IrMaster documentation

Table of contents

Note:
Development of this program has been discontinued. It has been superseeded by IrScrutinizer, offering much more functionality.

Warning:
Sending undocumented IR commands to your equipment may damage or even destroy it. By using this program, you agree to take the responsibility for possible damages yourself, and not to hold the author responsible.

1 Revision history

Date	Description
2011-10-23	Initial version.
2012-04-14	Many minor fixes and updates for the upcoming 0.1.2. (version not published)
2012-04-24	Converted to the document format of Apache Forrest. The program documentation is now generated from that file.
2012-06-06	Updated for upcoming version 0.2.0. Many minor impovements. Plotting and Audio new.
2012-08-19	(Not) updated for upcoming version 0.3.0.
2012-11-18	Updated for upcoming version 0.3.1.
2014-01-27	Updated for upcoming version 1.0.0.

2 Introduction

This is what the program can do: From a data base of known IR signals "recepies" (known as the IRP-notation, essentially corresponding to the collected know-how of the community in machine readable form), IR signals corresponding to certain parameter values can be computed. Export files in different formats can be generated, for usage of other programs. For this, two alternative renders (my own IrpMaster as well as the older Makehex) are available. By using the clipboard, IR signals in Pronto format (for example from Internet articles) can be directly sent to the analyzers AnalyzeIR and DecodeIR. An entered or computed signal can be sent proper hardware, or plotted. For investigating possible non-documented IR signals of owned equipment, a "war dialer" can send whole parameter regions of IR signals. For the latter possibilities, hardware support in the form of GlobalCaché, IRTrans, a LIRC server, or an IR-audio setup, is required. A simple calculator intended for simple computations on IR signal timings is provided.

This program is not a "program with a GUI", nor is it a GUI for a particular program. Rather, it is a "Meta program", offering a GUI for a number of IR related programs, presently IrpMaster (advanced IR rendering program by myself), Makehex (older IR rendering program), DecodeIR (tries to identify an IR signal), and AnalyzeIR (which is my name of the Analyze-Function of the ExchangeIR-Library), and the PtPlot library used for plotting of IR signals. Future extensions to other, possibly not yet written, programs are possible.

Note that I have written two different programs with quite similar names: IrMaster, the present one, a GUI, and Ir**p**Master, a library and a command line program, an IR signal render, but without a GUI. Please do not confuse.

In the sequel, the word "the program" will denote either the "shell" IrMaster, or the GUI together with its "clients" IrpMaster, Makehex, AnalyzeIR, and DecodeIR, as is hopefully clear from the context.

For someone with knowledge in the problem domain of IR signals and their parameterization, this program is believed to be very simple to use. This knowledge is assumed from the reader. Other can acquire that knowledge either from the JP1 Wiki or, e.g., this link.

Note that the screen shots are included as illustrations only; they may not depict the current program completely accurately. They come from different versions of the program, using different platforms (Linux and Windows), and using different "look and feels".

The present document is written more for completeness than for easy accessability. Possibly, in the future, there will be a user's manual as well as a reference manual.

Release notes for the current version.

2.1 Copyright and License

The program, as well as this document, is copyright by myself. My copyright does not extend to the embedded "components" Analyze, Makehex, DecodeIR, and PtPlot. Makehex was written by John S. Fine (see LICENSE_makehex.txt), and has been translated to Java by Bengt Martensson. ExchangeIR was written by Graham Dixon and published under GPL3 license. Its Analyze-function has been translated to Java by Bengt Martensson. DecodeIR was originally written by John S. Fine, with later contributions from others. It is free software with undetermined license. PtPlot is a part of the Ptolemy Project at the EECS department at UC Berkeley, licensed under the UC Berkeley copyright. IrpMaster is using ANTLR3.4 and depends on the run time functions of ANTLR3, which is free software with BSD license.

The "database file" IrpProtocols.ini is derived from DecodeIR.html, thus I do not claim copyright.

The `main()` method of the Wave class uses JCommander by Cédric Beust to parse the command line argument. (Likely, I will use it much more in the future.) It is free software with Apache 2 license.

Icons by Everaldo Coelho from the Crystal project are used; these are released under the LGPL license.

The Windows installer was built with Inno Setup, which is free software by Jordan Russel. To modify the user's path in Windows, the Inno extension modpath by Jared Breland, distributed under the GNU Lesser General Public License (LGPL), version 3.

The program and its documentation are licensed under the GNU General Public License version 3, making everyone free to use, study, improve, etc., under certain conditions.

2.2 Privacy note

Some functions (Help -> Project Homepage, Help -> IRP Notation Spec, Help -> Protocol Specs, Tools -> Check for updates) access the Internet using standard http calls. This causes the orginating machine's IP-address, time and date, the used browser, and possibly other information to be stored on the called server. If this is a concern for you, please do not use this (non-critical) functionallity (or block your computer's internet access).

3 Installation

3.1 General

IrMaster, and all but one of its third-party additions, are written in Java, which means that it should run on every computer with a modern Java installed; Windows, Linux, Macintosh, etc. Java 1.6 or later is required. The one exception is DecodeIR, which is written in C++, and invoked as a shared library (`.dll` in Windows, `.so` in Linux, etc). If DecodeIR is not available on your platform it is not a major problem, as IrMaster will work fine without it; just the DecodeIR-related functions will be unavailable.

There is unfortunately no good `make install` or such in the source distribution, so also source code distribution users are recommended to install the binary distribution. Also, all necessary third-party components are included in the binary distribution.

Both under Windows as well as under other operating systems, IrMaster (and IrpMaster) behave civilized, in that they do not write in the installation directory after the initial installation. In both cases (in contrast to the source distribution), the distribution contains everything needed including third party libraries like DecodeIR, AnalyzeIR, MakeHex (Java version) and its irp-files.

Under Windows, the properties are stored in `%LOCALAPPDATA%\IrMaster\IrMaster.properties.xml` using Windows Vista and later (on my Windows 7 system, this is `%HOME%\AppData\Local\IrMaster\IrMaster.properties.xml`), otherwise in `%APPDATA%\IrMaster\IrMaster.properties.xml`. Using other operating systems, it is stored under `$HOME/.irmaster.properties.xml`. It is not deleted by uninstall. (If weird problems appear when updating, try deleting this file.)

3.2 Windows

Download the Window setup file, save, and double click. Accept the license. Select any installation directory you like; suggested is `C:\Program Files\IrMaster`. Unless reason to do so, create the start menu folder, the desktop icon, and allow the program to add the application directory to your path (for IrpMaster as command line program). Administrator rights are probably needed, at least if you are installing in a directory like `Program Files`. (The *should* not be needed otherwise, but Windows Vista and later are always good for a surprise...) IrMaster can now be started from `Start -> IrMaster -> IrMaster`, or from the desktop icon.

To uninstall, select the uninstall option from the Start menu. Very pedantic people may like to delete the properties file too, see above.

3.3 MacOSX

Download and double click the binary distribution. Unpack it to a directory of your choice, e.g. on the desktop. Just double clicking the file IrMaster.jar should now start the program. Otherwise, try the "Other systems" instructions and adapt the wrapper irmaster.sh. This also includes invoking as command line program "IrpMaster".

