Rivet info

revision date: January 15 January 2026

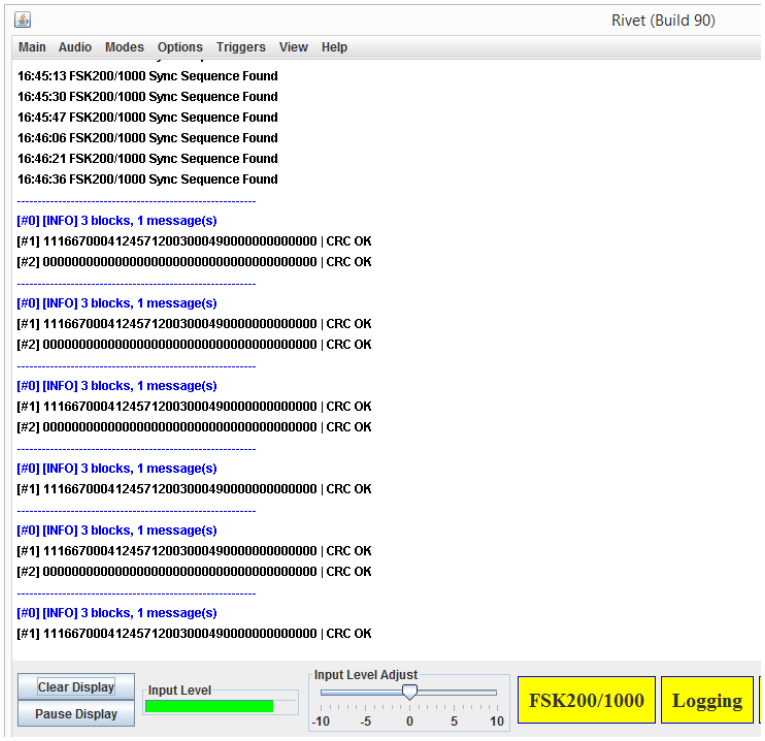
Rivet is a popular free decoder originally created by Ian Wraith. This manual is derived from info from the Rivet website plus some additional info. Compiled for UDXF and Numbers & Oddities by Ary Boender.

The original Rivet decoder was written by Ian Wraith. Updates by the Priyom Team.

Changes to build 91:
Selectable F06a decoding added in two different formats (ASCII and binary)
Improved F06a detection when using FSK200/1000
Improved FSK200/1000 metadata block displays
Minor fixes and display changes in FSK200/500, XPA, XPA2, Baudot
Add and update some links
Gradle wrapper update to 7.5

Download at: https://github.com/priyom/Rivet/releases

A word from Ian Wraith:



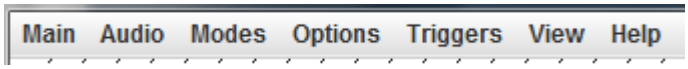
Rivet is a open source decoder written in Java which decodes various HF data modes that have either been forgotten by the better known decoder programs or which can only be decoded by expensive programs that are beyond the reach of the average hobbyist. The program doesn't need any other hardware to operate all it requires is a sound card with an audio feed from HF receiver.

Those of you who have used decoder programs for HF data modes before will find Rivet a little different. Firstly Rivet doesn't need you to precisely tune your receiver to signal for it to be decoded. Instead Rivet takes advantage of a modern PCs processing power and adjusts itself to the incoming signal. This means you don't waste vital minutes messing around with a tuning indicator when you want to see what is being transmitted. The downside is that Rivet does need to run on a fairly powerful PC with at least 2 cores. Secondly Rivet is designed not only to accept a direct sound input from a sound card but it can also decode data from a .WAV file (which must be mono and have a sample rate of 8 KHz). So you can record a transmission while you are searching through the HF spectrum and record it later. If you click here you will find several recordings you can use to test the program.

Lastly Rivet is a free and open source program something which is very unusual for a HF data mode decoder. To make it better I need your input so if you find any bugs or have any feature suggestions please email me. If you are a programmer then use Github to fork Rivet and add some new features yourself then if I like them I will add them to the program.

