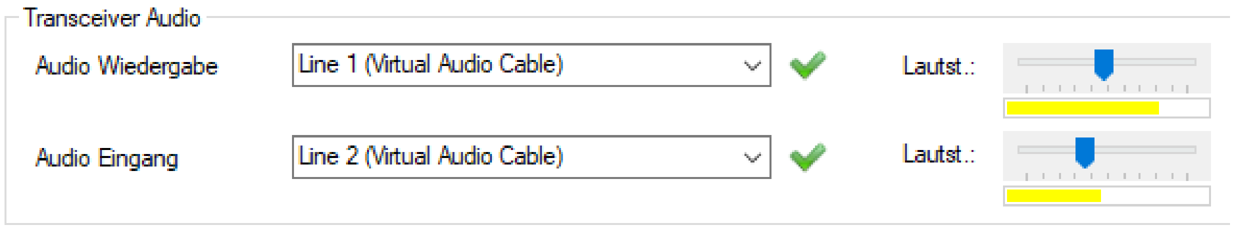
Transmit: 250ms Default

Receive: 250ms Default

Settings in HSmodem for SDR console

the only settings here are the selection of the virtual audio cables and the level settings:

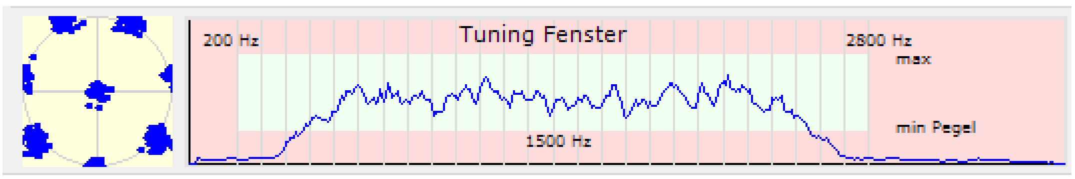
The audio cable Line-2 was selected on the receiving side of the SDR console. This reception signal must therefore be selected as the audio input for the HSmodem. The output on Line-1 corresponds to the transmitter.



Transceiver Audio

Audio Wiedergabe	Line 1 (Virtual Audio Cable)	✓	Lautst.:
Audio Eingang	Line 2 (Virtual Audio Cable)	✓	Lautst.:

The volume should be in the middle. Under no circumstances should the yellow bar turn red, as the signal would then be overdriven or underdriven. The input level is set so that the spectrum is roughly in the green area.



This picture shows the active HSmodem if all settings are correct. With this rather sharp constellation diagram (8APSK 6000 bit / s) data can be transmitted without errors. With SDR consoles and Pluto, you should not be satisfied with less.

Practical tips

if you have made all the settings of the SDR console, it should work fine.

However, there is one thing you should know about Windows: the level settings.

Both the output volume and the input level can be influenced at several places. This is important because a setting that is too loud will overload the signal and thus interfere.

Transmission level

is influenced in 4 places:

1. HSmodem: audio playback
2. Windows: the volume controls built into Windows
3. SDR console: Gain (on the transmitter side)
4. SDR console: Drive

this is the recommended procedure:

- In HSmodem, the controller for audio playback in the middle position, the bar below it will then be yellow
- Turn up the gain control in the SDR console until the ALC meter is around 25%
- Set the desired output power in the SDR console with the drive controller

only if you don't achieve your goal (e.g. too quietly or too loudly) you change the volume settings in Windows.

Reception level

is influenced in 3 places:

1. SDR console: set the volume control on the receiving side to approx. 80.
2. Windows: leave the volume controls built into Windows as they are.
3. HSmodem: Set the audio input so that the spectrum is in the green area (of course only if a signal is received) and the bar under the controller remains yellow.

only if you don't achieve your goal (e.g. too quietly or too loudly) you change the volume settings in Windows.

Frequency setting:

The frequency must not be changed during reception. The SDR console makes short interruptions when the frequency changes, which lead to data loss.

Operation using the Adalm PLUTO

the Pluto can be connected via USB or via USB-Ethernet adapter.

The following note is important when operating with USB-Ethernet adapters:

As described, the Pluto buffer sizes of the SDR console must be set to the maximum (250ms).

However, some USB Ethernet adapters have latency times of >500 ms, and the SDR console buffer can no longer cover this long time and dropouts occur.

How do you recognize a USB-Ethernet adapter with too high latency times:

You can hear the transmission signal back acoustically. If you run a BER test, you can hear the typical digital, noise-like modulation. If this interrupts briefly again and again, you have dropouts. These usually last 0.2 to 1 second, so they can be heard very clearly. The frequency of the dropouts is determined by the quality of the USB-Ethernet adapter and is often in the range of approx. 10 to 20 seconds.

Which USB Ethernet adapters work:

there are many adapters that are of course not available for testing here. The adapter with the lowest latency was the following:

UGREEN LAN Adapter USB 2.0 Network USB to RJ45 Ethernet Adapter 10 / 100Mbps (note: no USB 3.0 and no gigabit!)

With this adapter, error-free operation can be carried out on our reference PC (Intel i7) over very long periods of time.

Which USB Ethernet adapters don't work:

Bad experiences in the interaction between the SDR console and Pluto were made with USB 3.0 gigabit adapters, such as:

Techrise high-speed USB 3.0 to RJ45 Gigabit Ethernet LAN network adapter, supports 10/100/1000 Mbit/s

possibly this is because the Pluto supports USB 2.0 and therefore the 3.0 adapters switch back to 2.0 mode where they do not work optimally. But that is only an unconfirmed assessment.

Hint for operation:

Use the USB interface of your PC or use a USB2.0 adapter with max. 100Mbit/s

The Pluto's sampling rate should be set to the minimum, which is 550kHz in der SDR Console start dialog.

Technical Specifications for Developers

Audio Format

Audio samples are generated by a complex modulator in the base band. These samples are upmixed to 1.5 kHz (mid of an SSB channel) and sent to the sound card.

For compatibility between different programs we need to know the physical format, the modulation:

Modulation

The modulation is done with the liquid-DSP's modulator/demodulator. Formats used:

1. LIQUID_MODEM_BPSK
2. LIQUID_MODEM_QPSK
3. LIQUID_MODEM_8APSK

all in “normal” mode, NOT differential

to get a fully compatible program it is highly recommended to use the interpolator/filter settings as in liquid_if.cpp because these settings are optimized for the 2.7kHz SSB channel.

The complex output of the modulator is upmixed by a liquid-NCO to 1500 Hz.

Constellation

the default liquid-DSP constellation is used, no changes.

Frame & Rotation detection

Frame length:

the frame length is 255 Bytes (scrambled) as shown in the next chapters. Each frame has a 3 byte header, so in summary there are 258 bytes to transmit.

The number of 258 Bytes is important because it can be divided by 2 (129) and also by 3 (86). Therefore we can convert it to QPSK and 8APSK symbols without any fractional remaining bits which makes calculations more efficient.
The same is valid for the 3 Byte header which also is dividable by 2 or 3.

Header detection

in non-differential PSK the phase is unknown, so the symbols can be rotated by 4 (QPSK) or by 8 (8APSK). For detection of the rotation value the “unique word” method is used: The header is tested 4 times for each possible QPSK rotation. If the header is detected then the rotation value is found and the whole frame is back-rotated by this value resulting in the original stream (8 values for 8APSK).