3.4 Other systems (Linux etc)

For some reason, double clicking an executable jar file in my Gnome installation does not start the program, but starts a browser for the jar file (which is really a form of Zip-Archieve). Instead:

Create an installation directory (suggestion; `/usr/local/irmaster`), and unpack the current binary distribution therein. Examine the wrapper `irmaster.sh`, and, if desired, make desired changes to it with your favorite text editor. Then make two symbolic links from a directory in the path (suggestion; `/usr/local/bin` to the newly installed `irmaster.sh`, using the names `irmaster` and `irpmaster`. Example (using the suggested directories)

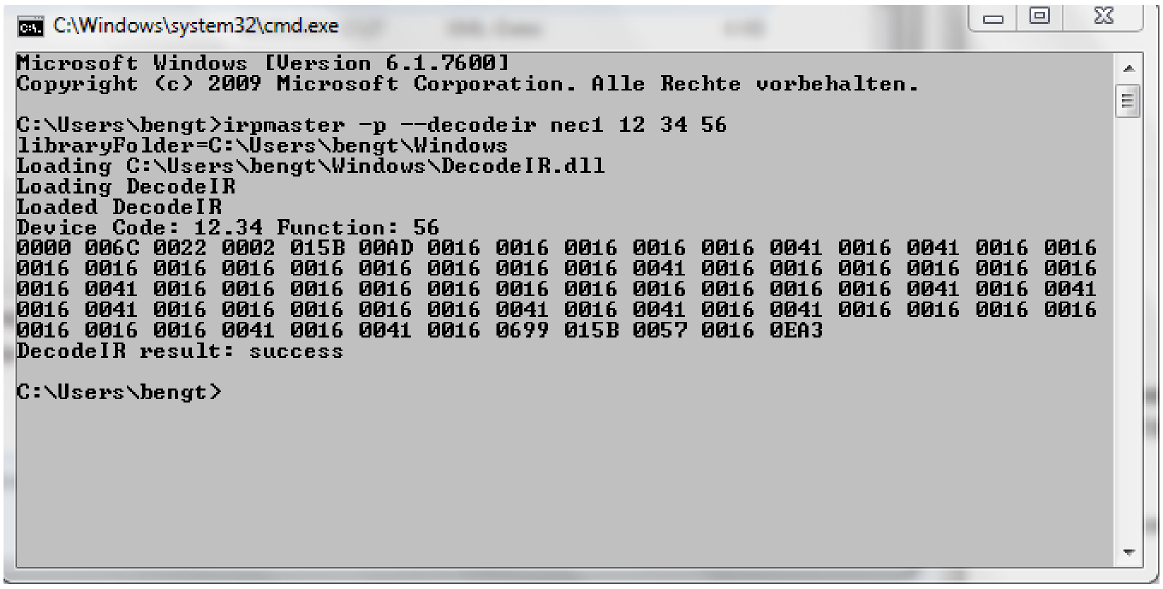
```
cd /usr/local/bin
ln -s ../irmaster/irmaster.sh irmaster
ln -s ../irmaster/irmaster.sh irpmaster
```

(`su` (or `sudo`) may be necessary to install in the desired locations.)

To uninstall, just delete the files. Very pedantic people may like to delete the properties file too, see above.

3.5 Wrapper for IRPMaster

Both under Windows and Unix-like systems, a wrapper for the command line program IrpMaster is installed. The user can simply open a command window (called anything like "xterm", "Terminal", "Shell", "DOS-Box", "cmd",...) and in that command window can call the program IrpMaster by simply typing `irpmaster`, followed by its arguments. See the screen shot below, that shows the generation of the Pronto form of a NEC1 signal for device 12, subdevice 34, command number 56, together with subsequent invocation of DecodeIR on the computed result. (The output on non-windows system is entirely similar.)



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\bengt>irpmaster -p --decodeir nec1 12 34 56
libraryFolder=C:\Users\bengt\Windows
Loading C:\Users\bengt\Windows\DecodeIR.dll
Loading DecodeIR
Loaded DecodeIR
Device Code: 12.34 Function: 56
0000 006C 0022 0002 015B 00AD 0016 0016 0016 0016 0016 0041 0016 0041 0016 0016
0016 0016 0016 0016 0016 0016 0016 0016 0016 0041 0016 0016 0016 0016 0016 0016
0016 0041 0016 0016 0016 0016 0016 0016 0016 0016 0016 0016 0016 0041 0016 0041
0016 0041 0016 0016 0016 0016 0016 0041 0016 0041 0016 0041 0016 0016 0016 0016
0016 0016 0016 0041 0016 0041 0016 0699 015B 0057 0016 0EA3
DecodeIR result: success