Ian Wraith

Menus



Main

- **Copy All to the Clipboard**
  The entire contents of the screen display are copied into your clipboard. Useful if you wish to include a section of a decode from Rivet in an email or word processor document.
- **Load a WAV File**
  Rivet not only decodes audio directly from your radio but can also decode data contained in WAV files which were recorded previously. Note that Rivet can only decode WAV files which were recorded in mono mode and at certain sample rates (usually 8000 Hz).
- **Reset Decoding State**
  If this menu item is selected then Rivet resets to the default settings for the particular mode you have selected. So if Rivet has set its levels and symbol timing on a signal and this option is selected then Rivet will reset and try to reacquire levels and symbol timing.
- **Save the Current Settings**
  If you click on this all of the settings you have selected (Mode etc) will be saved in a file called "rivet_settings.xml". From then onwards when you restart Rivet the program will load these settings and use them as its default. To change this select the settings you would like to be your new default and click on this item again.
- **Save to File**
  When this option is enabled all traffic that is decoded by Rivet will also be saved to an ASCII text file. When you select this option a dialog box will appear requesting you give a name and a location for this file.
- **Save Bit Stream to File**
  When you enable this option the program will ask you to select a file name and folder for the file. Afterwards the raw binary output from the program will be a saved in that file (which has a .bsf prefix). It is then possible for you to analyse the data in this file for the purpose of reverse engineering a mode. Please note that this option only currently works when decoding selected modes.
- **Soundcard Input**
  When this is enabled Rivet will accept and attempt to decode audio from the audio source selected by your PCs mixer. I would recommend that you use your PCs Line Input source for this.
- **Exit**
  Use this to shut down the program.

Audio

- **Audio devices**
  Choose the sound source

Modes

Use this menu to select the mode you wish to decode.

- Baudot
- CCIR493-4 : a HF selective calling mode
- CIS36-50 (50 baud only currently) : used by the Russian Navy
- CROWD36 : used for Russian diplomatic and intelligence traffic
- F01 (FSK200/500) : used for Russian diplomatic and intelligence traffic
- F06 (FSK200/1000) : used for Russian diplomatic and intelligence traffic
- F06a (FSK 200/1000) : used for Russian diplomatic and intelligence traffic
- FSK (raw) : for advanced users to investigate unknown FSK modes
- GW FSK (100 baud) : a commercial ship to shore data system (now defunct)

- XPA (10 baud) : used for Russian intelligence traffic
- XPA (20 baud) : used for Russian intelligence traffic
- XPA2 : used for Russian intelligence traffic
- Experimental

Options

- **Baudot & FSK options**
  Several options for Baudot & FSK modes
- **CIS36-50 options**
  Several options for the CIS36-50 mode.
- **Debug Mode**
  When this mode is enabled Rivet will display debugging and diagnostic information about the signal it is decoding. Unless you are a developer I wouldn't recommend you enable this option.
- **Invert**
  When enabled the data received by Rivet will be inverted. So for example if a FSK (Frequency Shift Keying) mode is being received and the high tone normally represents a 1 and a low tone a 0 , if this option is enabled then the high tone will represent a 0 and the low tone a 1. In some modes (e.g CCIR493-4) Rivet will automatically decide if Invert needs to be enabled or disabled and will change it accordingly.
- **Set the CROWD36 High Sync Tone**
  This option is used when you are in CROWD36 mode to tell Rivet the tone number of the high sync tone being used by the CROWD36 station you are monitoring.
- **F06a ASCII parsing**
  Process of reading and interpreting data from plain text (ASCII) files.

Triggers

See appendix.