The header has the values 0x53 - 0xe1 - 0xa6 which corresponds to these symbols:

Byte: 0x53 0xe1 0xa6
bits: 01010011 11100001 10100110
BPSK:
syms: 01010011 11100001 10100110
QPSK:
syms: 1 1 0 3 3 2 0 1 2 2 1 2
8PSK:
syms: 2 4 7 6 0 6 4 6

these symbols are tested for 2 rotations (BPSK), 4 rotations (QPSK) or 8 rotations (8APSK). Functions to calculate the rotations and to back-rotate a complete frame can be found in constellation.cpp

False Detections

it is possible, and it will happen quite often. A header is detected but it is not a real header. This is totally irrelevant because the CRC16 will cancel the mistakenly detected frame. The only downside is that the CPU load is a bit higher because an invalid frame was checked.

Therefore we can allow some inaccuracy in the header. If only a few bits are wrong we still accept it as a header. This improves detection when weak signals are received.

See frame_packer.cpp:
#define MAXHEADERRS

Under Windows we allow 5 wrong header bits because Windows PCs usually have powerful CPUs. Under Linux only 2 wrong bits are allowed to keep the CPU load low on Rapberry PI and other SBCs.

Coding and Decoding of a frame

a frame consists of 255 byte data and 3 byte header. The header is a fixed 3-byte value and will not be processed further. The sequence of header processing is described in “Frame Format”. This document show the details of the various processing stages.

If you want to program a compatible modem, just copy these functions as-is into your application, initialize them at program start and use them.

Scrambler

Scrambling is used to prevent long sequences of same values, which would make synchronization in the receiver impossible.

scrambler.cpp uses a table with pseudo-random values and X-ORs the complete frame (except the header) with these values.

The receiver unscrambles the frame with the same function to get the original data.

FEC

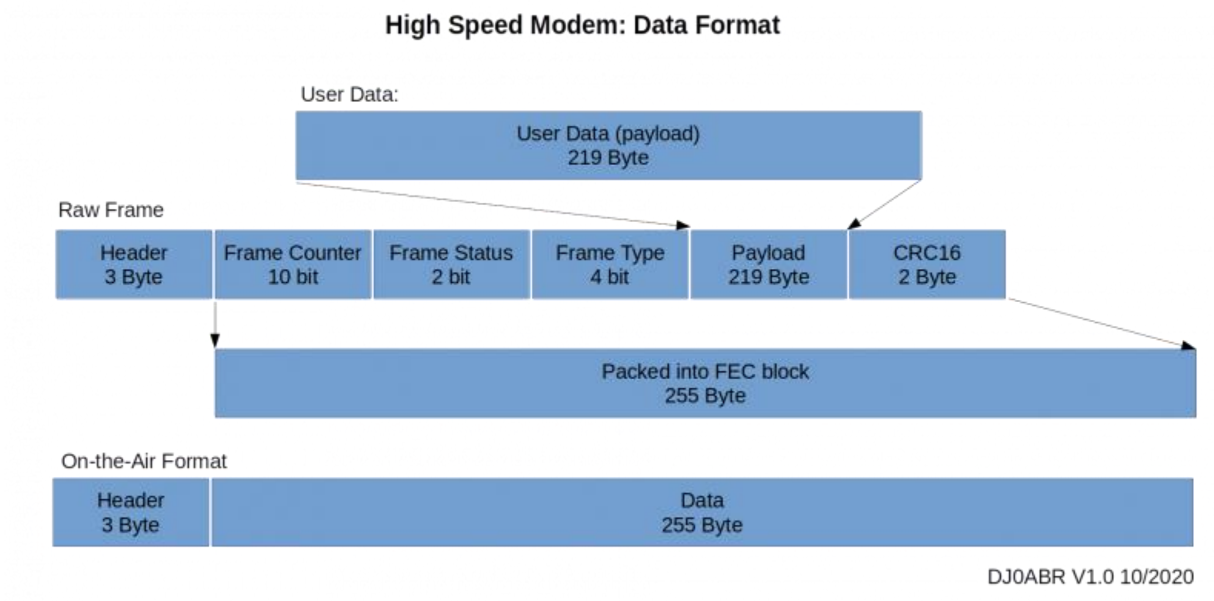
many different FEC/Reed Solomon algorithms have been tested at a signal SNR of +15dB above noise.

The best results delivered the “Shifra” reed solomon code. This code is free and open source for non-commercial projects like HSmodem. A wrapper for HSmodem is in fec.cpp. It has functions to initialize, generate and decode the FEC for one complete frame.

CRC16

A standard CRC16 algorithm is used with a generator polynom of 0x8408. The code is simple and located in crc16.cpp

Frame Format



Data Processing Sequence - Transmitter:

(see file: frame_packer.cpp)

- The GUI (user interface or other application that needs to send data) delivers data in blocks of 219 bytes (payload). If a larger file has to be transmitted then the application has to split it into 219 byte blocks.

- this 219 byte payload is extended with additional information with 2 Byte (16 bit) length:

Frame Counter: a 10 bit counter (0..1023) which starts with value 0 at the beginning of a file transmission and then automatically increases by one for each sent frame.

Frame Status: a two bit status information:
0 … first frame of a larger file
1 … next frame
2 … last frame of a larger file
3 … first and last frame of a short file which takes only one frame.

Frame Type: specifies the type of the file, which has no meaning for the modem itself, but is important for the application:
1 … BER Test
2 … Image
3 … Ascii File
4 … HTML File
5 … Binary File

6 … Audio (i.e. Codec2)
7 … User Info (Callsign, Locator…)

- these 2 Bytes are added in front of the payload. Now we get 2 byte + 219 byte = 221 byte.

- in the next step the CRC16 is calculated over all 221 bytes. The two CRC16 bytes are added at the end, which results in a length of 223 byte

- these 223 bytes are fed into the Shifra-FEC function which generates 32 bytes of FEC information

- the 32 byte FEC is added at the end. Now we got: 223 byte + 32 byte = 255 byte

- finally the fixed 3 byte header is added and we got the complete frame of 258 byte ready to be sent

- convert bits to symbols according to the selected mode (BPSK, QPSK, 8APSK). See functions in constellation.cpp

- send these symbols to the modulator

- the modulator generates the audio samples

- send samples to sound card

Data Processing Sequence - Receiver:

receiving is almost the same procedure as transmitting, in reverse order:

- receive audio samples from the sound card

- demodulate the samples and generate symbols

- fill a received symbol in a FIFO which can hold a complete frame

- try to find the header-symbols at the beginning of the FIFO. Do this for each possible rotation.