C:\Users\bengt>
```

4 Usage

As stated previously, for anyone familiar with the problem domain, this program is believed to be easy to use. Almost all user interface elements have toolhelp texts. In what follows, we will not attempt to explain every detail of the user interface, but rather concentrate on the concepts. Furthermore, it is possible that new elements and functionality has been implemented since the documentation was written.

This program does not disturb the user with a number of annoying, often modal, pop ups, but directs errors, warnings, and status outputs to the *console window*, taking up the lower third of the main window. Starting with version 0.2.0, this window is resizeable. There is a context menu for the console, accessible by pressing the right mouse button in it.

In the upper row, there are four pull-down menus, named File, Edit, Options, and Help. Their usage is believed to be self explanatory, with the exception of the entries in the Options menu. The latter mimics the Options subpane, and are explained later.

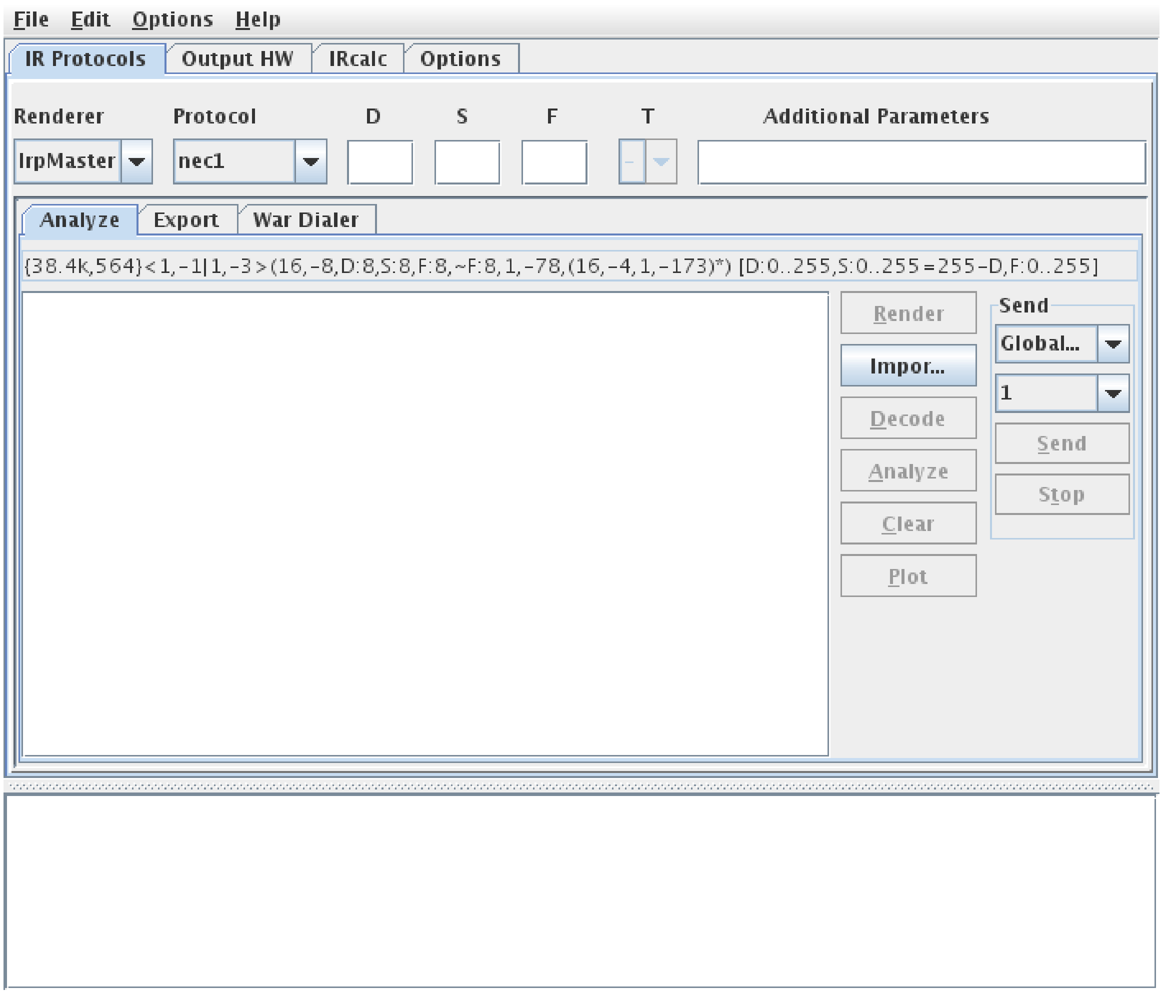
The main window is composed of presently two sub panes denoted by "IR Protocols" and "Hardware" respectively. These will be discussed now.

4.1 The "IR Protocols" pane

In the upper third of this pane, a render program (IrpMaster or Makehex) can be selected, together with the IR protocol identified by name, and the parameters D ("device", in almost all protocols), S ("sub-device", not in all protocols), F ("function", also called command number or obc, present in almost all protocols), as well as "T", toggle (in general 0 or 1, only in a few protocols). These number can be entered as decimal numbers, or, by prepending "0x", as hexadecimal numbers. Note that the supported protocols are different between the different rendering engines. Not all possibilities of IrMaster are available when using the simpler render Makehex.

The lower two thirds of the window is occupied by another pane setup, consisting of the sub-panes "Analyze", "Export", and "War dialer".

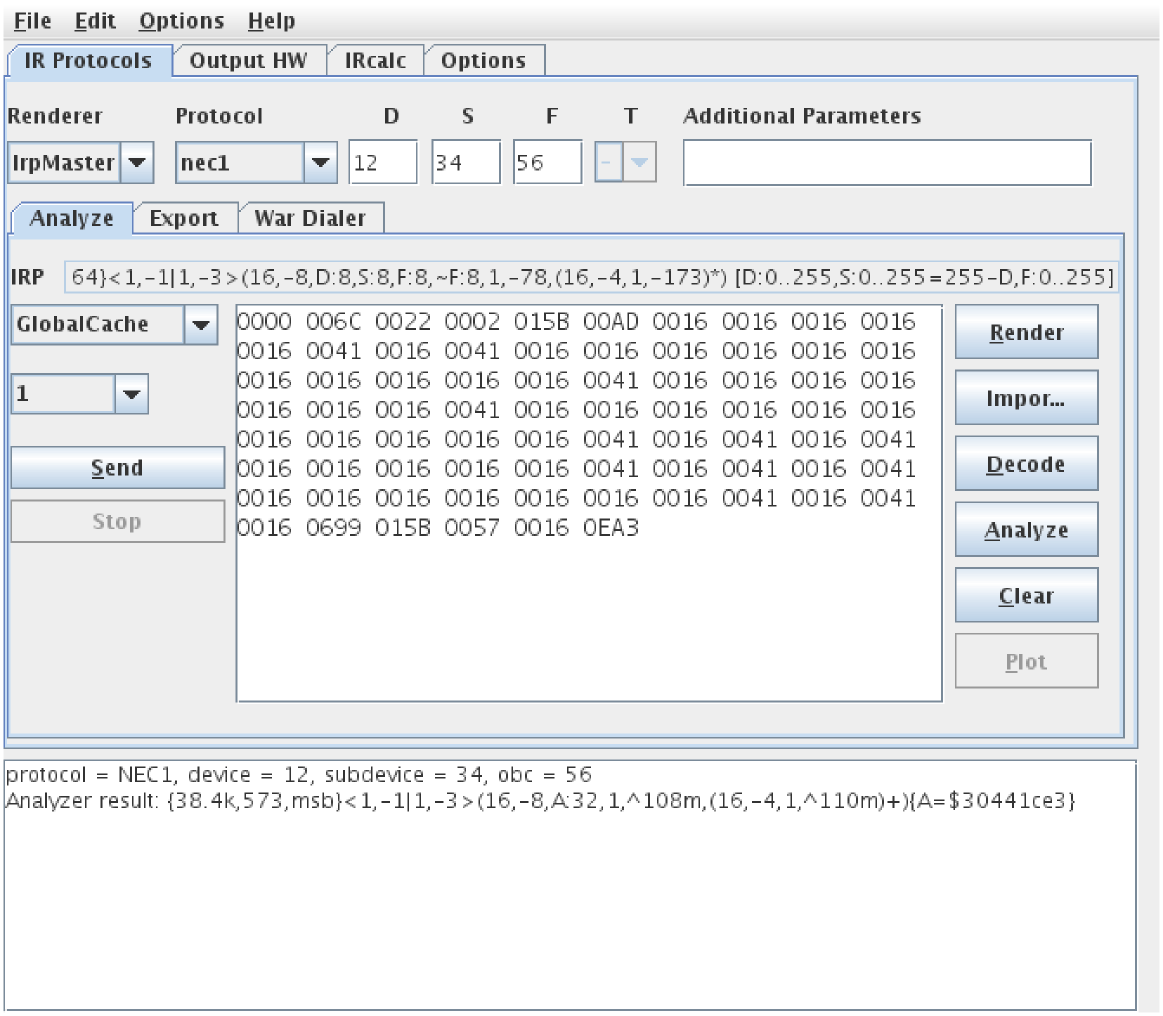
4.1.1 The "Analyze" pane



Renderer: IrpMaster
Protocol: nec1
Send: Global... 1