View

- **Clear display**
- **Display possible bad data**
- **Display UTC time**
- **View GW Free Channel Markers** (the GW mode is defunct)

Help

- **About Rivet :** Informs you of the version of Rivet you are using and some information about the author.
- **Credits**
- **Enigma2000 :** Link to the Enigma2000 group website
- **UDXF:** Link to the UDXF website.
- **Follow Rivet riginal auhtor on Twitter:** Link to Rivets authors Twitter page.
- **Help:** This takes you to the Rivet help pages.
- **Download the latest version of Rivet or Sound Sample Files**
- **System information**
- **Priyom.org:** Link to the Priyom.org website

Appendix

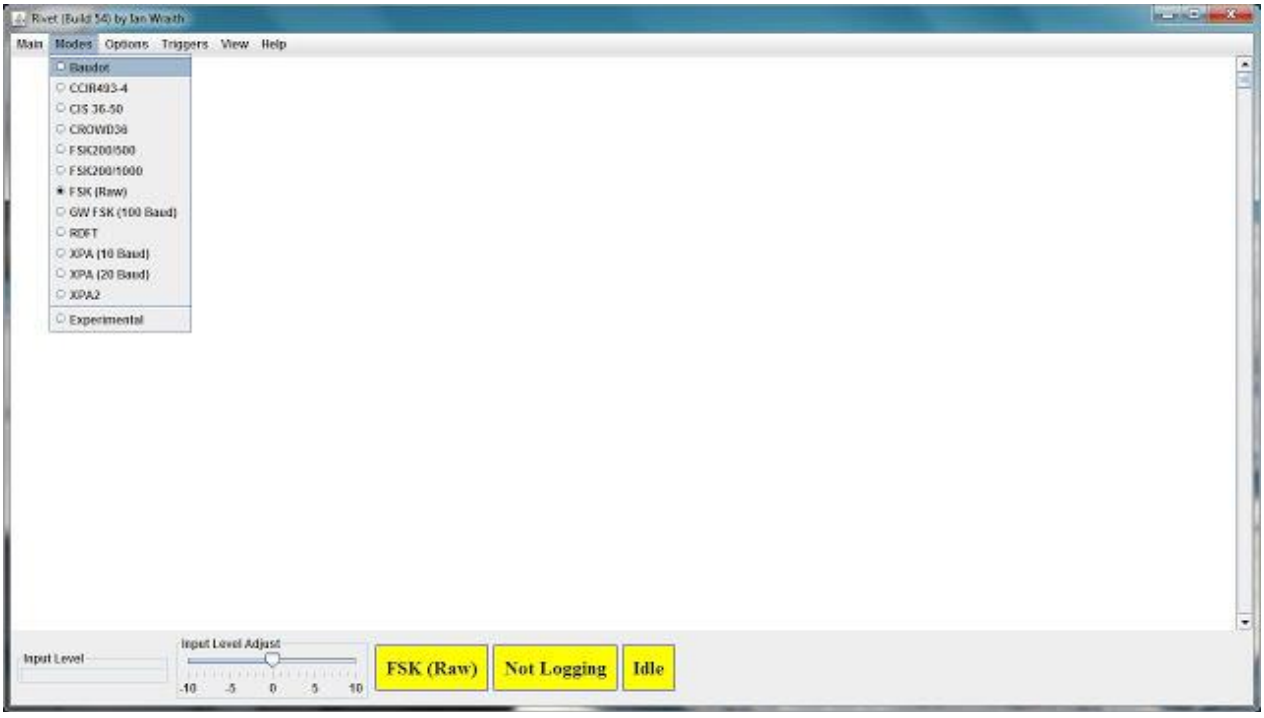
An introduction to the Rivet Trigger feature

Rivet build 90 allows the user to add , edit and delete Triggers from within the program. Previously users had to manually edit a XML file using a text editor which I realise isn't easy if you don't have any experience with that sort of thing. Since using Triggers is now within anyone's grasp I thought I would write a little step by step tutorial on how to use it.