- when a valid header is detected, check rotation, if required back-rotate the complete FIFO

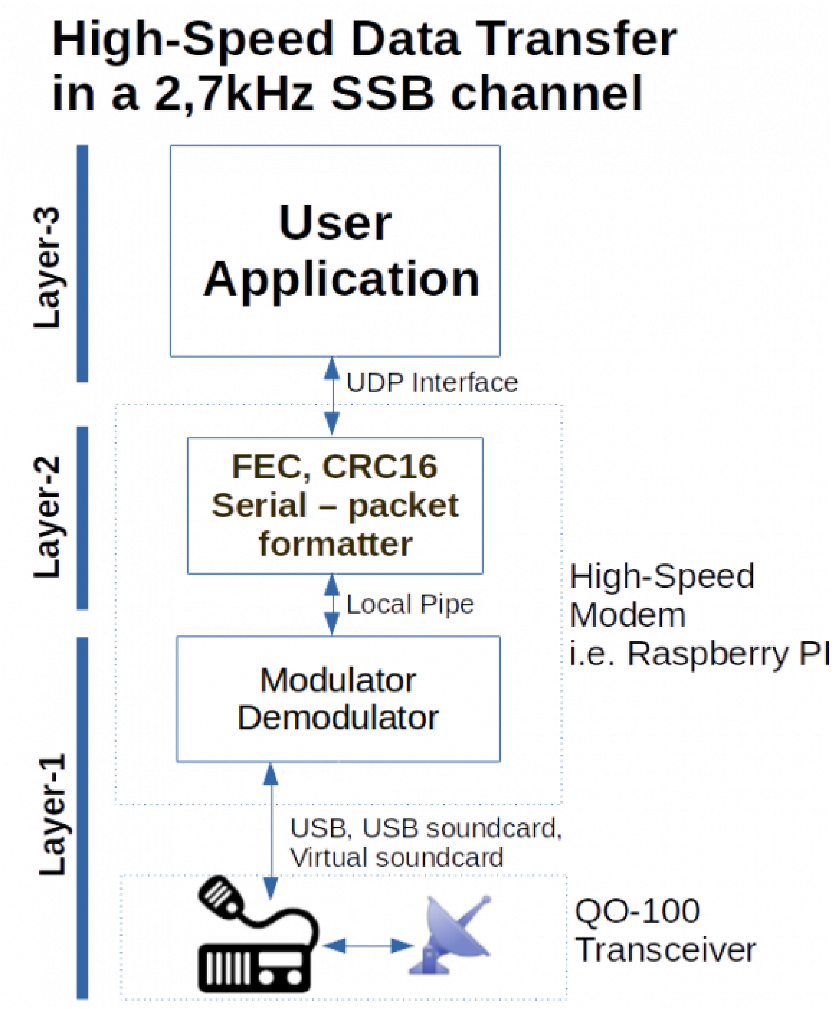
- convert symbols to bits

- de-scramble it (length: 255 byte)

- run the FEC, cancel the frame in case of FEC error. Remaining length: 255-32 = 223 bytes.

- run the CRC16 check, cancel the frame in case of CRC error. Remaining length: 223-2 = 221 bytes.
- read the 2 byte frame status/counter. Remaining length: 221-2 = 219 bytes.
- send the 219 byte payload to the application

Modem <-> App Interface



The AMSAT-DL High Speed Modem is a combination of two programs:

1. the modem functions are handled by hsmodem.exe (Linux: hsmodem). This is a pure console program without GUI. It has two interfaces: an UDP interface to the application and an audio driver to the sound card.
2. the Application (GUI) interacts with the user and exchanges data with the modem over an UDP link. The default Amsat applications is oscardata.exe

This makes it easy to run hsmodem on one computer and the application (oscardata.exe) on another, communication via network.

This document is addressed to programmers who want to use hsmodem.exe but make their own application to transfer any possible data.

The picture shows these functional blocks. hsmodem.exe covers the modulator/demodulator, the frame packing and data security. On the top an UDP interface handles the complete communication with the User application. This enables programmers to build their own application and use the modem as it is. The Amsat default application oscardata.exe is written in C# and therefore runs on Windows as well as on Linux (using mono).

UDP ports:

App to modem, broadcast messages only: 40131
App to modem, data and configuration: 40132
Modem to app, all data: 40133

IP address:

before the app can send data to the modem it has to know the modem's IP address. This address can be discovered automatically with the following procedure:

App: send a broadcast UDP message (addressed to 255.255.255.255) with a specific content (see below)
Modem: response to this UDP message
App: read the modem-IP from the response and address all further messages directly to the modem

Broadcast Message App->Modem:

this message contains some configuration data and should be sent every couple of seconds to the modem. Format of the broadcast message:

- Byte 0 … 0x3c
- Byte 1 … initial Audio volume Modem→Transceiver (0..100 %). *Attention! this is only the inital value after program start. To set the volume during program run use the message No.21,22,24 and 25 as described in GUI data transfer chapter)*
- Byte 2 … initial Audio volume Transceiver→Modem (0..100 %)
- Byte 3 … Timespan for voice announcements (make announcement every n transmissions, default 4)
- Byte 4 … initial Audio volume Modem→Loudspeaker (0..100 %), used for Codec-2 or Opus Codec
- Byte 5 … initial Audio volume Microphone→Modem (0..100 %), used for Codec-2 or Opus Codec
- Byte 6 … Number of retransmission of each frame (default: 1). Every single frame can be retransmitted by the modem for n times. This makes transmission time longer but increases the chance to have a good transmission. On QO-100 usually not used and should have value:1
- Byte 7 … 0 or 1: send a prerecorded PCM audio file before each announcement. This PCM file must be located in …/oscardata/intro/ and has the name intro.pcm
- Byte 8 … 0 or 1: RTTY autosync. If 1 then (in RTTY mode only) the receiver will find the RTTY signal within the passband automatically.
- Byte 9 … Transmission Speed Selection

```
0: 1200 BPSK BW: 1300 Hz
1: 2400 BPSK BW: 2500 Hz
2: 3000 QPSK BW: 1700 Hz
3: 4000 QPSK BW: 2400 Hz
4: 4410 QPSK BW: 2500 Hz (QO-100 Standard)
5: 4800 QPSK BW: 2700 Hz
6: 5500 8APSK BW: 2300 Hz
```

```
  7: 6000 8APSK BW: 2500 Hz (QO-100 Transceiver)
  8: 6600 8APSK BW: 2600 Hz
  9: 7200 8APSK BW: 2700 Hz (QO-100 SDR)
 10: 45.45 Baud RTTY
```

- Byte 10-19 … unused
- Byte 20 - 119 … name of the selected audio playback device (modem→transceiver). May be left empty if no device is selected (i.e. after the program was started for the very first time)
- Byte 120 - 219 … name of the selected audio capture device (transceiver→modem). May be left empty if no device is selected (i.e. after the program was started for the very first time)
- Byte 220 - 239 … my callsign
- Byte 240 - 249 … my QTH locator
- Byte 250 - 269 … my name (callsign QTHloc and name are automatically transmitted by the modem at the beginning of a new file transmission)

Modem's response to the App's broadcast message:

the modem sends an UDP message to the IP of the application. If the modem receives broadcast messages from multiple applications then it prefers the local one (the app running on the same computer). This is the message format of the response to the App's broadcast message:

- Byte 0 … 0x03
- Byte 1 … 0 or 1. Status of the capture device connected to the transceiver. Can be used to show a red or green marker in the GUI
- Byte 2 … 0 or 1. Status of the playback device connected to the transceiver.
- Byte 3 … 0 or 1. Status of the capture device connected to the microphone.
- Byte 4 … 0 or 1. Status of the capture device connected to the loudspeaker.
- Byte 5 - end of frame … List of all sound cards available in the modem computer. This is an ASCII text string in the following format:

Name of a playback device - 1 followed by delimiter '~'
Name of a playback device - 2 followed by delimiter '~'
Name of a playback device - n followed by delimiter '~'
delimiter '^'
Name of a capture device - 1 followed by delimiter '~'
Name of a capture device - 2 followed by delimiter '~'
Name of a capture device - n followed by delimiter '~'

Fixed IP address

in some cases it may be required to set fixed IP adresses (i.e. if multiple hsmodems operate in the same network).

hsmodem.exe accepts a command line parameter to specify a fixed IP address of the application. If specified the modem will send messages ONLY to this IP.

usage: hsmodem -m 192.168.0.5 (or any other IP address)

Data and Configuration App -> Modem

send an UDP message to the Modem-IP to port 40132 with the following contents:

Byte 0 … File Type
Byte 1 … Frame Information
Byte 2 - 221 … payload with length 219 bytes. If less space is needed then fill the rest with 0x00.