```
{38.4k,564}<1,-1|1,-3>(16,-8,D:8,S:8,F:8,~F:8,1,-78,(16,-4,1,-173)*) [D:0..255,S:0..255=255-D,F:0..255]
```

By pressing "Render", the signal is computed, and the middle window is filled with the Pronto representation of it. Pressing "Decode" sends the Pronto representation to DecodeIR. Pressing "Analyze" sends it to AnalyzeIR. In both cases, the programs will send their output to the console window, as can be seen below.



Renderer: IrpMaster
Protocol: nec1
D: 12
S: 34
F: 56
T: -
Additional Parameters:

IRP 64}<1,-1|1,-3>(16,-8,D:8,S:8,F:8,~F:8,1,-78,(16,-4,1,-173)*) [D:0..255,S:0..255=255-D,F:0..255]

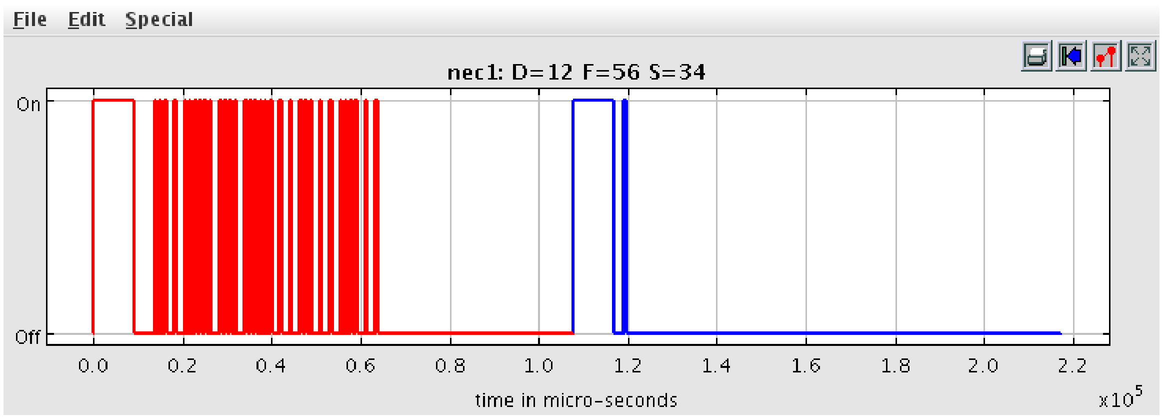
GlobalCache
1

```
0000 006C 0022 0002 015B 00AD 0016 0016 0016 0016
0016 0041 0016 0041 0016 0016 0016 0016 0016 0016
0016 0016 0016 0016 0016 0041 0016 0016 0016 0016
0016 0016 0016 0041 0016 0016 0016 0016 0016 0016
0016 0016 0016 0016 0016 0041 0016 0041 0016 0041
0016 0016 0016 0016 0016 0041 0016 0041 0016 0041
0016 0016 0016 0016 0016 0016 0016 0041 0016 0041
0016 0699 015B 0057 0016 0EA3
```