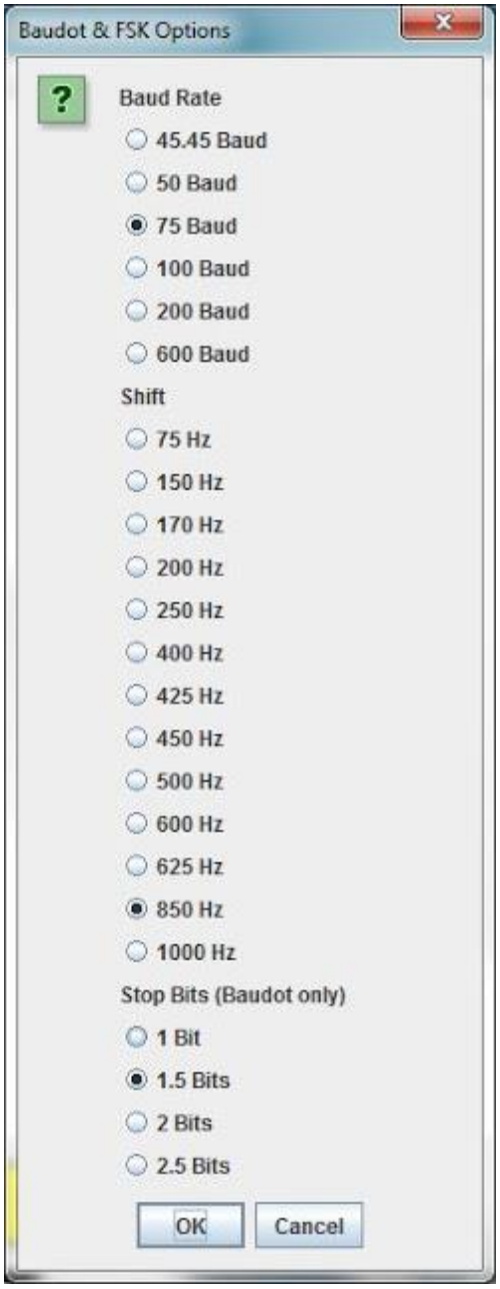
Firstly I had better explain what Triggers are. Rivet has a mode called **FSK (Raw)** which is intended for users who wish to investigate unknown or little known FSK modes. When selected and after the correct baud rate and shift have been set Rivet displays what the station is transmitting as binary. This is useful when you are initially investigating a mode but soon the amount of data becomes rather overwhelming. To make this mode more useful one of Rivet's users suggested a feature we decided to call a Trigger. Now a Trigger is nothing more than a binary sequence which Rivet stores in its memory and constantly compares with incoming raw FSK data. When the incoming data matches the binary sequence in the trigger then the program does *something* where the something depends on the type of trigger. So far there are three types of trigger:

- **Start** trigger. With this type of trigger Rivet displays nothing until it receives the sequence of data defined in the start trigger. At which point the program displays the name of the trigger that has been activated and then displays all incoming data.
- **End** trigger. When Rivet matches the sequence of data in this type of trigger with the incoming data it displays the name of the trigger and then stops displaying incoming data.
- **Grab** trigger. With this type of Trigger when Rivet matches the incoming data with the trigger sequence then it displays the name of the Trigger , the previous *backward grab* number of bits (a number defined by the user) which were transmitted before the trigger sequence and the *forward grab* number of bits (which again has been defined by the user) which follow the trigger sequence. This type of trigger is very useful for packets of data where the synchronisation sequence is in the middle of the packet.

So that is what triggers are now let me show you how to use them. For this example we will have a look at a 75 baud 850 Hz shift synchronous FSK signal using KG-84 encryption most likely sent by the armed forces of a NATO member country. To use it start up Rivet then select **Raw (FSK)** as the decoder mode.