File Type:

Data transfer types (will be sent to the receiver):

- 1 … BER Test Pattern
- 2 … Image
- 3 … Ascii File
- 4 … HTML File
- 5 … Binary File
- 6 … Voice Audio (for Codec 2 or Opus)
- 7 … UserInfo

Configuration types (for internal use, not sent over the air):

- 16 … (reserved for future use)
- 17 … (reserved for future use)
- 18 … (reserved for future use)
- 19 … shutdown (if modem runs on a linux computer, the a “shutdown” command will be executed). No payload for this command.
- 20 … reset modem receiver. Can be used to resync the receiver in case of RX problems. Usually not used because the modem does this automatically. No payload for this command.
- 21 … set playback volume (modem→transceiver) 0..100%
- 22 … set capture volume (transceiver→modem) 0..100%
- 23 … set playback volume (modem→loudspeaker) 0..100%
- 24 … set capture volume (microphone→modem) 0..100%
- 25 … voice mode
- 26 … terminate hsmodem (can be called when app is closed). No payload for this command.
- 27 … send tuning tones
- 28 … send tuning marker
- 29 … (reserved for future use)
- 30 … write a letter or number to RTTY modem for transmission
- 31 … send RTTY string
- 32 … RTTY TX:on/off
- 33 … stop running RTTY transmission immediately. No payload for this command.

Frame Information:

- 0 … first frame of a file

- 1 … next frame
- 2 … last frame
- 3 … single frame (file needs only one frame)

Payload contents for file types 1 - 7 :

common values in Byte 0 and 1:

Byte 0 … file type (as specified above)
Byte 1 … frame information (as specified above)

BER Test Pattern

complete payload is filled with Ascii characters from A..zA..z as many as fit into one payload

Images, Ascii and other Files

Images and file transfer uses the same format with one exception: files are always ZIP compressed while images are not. The format of the file/image transfer is documented in another chapter

Voice Audio (for Codec 2 or Opus)

not used for the application program. Voice is handled by the modem without interaction from the application. The app just switches digital voice mode on/off

UserInfo

used to send personal information to the modem which will automatically transmit if with every new file/image payload contents:
20 bytes: callsign
10 bytes: qth locator
20 bytes: name

Payload contents for file types 16 ...

common values in Byte 0:

Byte 0 … file type (as specified above)
Byte 1 - n … contains following data

Transmission Speed Selection

set the modem's speed. Implemented values:
one byte:
0 … 1200 BPSK BW: 1300 Hz
1 … 2400 BPSK BW: 2500 Hz
2 … 3000 QPSK BW: 1700 Hz
3 … 4000 QPSK BW: 2400 Hz

4 … 4410 QPSK BW: 2500 Hz (QO-100 Standard)
5 … 4800 QPSK BW: 2700 Hz
6 … 5500 8APSK BW: 2300 Hz
7 … 6000 8APSK BW: 2500 Hz (QO-100 Transceiver)
8 … 6600 8APSK BW: 2600 Hz
9 … 7200 8APSK BW: 2700 Hz (QO-100 SDR)
10 .. 45.45 Baud RTTY

playback and capture volume

one byte 0-100% to specify the volume

voice mode

one byte with these values:

0 … switch off the following commands
1 … listed the transceiver audio with the loudspeaker
2 … directly connect microphone to loudspeaker (internal audio loop)
3 … like 2, but route it via the selected codec (internal audio loop via codec)
4 … DV mode, speak and receive digital auio
5 … like 4, but RX only (monitoring mode)
6 … start recording the PCM file: intro.pcm
7 … start playback the PCM file: intro.pcm

send tuning tones

one byte specifying:

0 … switch off tuning tones
1 … send pattern with a peak every 200 Hz
7 … send 1500 Hz peak

send tuning marker

one byte specifying:

0 … switch OFF the 100 Hz marker
1 … switch ON the 100 Hz marker

the tuning tone is ON by default, you should leave it ON, it helps the receiver to set the frequency correctly

write a letter or number to RTTY modem for transmission

one byte containing the letter or number to be sent

send RTTY string

used to send predefined messages.
message as ASCII text

RTTY TX:on/off

one byte 0 or 1: switch on/off the RTTY transmission. If switched on then the RTTY idle pattern will be sent.

File Transfer Format

the File Types 2-5 (Image and File Transfer) are using this payload data which is transferred after Byte 0 (File Type) and Byte 1 (Frame Information). So the following contents start at Byte 2.

The file has to be splited into parts fitting into the payload (219 bytes) until everything is transferred. If the last frame is not completely used fill the remaining bytes with 0x00.

First Frame:

the first frame does not only contain file data but has also a file header with this contents. The Frame Information byte must be set to 0.

- 50 Bytes … Filename. The receiver will store the file under this name. Images should have extension .jpg, HTML file .htm or .html and other files can have any filename.
- 2 Bytes … CRC16 of the file name or any other unique 16bit ID which can be used to identify this file. The receiver will use it to store file fragments in case the file cannot be transferred without errors.
- 3 Bytes … (24bit) file size in Bytes. Please note that the maximum file size in HSmodem is 200kByte. The file size is stored MSB first.
- remaining 163 Bytes … first junk of file data

Example for the first frame of a larger image file:
length: 2 Bytes (Frame Type and Frame Information) + 219 Bytes payload = 221 byte

Byte 0 … 0x02 (file type: image)
Byte 1 … 0x00 (first frame of this file)
Byte 2-51 … filename
Byte 52-53 … CRC16 of the 50 byte long filename
Byte 53-55 … file size in bytes (max 200kByte)
Byte 56-220 … first fragment of file data

Next Frame:

second or consecutive frames. The Frame Information byte must be set to 1.

length: 2 Bytes (Frame Type and Frame Information) + 219 Bytes payload = 221 byte

Byte 0 … 0x02 (file type: image)
Byte 1 … 0x01 (next frame of this file)
Byte 2-220 … next fragment of file data

Last Frame:

Last frame for this file transfer. The Frame Information byte must be set to 2.

length: 2 Bytes (Frame Type and Frame Information) + 219 Bytes payload = 221 byte

Byte 0 … 0x02 (file type: image)
Byte 1 … 0x02 (next frame of this file)
Byte 2-220 … last fragment of file data (unused bytes filled with 0x00)

very small files:

for small files using only one frame (length ⇐163): the Frame Information byte must be set to 3.

Example for a very small file:
length: 2 Bytes (Frame Type and Frame Information) + 219 Bytes payload = 221 byte

Byte 0 … 0x02 (file type: image)
Byte 1 … 0x03 (first+last frame of this file)
Byte 2-51 … filename
Byte 52-53 … CRC16 of the 50 byte long filename
Byte 53-55 … file size in bytes (max 200kByte)
Byte 56-220 … file data (unused bytes filled with 0x00)

Compression

Images: images have to be sent as JPG files with a resolution of max. 640 x 480 pixel or less (this is just a limit of the oscardata.exe GUI, HSmodem can transfer any size as long as the total file size is smaller 200kbyte)

Files (Ascii, HTML or binary): files have to be ZIP compressed before transmission.