```
protocol = NEC1, device = 12, subdevice = 34, obc = 56
Analyzer result: {38.4k,573,msb}<1,-1|1,-3>(16,-8,A:32,1,^108m,(16,-4,1,^110m)+){A=$30441ce3}
```

Using context menus, the result can be sent to the clipboard or saved to a file, after invoking a file selector. It is also possible to fill the code window by pasting from the clipboard. Pressing the "Import" button allows to import to import a wave file (see this for a background) or a file in ict-format, for example from the IRScope-Program. The imported IR sequence can be subsequently Decode-d, Plot-ted, and Analyze-d, etc. Note that an ICT file produce by IRScope may contain several IR signals, which cannot be easily separated. They will be imported as one giant signals, with all the gaps and spaces concatenated together. This is a flaw in the IRScope program.

By pressing the "Plot" button, either an IR signal in the CCF window, or a newly rendered signal corresponding to the selected parameters, is shown in a popup plot window, using the PtPlot library.



Using its own controls, the plot can be zoomed (press left button and drag), printed, or exported to encapsulated PostScript. Once created, it cannot be changed (other than zooming, resizing etc), just closed. However, an "unlimited" number of such popup windows are possible.

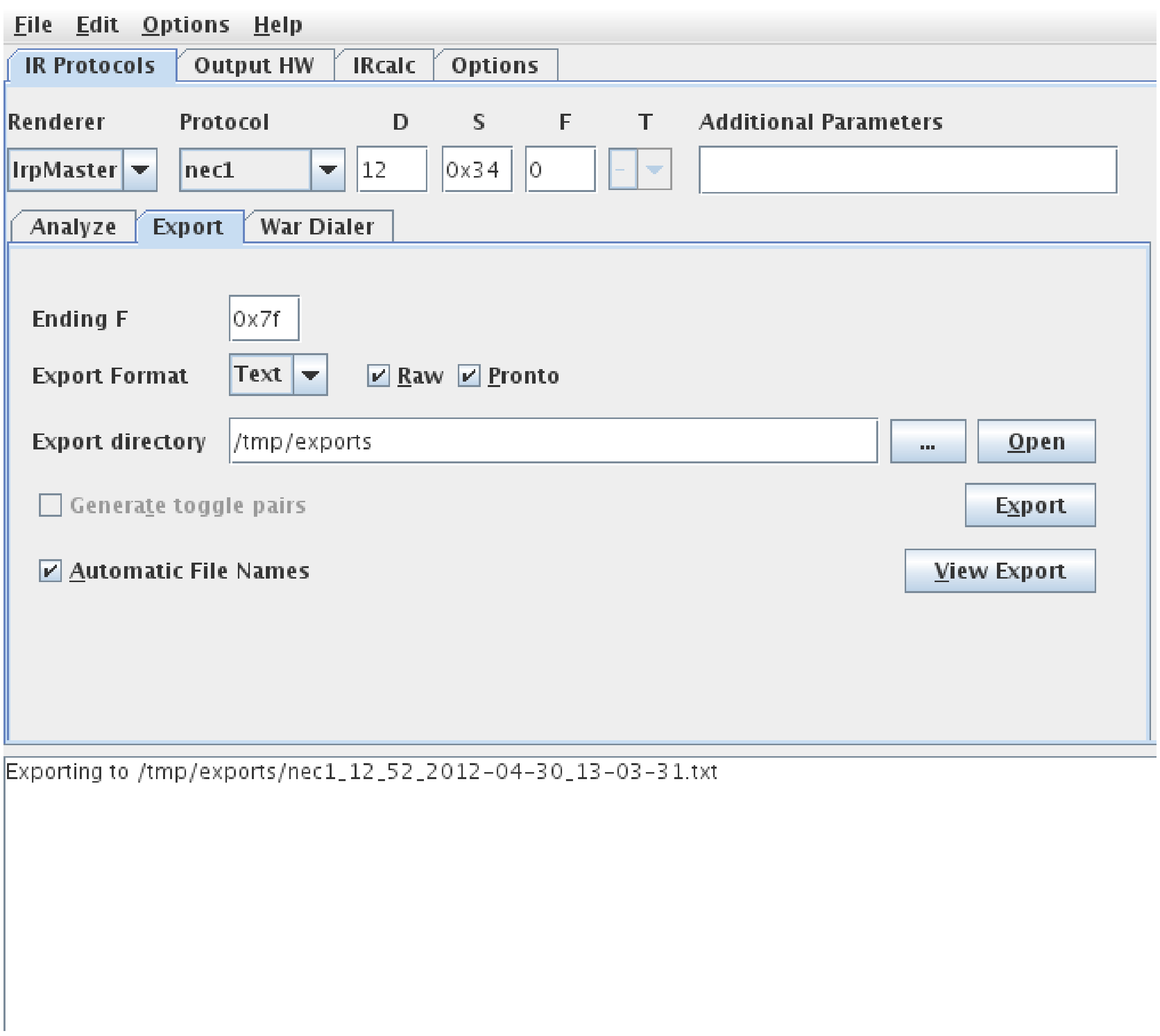
The right part of the middle window allows for sending the code in the code window to hardware, presently GlobalCaché, IRTrans, or LIRC (requires a patch), or an audio-IR-setup, any number of times the user desires. These run in their own threads, and can be stopped anytime.

In all cases, if the CCF window is non-empty and starts with "0000", DecodeIR/Analyze operates on the actual Pronto code, which may even be manually manipulated by the user. If it start with a "+"-character, it is attempted to interpret it as a "raw" signal, consisting of a number of gaps in pulses, given in microseconds. If it consists of a number of hexadecimal two-digit numbers, it is attempted to interpret the signal as a UEI learned signal.

If the window is empty, new signal is rendered according to the parameters, and subsequently used for sending or plotting. There is thus no need to "render" before plotting or sending.

In rare cases, transforming the signal to the Pronto format may introduce some rounding errors causing DecodeIR to fail to indicate some IR signals it would otherwise identify.

4.1.2 The Export pane



Renderer: IrpMaster
Protocol: nec1
D: 12
S: 0x34
F: 0
T: -
Additional Parameters:

Ending F: 0x7f
Export Format: Text ☑ Raw ☑ Pronto
Export directory: /tmp/exports
☐ Generate toggle pairs
☑ Automatic File Names

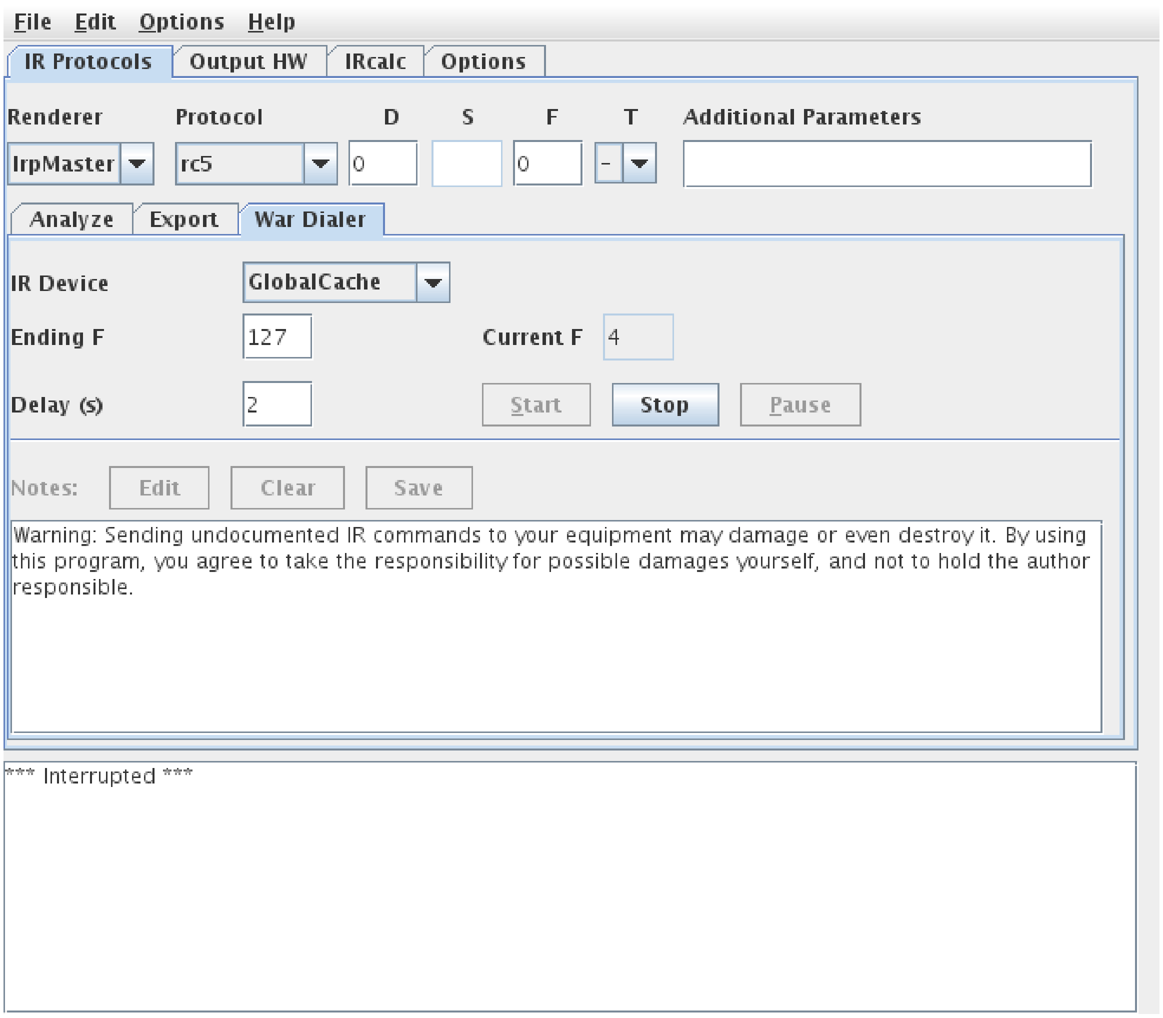
```
Exporting to /tmp/exports/nec1_12_52_2012-04-30_13-03-31.txt
```

Using the export pane, export files can be generated. These allow e.g. other programs to use the computed results. Using IrpMaster as rendering engine, exports can be generated in text files, XML files, LIRC-format, Lintronic-format, or as Wave-files, the first two both in Pronto format and in "raw" format (timings in microseconds, positive for pulses, negative for gaps). The XML export is intended as a starting point for generate yet other formats, by invoking easily written transformations on it. In this article I demonstrate how to generate C code from it using XSLT transformations. The LIRC-exports are in lirc.conf-format using the raw format. They can be concatenated together and used as the LIRC server data base, typically `/etc/lirc/lircd.conf`. Of courses, they can also be used with WinLirc. For the wave-export, parameters are "inherited" from the Output HW/Audio pane. Options for the wave-export, as well as some of its usage, is explained there. Optionally, for protocols with toggles, both toggle pairs may optionally be included in the export file by selecting the "Generate toggle pairs"-option. Export file names are either user selected from a file selector, or, if "Automatic file names" has been selected, automatically generated.

The export is performed by pressing the "Export" button. The "..."-marked button allows for manually selecting the export directory. It is recommended to create a new, empty directory for the exports. The just-created export file can be immediately inspected by pressing the "View Export"-button, which will open it in the "standard way" of the used operating system. The "Open" button similarly opens the operating systems standard directory browser (Windows Explorer, Konquistador, Nautilus,...) on the export directory.

The export formats Wave and Lintronic export an IR sequence rather than an IR signal (consisting of an intro sequence, an repetition sequence (to be included 0 or more times), and an (most often empty) ending sequence). Therefore, using the Wave and Lintronic formats, the number of repetition sequences to include can be selected.

4.1.3 The "War Dialer" pane



Renderer: IrpMaster
Protocol: rc5
D: 0
S:
F: 0
T: -
Additional Parameters:

IR Device: GlobalCache
Ending F: 127
Current F: 4
Delay (s): 2

Warning: Sending undocumented IR commands to your equipment may damage or even destroy it. By using this program, you agree to take the responsibility for possible damages yourself, and not to hold the author responsible.