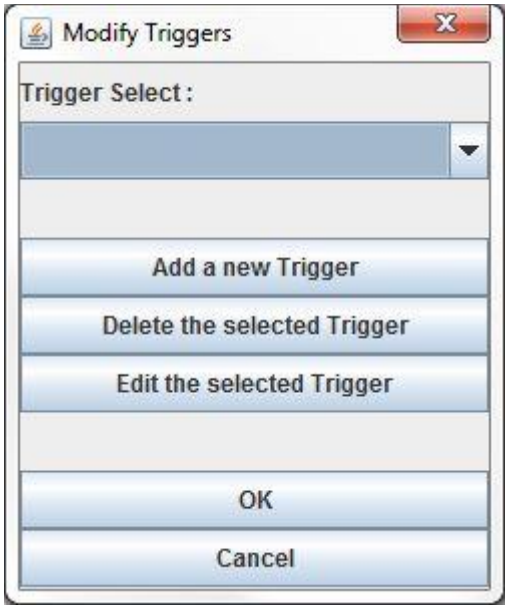
Next from the **Options** menu click on the **Baudot & FSK Options item**. Now select the options "75 Baud" and "850 Hz" as you can see below ..



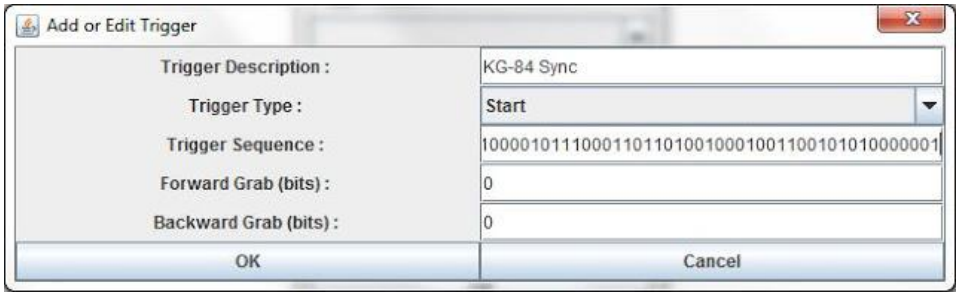
(Ignore the Stop Bits section which doesn't apply here). Now if you were to load the KG-84 sound sample all you would see is a lot of a binary data. Now it is common knowledge that KG-84 crypto systems use the following binary sequence for synchronisation ..

1111101111001110101100001011100011011010010001001100101010000001

So what we shall do is make this sequence so it is a Rivet start trigger. To do this click on the Triggers menu followed by the "Add , Edit or Delete a Trigger" item. Once you do that then this dialog box will appear:



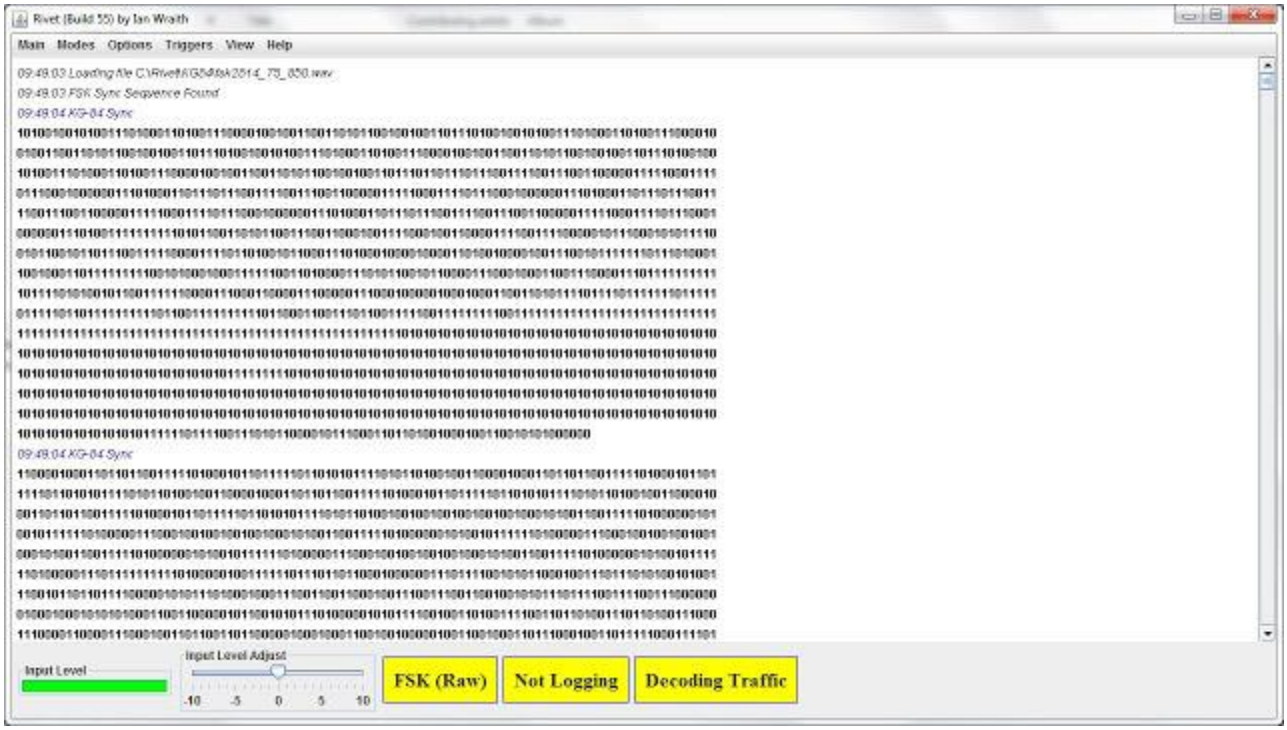
Now click on the "Add a new Trigger" button at which point you will see another dialog box which you need to fill in to look like this:



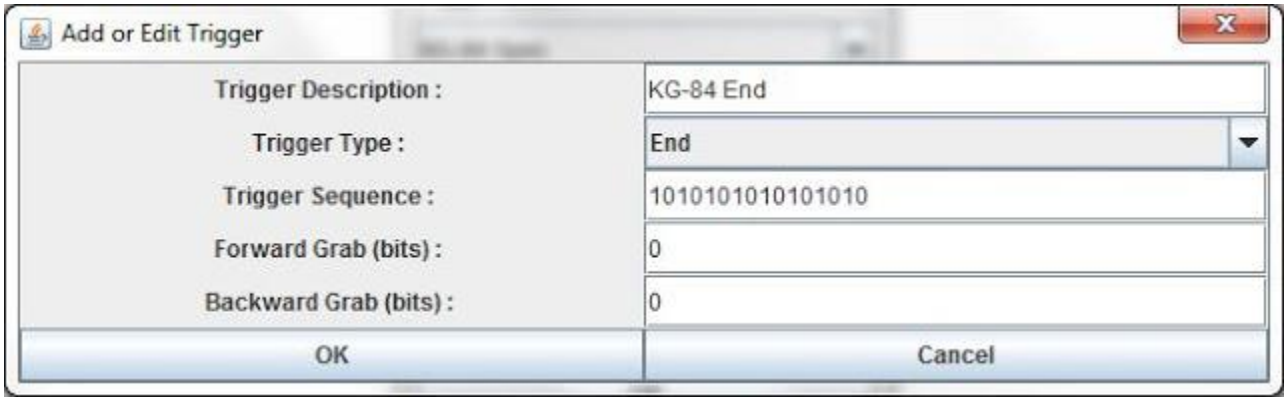
(you can cut and paste the binary value from this blog post rather than typing it in yourself). After that click on the OK button in this dialog box followed by the OK button in the Modify Triggers dialog box. Once you have done that the program will return to the main screen and automatically save the trigger to your hard disk.

Next you need to enable the new Trigger. Click on the Triggers menu and you will see the name of your new trigger with a radio button next to it. Click on this to enable the trigger as you see below.

With that all done either tune into a KG-84 transmission or load the recording I mentioned earlier. When you do that you will see something like the picture below.