ZIP-Filename: must be the pure filename (without path) of the file to be transferred.
Filename of the compressed file inside the ZIP file: same as ZIP-Filename.

The ZIP-Filename will be shown to the user during a reception. The filename inside the ZIP will be used to store this file on the receivers disk, so the best rule is to keep both filenames identical.

Data and Configuration Modem -> App

the modem sends these data to the application via UDP port 40133:
the first Byte is a message ID:

- 3: responses to broadcast messages (see: GUI Interface: UDP/IP/Initialization)
- 1: received payload data
- 4: FFT data for a spectrum monitor
- 5: IQ data for a constellation display
- 6: received RTTY characters

Response to broadcast

see: GUI Interface: UDP/IP/Initialization

Data Reception

when the modem detects incoming data it unpacks the payload and sends this message to the user application:

```
Byte 0 ... 0x01
Byte 1 ... frame type (which was inserted by the sender)
Byte 2 ... frame counter MSB
Byte 3 ... frame counter LSB (10 bits used)
Byte 4 ... frame information (which was inserted by the sender)
Byte 5 ... unused
Byte 6 ... measured line speed MSB
Byte 7 ... measured line speed LSB
Bytes 8-10 ... unused
Bytes 11-229 ... 219 bytes payload
```

FFT data for a spectrum monitor

```
Byte 0 ... 0x04
Byte 1 ... usage of the TX fifo (used by the transmitter to sync its data
           output to the modem). This is a value between 0..255. During
           an active transmission keep it above 4.
Byte 2 ... usage of RX fifo (not important, but can be displayed to the
           user). A very high RX fifo usage indicates the the computer
           is too slow for HSmodem.
Byte 3 ... 0 or 1. Indicates that an RF level was detected
Byte 4 ... 0 or 1. Indicates that the HSmodem receiver is synchronized
           with a signal
Byte 5 ... maximum audio level (0..100%) of the audio input from the
           transceiver. Can be used to detect clipping.
Byte 6 ... maximum audio level (0..100%) of the audio output to the
           transceiver. Can be used to detect clipping.
Byte 7 ... in RTTY mode this is the auto-locked RTTY frequency MSB
Byte 8 ... and LSB
Byte 9 ... RTTY: 0=tx off, 1=txon
Byte 10 to the end ... FFT spectrum data, beginning at 0 Hz to 4kHz with
           a resolution of 10 Hz
```

IQ Data for constellation diagram

```
Byte 0 ... 0x05
Byte 1 ... point-1 real part MSB
Byte 2 ... point-1 real part LSB
Byte 3 ... point-1 imaginary part MSB
Byte 4 ... point-1 imaginary part LSB
Byte 5 ... point-2 real part MSB
Byte 6 ... point-2 real part LSB
Byte 7 ... point-2 imaginary part MSB
Byte 8 ... point-2 imaginary part LSB
.
.
.
. to the end of the message
```

not all IQ values are transferred to avoid flooding the computer running the GUI. But the number is sufficient for a good constellation display.

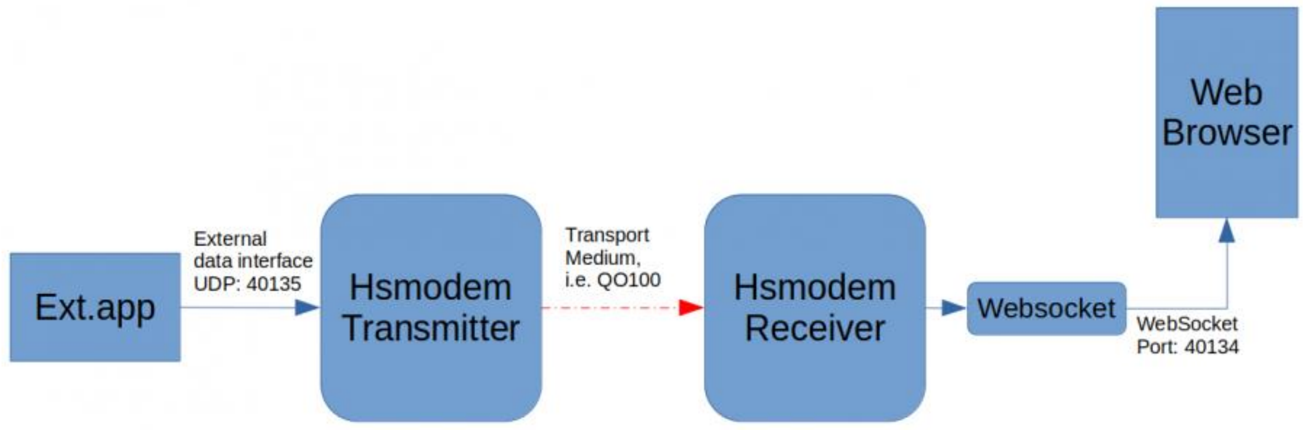
RTTY

```
Byte 0 ... 0x06
Byte 1 ... received RTTY character (Ascii)
Byte 2 ... 0 (unused)
Byte 3 ... 0=RX idle, 1=RX in sync with a received RTTY signal
```

External Data Interface

the external data interface is an UDP socket which accepts any kind of data which will be sent via HSmodem with the selected operation mode/speed.

Example: You want to send data of your weather station via HSmodem. In this case you collect and format the data with some other software, then you send this pre-formatted data into the external data interface of HSmodem. This picture shows the data flow of external messages:



Example:

A message is sent from an external application (i.e. weather station) to HSmodem. The receiver should have a user interface (running in a web browser) to see a nice presentation of your weather data. This project requires the following steps:

Preparation:

1. write a program which gets the data from the weather station and formats it according to the following data format
2. send this pre-formatted data into HSmodem via UDP port 40135
3. make a HTML file based on websample.html which presents your weather data at the receiving stations

Operation:

- using the HSmodem HTML-Filetransfer feature send your HTML file to the receiving stations
- stream the weather data into HSmodem and all receiving stations will see it in their browsers

Transmission:

The external message is sent via HSmodem to the transceiver. This is done in parallel to any other running transmission: you can send a picture and additionally send external data. Since both data streams share the same channel the transmission will slow down if too many external messages are sent. But in case of above example (weather station) you i.e. send only every minute, then this will work without any slowdown of the current transmissions.

Data Format: the external data interface accepts messages in this format:

Total Length: 224 Byte

Byte 0 … ID MSB
Byte 1 … ID
Byte 2 … ID
Byte 3 … ID LSB
Byte 4 … message type
Byte 5-223 … message

message ID : a 32 bit random number and works as a first filter for your personal installation to remove any "random" unwanted data. This ID should be a secret number, do not use the default if you plan to open this UDP interface to the internet. Your local HSmodem installation must also have this ID, you can modify it in extData.c.

message type : a number between 16 and 255 specifying the type of the following data. To keep HSmodem compatible between all installations it is highly recommended to request a message type number from Amsat-DL. If you just need a type number for experimentation then use 255.

message : a byte stream with length 219 bytes. May contain any possible data. It is recommended to insert a personal ID to your data. This allows your receiving HTML script to filter out unwanted random data, if somebody else uses the same message type.