```
*** Interrupted ***
```

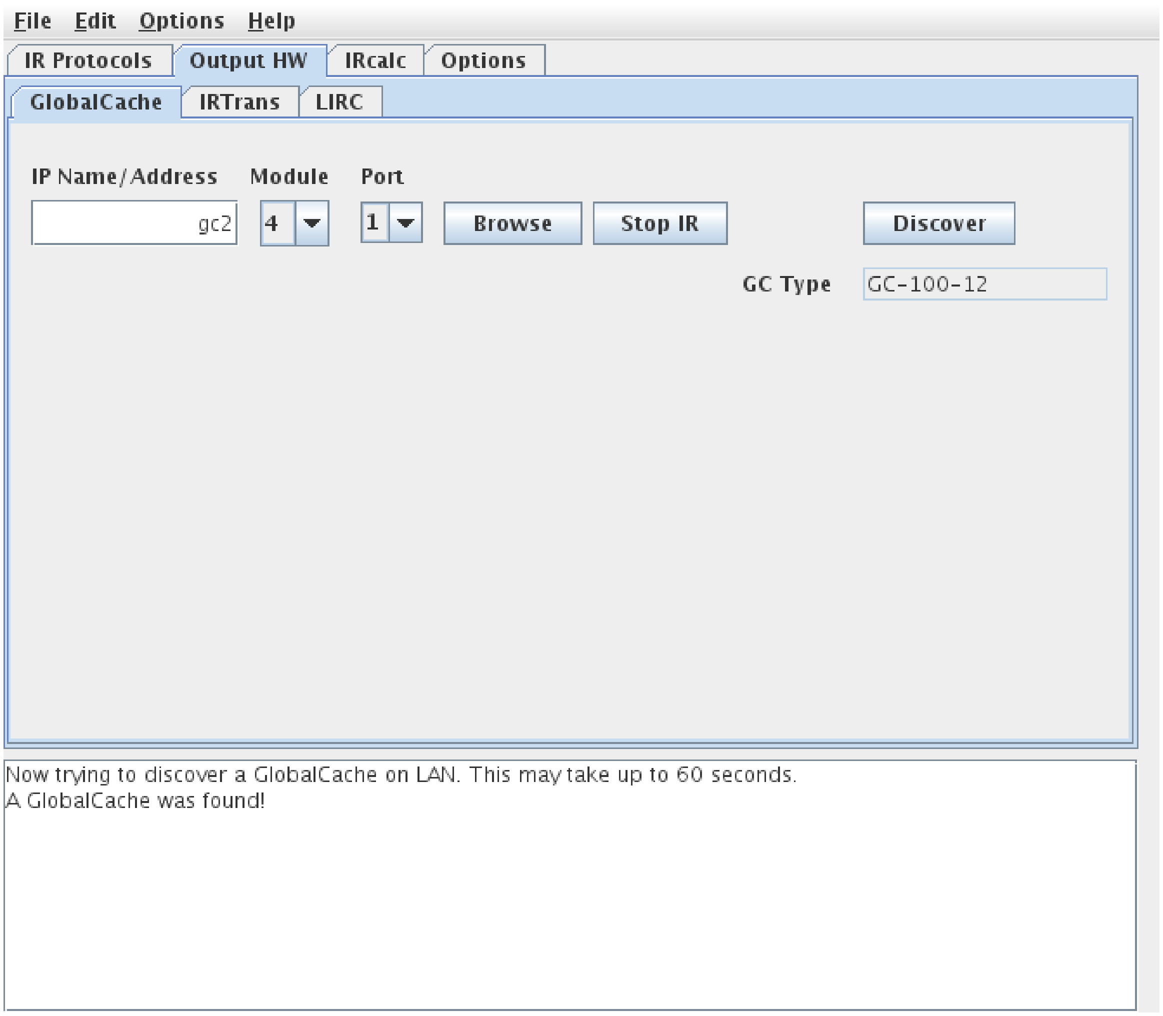
This functionality is intended for the search for undocumented IR commands for customer equipment. It allows for sending a whole interval of IR signals to the equipment, and taking notes when something reacts on the sent signal. The "Start" and "Stop" functions are probably self explaining; the "Pause" button allows for interrupting and later resuming, but is presently not implemented. A note-taking function is planned but presently not implemented: when "Edit" is pressed, a "Document" pops up with current IR signal and time, allowing the user to write a note on that signal, which can later be saved by "Save".

4.2 The "Hardware" pane

The sub-panes of this pane allows for the selection and configuration of the employed IR hardware.

4.2.1 The "GlobalCache" pane.

This allows for configuring GlobalCaché support. The Discover-button attempts to find the identifying beacon of GlobalCaché modules (only present on recent firmware). If successful, will fill in the IP-Box, its model, the default IR-module and IR-port (see the GlobalCaché API specification for the exact meaning of these terms). In any case, the IP Name/address window, the module and port selection can be changed manually. The Browse-button directs the selected browser to the built-in WWW-server of the module, while the "Stop IR"-Button allows the interruption of ongoing transmission, possibly initiated from another source.



IP Name/Address: gc2
Module: 4
Port: 1
GC Type: GC-100-12

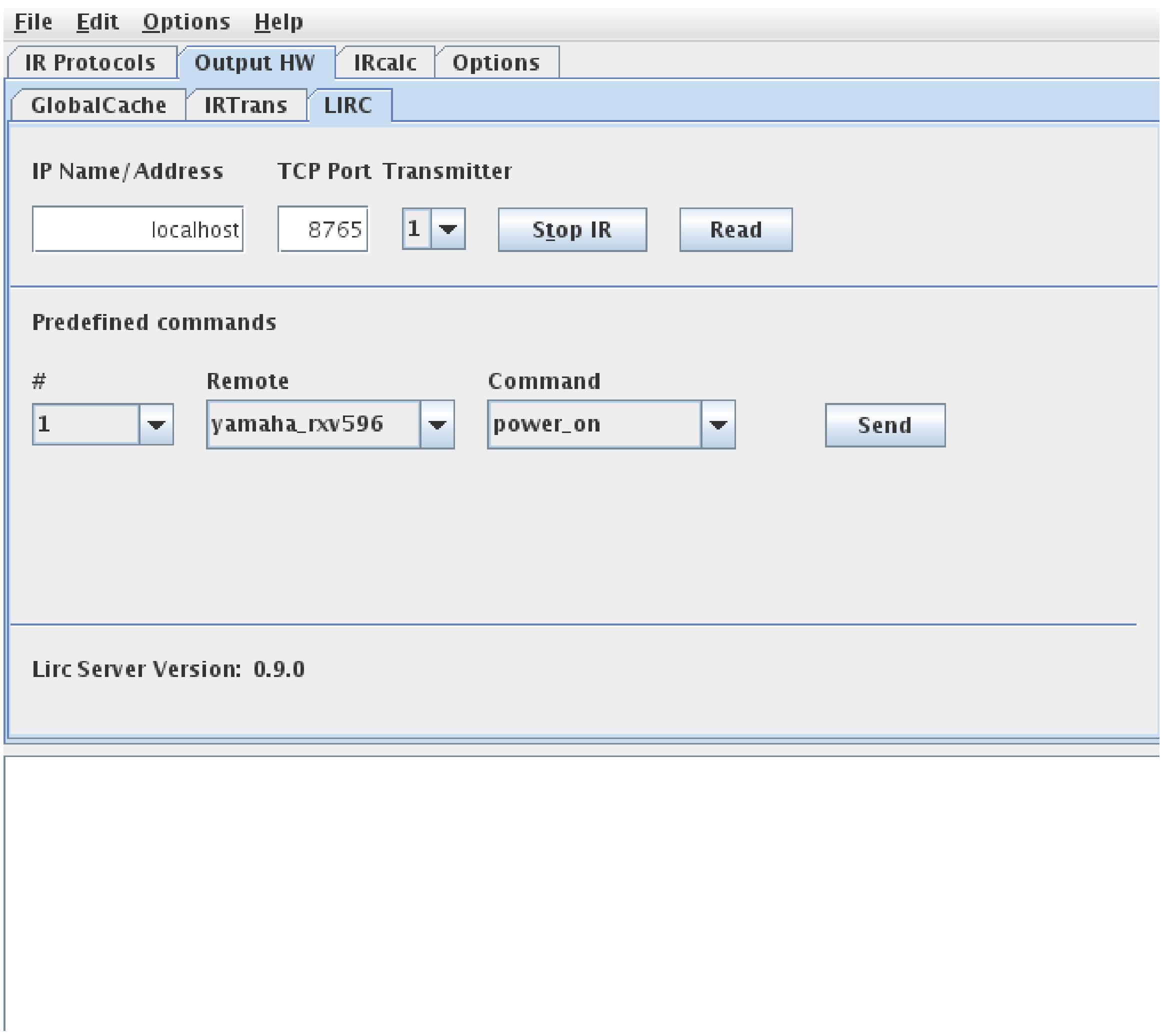
```
Now trying to discover a GlobalCache on LAN. This may take up to 60 seconds.
A GlobalCache was found!
```

4.2.2 The "LIRC" pane

To be fully usable for IrMaster, the LIRC server has to be extended to be able to cope with CCF signal not residing in the local data base, but sent from a client like IrMaster, thus mimicing the function of e.g. a GlobalCaché. The needed modification ("patch") is in detail described here. However, even without this patch, the configuration page can be used to send the predefined commands (i.e. residing it its data base `lirc.conf`). It can be considered as a GUI version of the `irsend` command.

The LIRC server needs to be started in network listening mode with the `-l` or `--listen` option. Default TCP port is 8765.