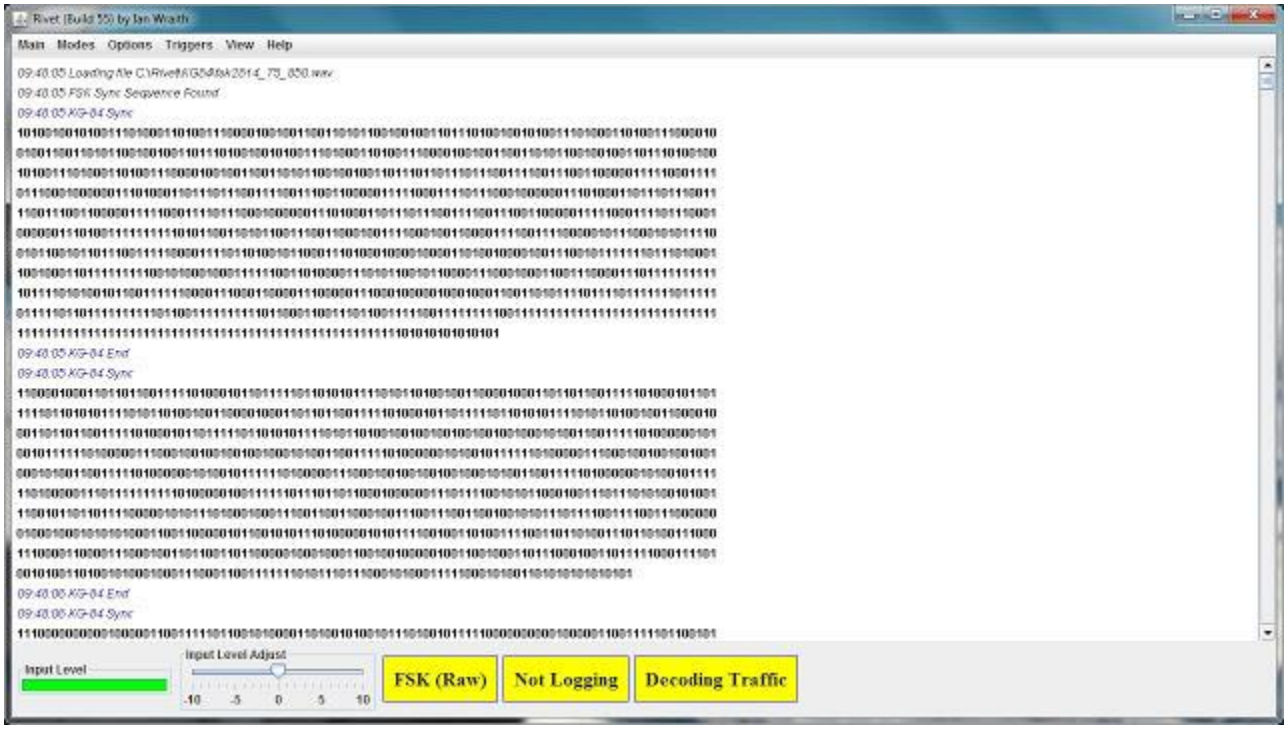


You can see from the words "KG-84 Sync" in blue (this triggers name) that the trigger activated several times. The display shows the trigger sequence followed by the data transmitted afterwards. Now that's all well but the display is cluttered with the alternating bits that are transmitted between the messages. It would be better get rid of those. We can do this by setting the alternating sequence of bits ***1010101010101010*** as an end trigger. So when Rivet detects this sequence it stops displaying decoded data until another start trigger sequence is received. To add this trigger click on the Trigger menu then again on the "Add , Edit or Delete a Trigger" item , then on the "Add a New Trigger Button" and fill in the dialog box that appears in the following way.



Click once again on both OK buttons and then enable the new trigger.

Now if you try to decode a KG-84 recording or live transmission you will see something looking like this …



Both types of triggers can be seen (in blue text) when activated and the KG-84 messages (or blocks of a message) are clearly defined.