Reception:

On the receiving side HSmodem will detect external messages and separate it from other data streams. The message is then forwarded to a websocket-server (included in HSmodem) which sends it via TCP port 40134 to all connected webbrowsers. No webserver is required, a local HTML page stored as a file can be used. The websocket connection is done by a small piece of javascript.

Now the data are in the bowser and you can present it to the user the way you want.

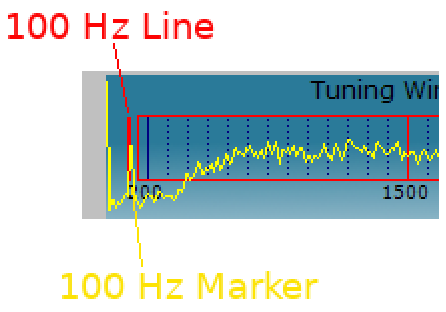
Implementation in the web browser:

for an example how to do the javascript implementation see the file websample.html which is included in the github project in folder …hsmodem/html

Frequently Asked Questions:

Why is there a line in the spectrum at 100 Hz?

this line is a tuning aid. This makes it very easy to find the exact frequency. In the spectrum display there is a red line at 100 Hz. You bring the 100 Hz peak to this line and the modem will synchronize immediately. A drift “after” locking is not a problem, it is not necessary to always follow the line.



The received, yellow 100 Hz peak should lie reasonably on the red 100 Hz line.

Should you readjust the frequency while reception is running?

This is not necessary. Once the modem is locked in, it has a relatively large capture range.

Transceiver: a careful frequency correction is possible

SDR console: The frequency must NOT be changed during reception. The SDR console makes tiny dropouts during frequency changes, which lead to reception errors.

Is the Raspberry-4 suitable for HS modem?

yes, but a 64-bit operating system must be used, which is available: Raspberry PI OS 64

What is the status of the DV operating modes?

Digital voice with Codec2 or Opus codec was implemented just for fun.
It is not intended to replace or compete with FreeDV. It cannot do that either because the latency times are far too long for QSO operation.

Due to its high speed, HS-Modem is primarily intended for higher bit rates, which are currently not supported by Codec-2. If there will be a Codec-2 with bit rates in the range 4000 to 6000 in the future, it makes sense to operate it in HS modem, and then it will be installed immediately.

In the meantime, FreeDV is the ideal program for DV operation.

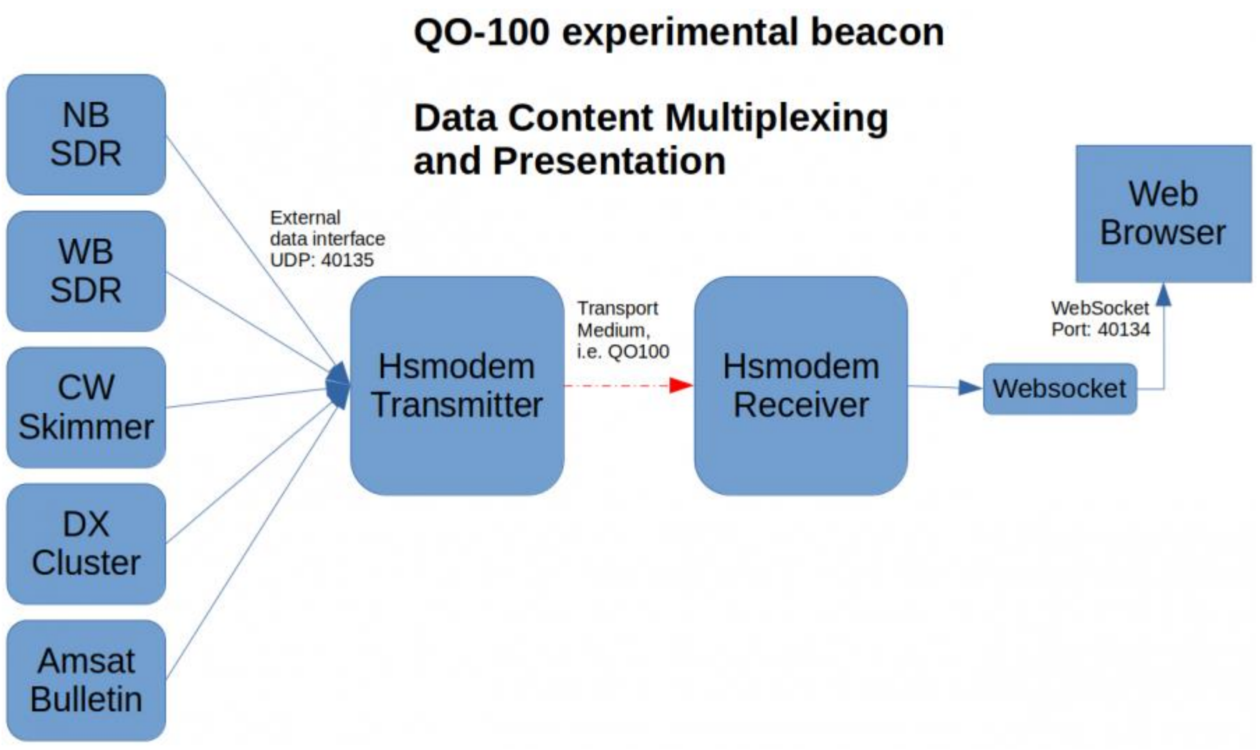
The QO-100 Multimedia Beacon

This beacon can be received in experimental mode on 10489.580 MHz, at irregular times. A high-speed modem version 0.8 or newer is required.

The multimedia beacon uses the streaming mode of HS modem to transmit several data streams at the same time. The individual data streams are mixed in a time grid of 250ms, so that a simultaneous impression is created for the recipient.

At the current status 5 streams are transmitted:

1. Narrow band transponder spectrum and waterfall
2. Wide band transponder spectrum and waterfall
3. DX Cluster (all frequencies, selectable)
4. CW skimmer (QO-100)
5. Amsat Bulletin, breaking news


QO-100 experimental beacon
Data Content Multiplexing and Presentation
NB SDR
WB SDR
CW Skimmer
DX Cluster
Amsat Bulletin
External data interface UDP: 40135
Hsmodem Transmitter
Transport Medium, i.e. QO100
Hsmodem Receiver
Websocket
WebSocket Port: 40134
Web Browser

The data streams are fed into the multimedia HS modem beacon at the Amsat-DL ground station and sent to the QO-100. At the receiver, the data streams are separated automatically, and operation is very simple.