After entering IP-Address or name, and port (stay with 8765 unless a reason to do otherwise), press the "Read" button. This will query the LIRC server for its version (to replace the grayed out "<unknown>" of the virgin IrMaster), and its known remotes and their commands. Thus, the "Remote" and "Command" combo boxes should now be selectable. After selecting a remote and one of its command, it can be sent to the LIRC server by pressing the "Send" button. If (and only if) the LIRC server has the above described patch applied, selecting "LIRC" on the "Analyze" and "War Dialer" panes now works.



IP Name/Address: localhost
TCP Port: 8765
Transmitter: 1

Predefined commands

#: 1
Remote: yamaha_rxv596
Command: power_on

Lirc Server Version: 0.9.0

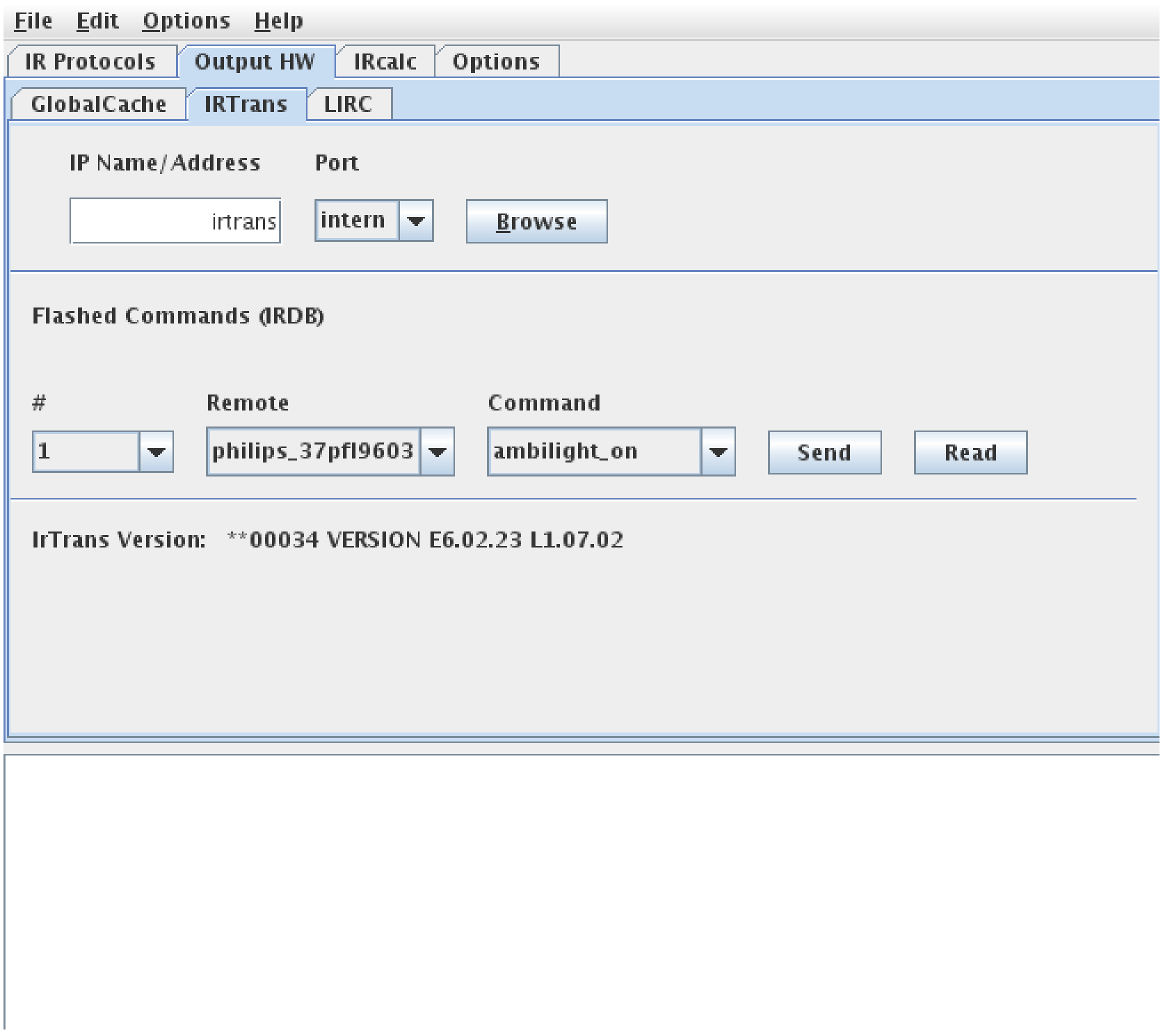
Due to LIRC's pecular form of API stop command, the "Stop IR" command presently does not work. See this thread in the LIRC mailing list for a background.

4.2.3 The "IRTrans" pane

The configuration of IRTrans is similar to LIRC, so it will be described more briefly.

Enter IP name or -address and select an IR port (default "intern"). If the Ethernet IRTrans contains an "IR Database" (which is a slightly misleading term for an internal flash memory, that can be filled by the user), its commands can be sent from this pane. By pressing the "Read" button, the known remotes and commands are loaded, and the version of the IRTrans displayed. The selected command can now be sent by the "Send" button. (However, this functionality is otherwise not used by IrMaster.) Selecting

"IRTrans" on the "Analyze" and "War dialer" pane should now work. The IRTrans module is then accessed using the UDP text mode.



IP Name/Address: irtrans
Port: intern

Flashed Commands (IRDB)

#: 1
Remote: philips_37pfl9603
Command: ambilight_on

IrTrans Version: **00034 VERSION E6.02.23 L1.07.02

4.2.4 The "Audio" Pane

As additional hardware device, IrMaster can generate wave files, that can be used to control IR-LEDs. This technique has been described many times in the internet the last few years, see for example this page within the LIRC project. The hardware consists of a pair of anti-paralell IR-LEDs, preferably in series with a resistor. Theoretically, this corresponds to a full wave rectification of a sine wave. Taking advantage of the fact that the LEDs are conducting only for a the time when the forward voltage exceeds a certain threshold, it is easy to see that this will generate an on/off signal with the double frequency of the original sine wave. (See the first picture in the LIRC article for a picture.) Thus, a IR carrier of 38kHz (which is fairly typical) can be generated through a 19kHz audio signal, which is (as opposed to 38kHz) within the realm of medium quality sound equipment, for example using mobile devices.