Receiving the QO-100 multimedia beacon

If you want to receive the beacon and see the information, do the following:

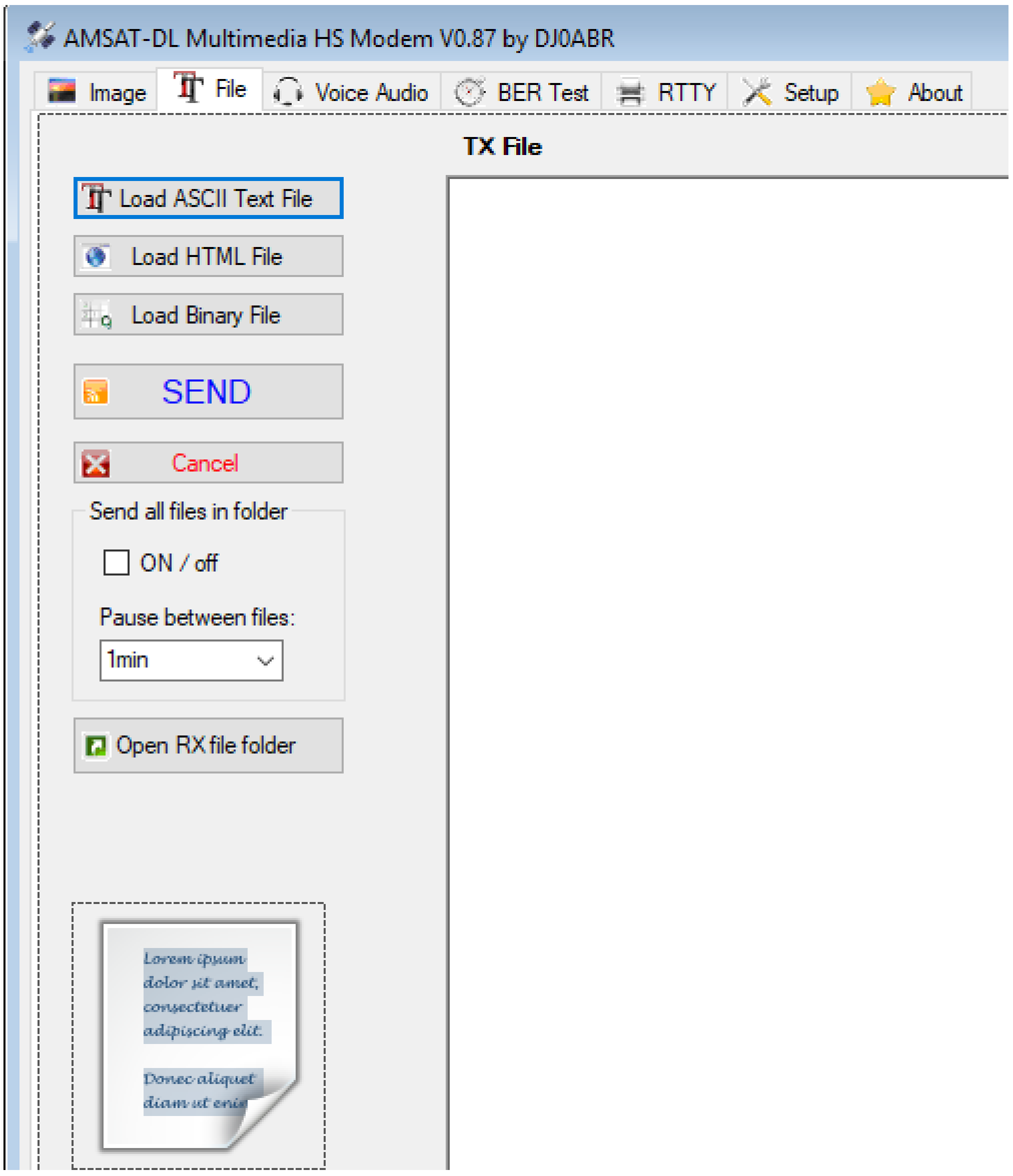
1. Tune the receiver with HS modem running to the beacon
2. Set the HS modem to 8APSK - 7200 bps

3. check that your RX filter bandwidth is at maximum (3.2kHz or even more), HSmodem has its own filters and does not need additional filters in the receiver.
4. Fine-tune the frequency while observing the 100Hz subcarrier until the synchronization is established.
5. Open the file QO100info.html and you will see the information transferred.

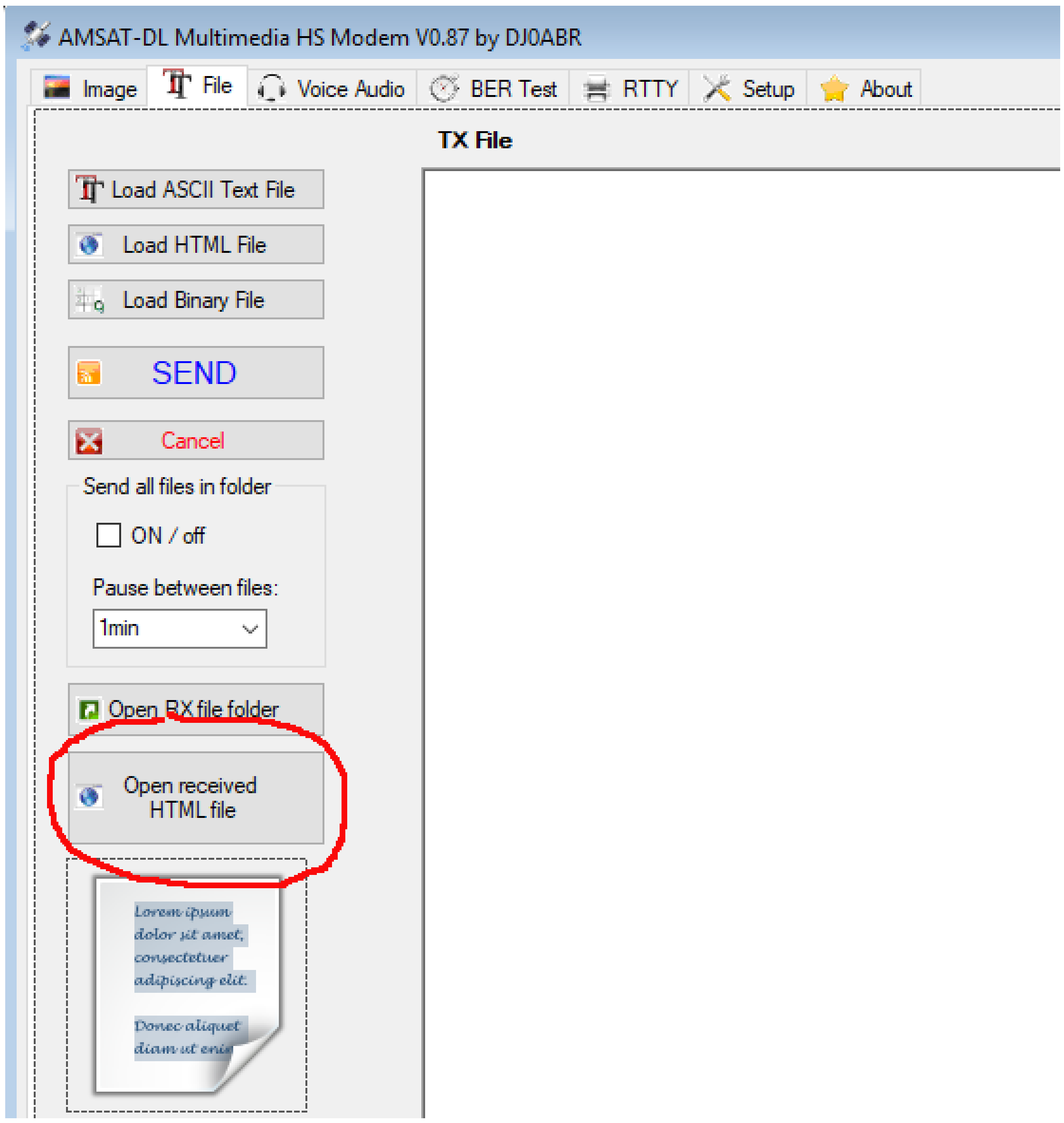
Open the file QO100info.html

this file is transmitted via the beacon every few minutes.

If the file QO100info.html has not yet been received, it looks like this:



As soon as the file QO100info.html has been received, an additional button is displayed:



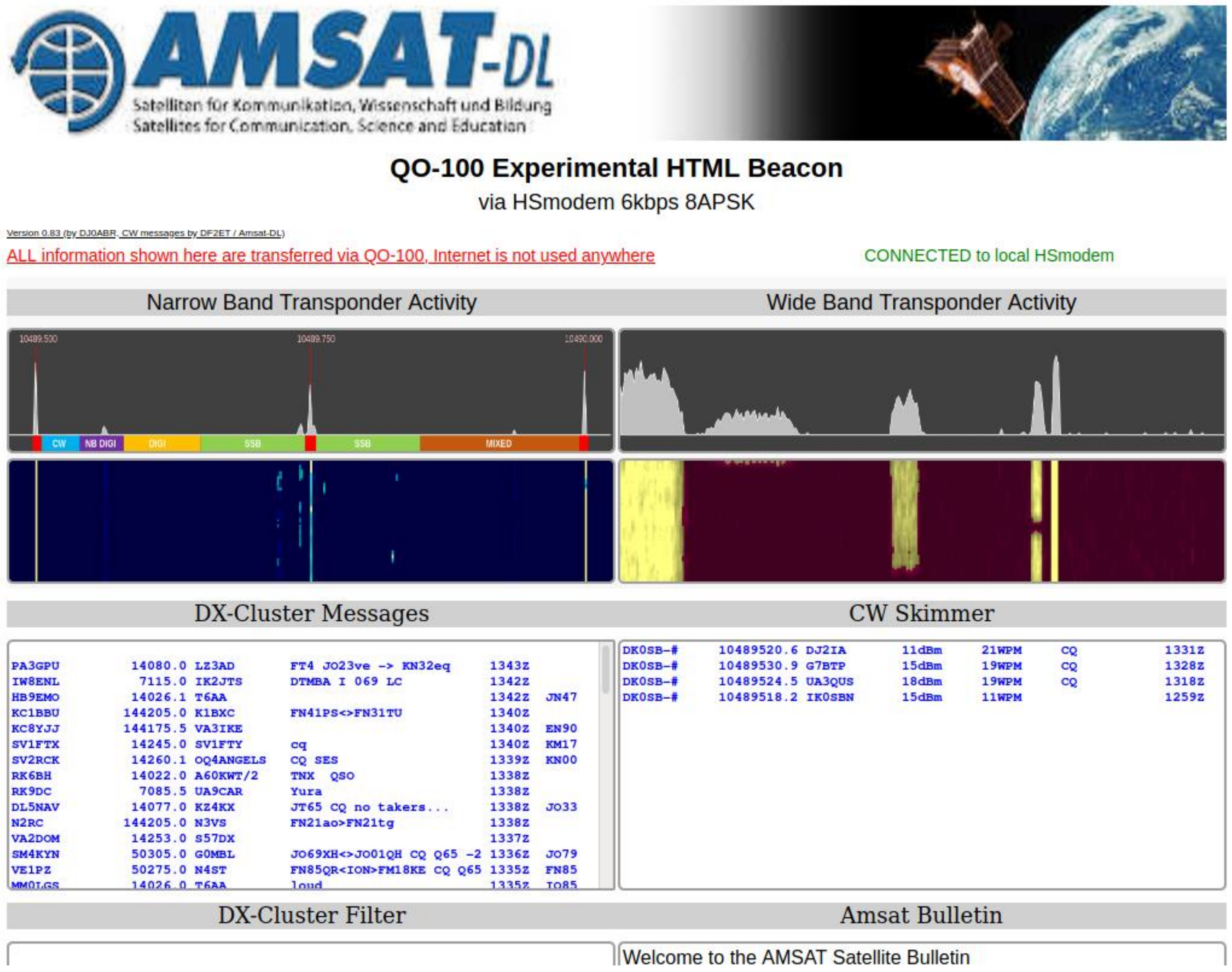
If you click on the button which is marked in the screenshot with a red circle, the web browser is opened and the file QO100info.html is displayed.

if the file QO100info.html has already been received

Then it is already saved on the hard drive and you don't have to wait any longer. Press “open RX files”. Here you can find QO100info.html and open it with a double click.

Using QO100info.html

This is how the beacon QO100info.html is displayed in the browser (note that the layout is getting updated sometime, thus it might look a bit different):



Satelliten für Kommunikation, Wissenschaft und Bildung
Satellites for Communication, Science and Education

QO-100 Experimental HTML Beacon

via HSmodem 6kbps 8APSK

Version 0.83 (by DJ0ABR, CW messages by DF2ET / Amsat-DL)

ALL information shown here are transferred via QO-100, Internet is not used anywhere

CONNECTED to local HSmodem

Narrow Band Transponder Activity

Wide Band Transponder Activity

DX-Cluster Messages

```
PA3GPU       14080.0 LZ3AD      FT4 JO23ve -> KN32eq     1343z
IW8ENL        7115.0 IK2JTS     DTMBA I 069 LC           1342z
HB9EMO       14026.1 T6AA                                1342z  JN47
KC1BBU      144205.0 K1BXC      FN41PS<>FN31TU           1340z
KC8YJJ      144175.5 VA3IKE                              1340z  EN90
SV1FTX       14245.0 SV1FTY     cq                       1340z  KM17
SV2RCK       14260.1 OQ4ANGELS  CQ SES                   1339z  KN00
RK6BH        14022.0 A60KWT/2   TNX  QSO                 1338z
RK9DC         7085.5 UA9CAR     Yura                     1338z
DL5NAV       14077.0 KZ4KX      JT65 CQ no takers...     1338z  JO33
N2RC        144205.0 N3VS       FN21ao>FN21tg            1338z
VA2DOM       14253.0 S57DX                               1337z
SM4KYN       50305.0 G0MBL      JO69XH<>JO01QH CQ Q65 -2 1336z  JO79
VE1PZ        50275.0 N4ST       FN85QR<ION>FM18KE CQ Q65 1335z  FN85
MM0LGS       14026.0 T6AA       loud                     1335z  IO85
```

CW Skimmer

```
DK0SB-#    10489520.6 DJ2IA     11dBm    21WPM    CQ        1331z
DK0SB-#    10489530.9 G7BTP     15dBm    19WPM    CQ        1328z
DK0SB-#    10489524.5 UA3QUS    18dBm    19WPM    CQ        1318z
DK0SB-#    10489518.2 IK0SBN    15dBm    11WPM              1259z
```

DX-Cluster Filter

Amsat Bulletin

Welcome to the AMSAT Satellite Bulletin

A few seconds after opening the file in the browser, a connection to the HS-Modem is established and “Connected to local HSModem” appears in green at the top right. If this message does not appear, HS-Modem is not running or may have to be restarted. With a Windows 10 PC you may be asked whether you want to release port 40134, which you also have to do so that the data can be forwarded to the browser.

The beacon can now be viewed, no operation is required, the information is constantly updated. Filters are available for the DX cluster messages that can be set as required.

References:

- Wikipedia: Shannon-Hartley
- Shifra Error Correction
- libsoundio Audio Library
- FFTW3 Library
- liquid-SDR library
- Codec-2
- Microsoft Packages