IrMaster can generate these audio signals as wave files, which can be exported from the export pane, or sent to the local computers sound card. There are some settings available: Sample frequency (42100, 48000, 96000, 192000Hz), sample size (8 or 16 bits) can be selected. Also "stereo" files can be generated by selecting the number of channels to be 2. The use of this feature is somewhat limited: it just generates another channel in opposite phase to the first one, for hooking up the IR LEDs to the difference signal between the left and the right channel. This will buy you double amplitude (6 dB) at the cost of doubling the file sizes. If the possibility exists, it is better to turn up the volume instead.

Data can be generated in little-endian (default) or big-endian format. This applies only to 16-bit sample sizes.

As an experimental option, the carrier frequency division as described above can be turned off (the "Divide carrier" checkbox). This is only meaningful for sample frequencies of 96kHz and higher, and for "audio equipment" able to reproduce frequencies like 36kHz and above.

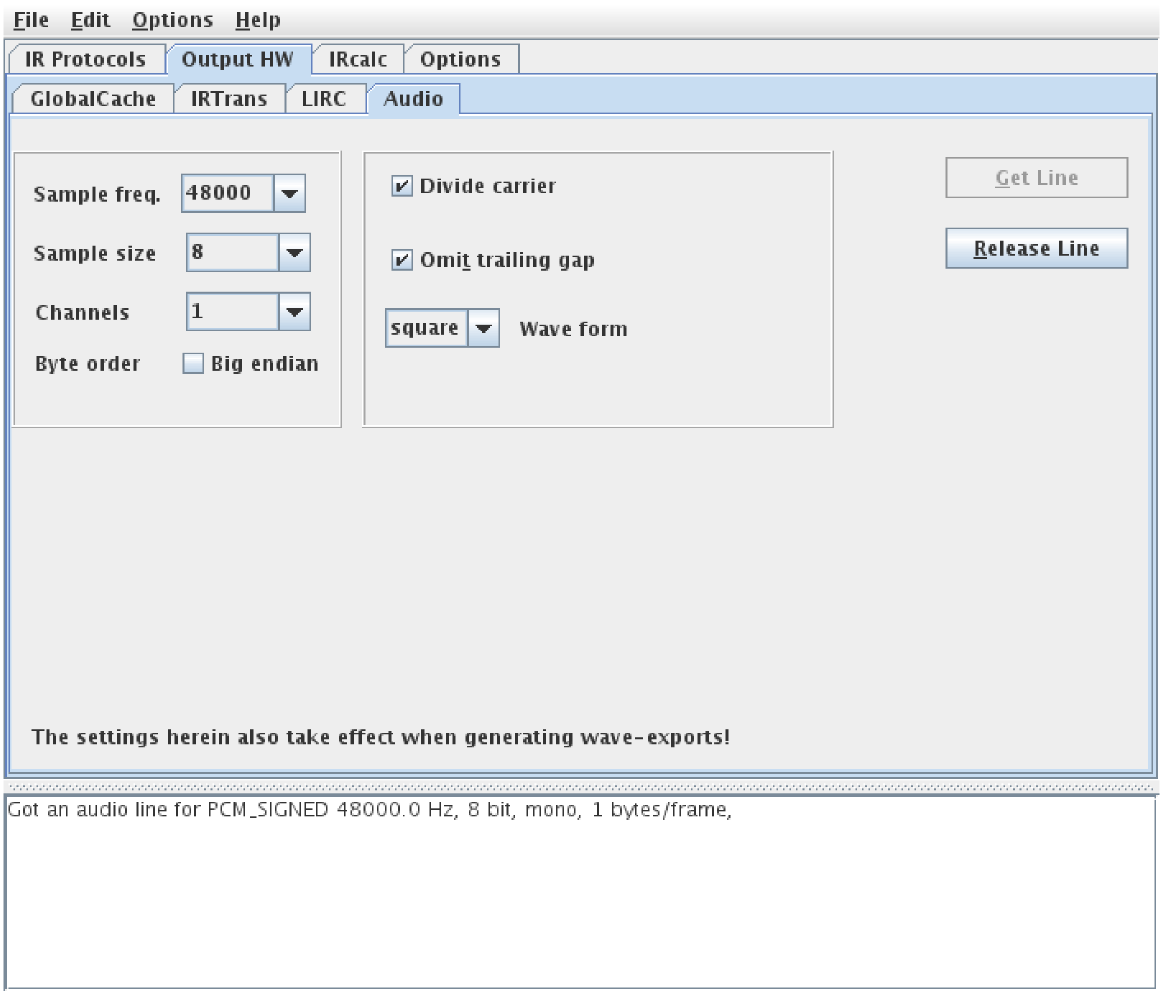
Most of "our" IR sequences ends with a period of silence almost for the half of the total duration. By selecting the "Omit trailing gap"-option, this trailing gap is left out of the generated data -- it is just silence anyhow. This is probably a good choice (almost) always.

Finally, the wave form on the modulation signal can be selected to either sine or square wave. For practical usage, my experiments shown no real performance difference.

Note that when listening to music, higher sample rates, wider sample sizes, and more channels sound better (in general). However, generating "audio" for IR-LEDs is a completely different use case. The recommended settings are: 48000kHz, 8bit, 1 channel, divide carrier, omit trailing gap, and square wave form.

Note that the settings on this pane also take effect when exporting wave files from the export pane.

By pressing "Get Line" a "line" to the audio system on the local computer is allocated. This is actually superflous, since the send-functions make this automatically anyhow. It will possibly be removed in future versions.



Sample freq. 48000

Sample size 8

Channels 1

Byte order ☐ Big endian

☑ Divide carrier

☑ Omit trailing gap

square Wave form

The settings herein also take effect when generating wave-exports!

```
Got an audio line for PCM_SIGNED 48000.0 Hz, 8 bit, mono, 1 bytes/frame,
```

4.3 Command line arguments

Normal usage is just to double click on the jar-file, or possibly on some wrapper invoking that jar file. However, there are some command line arguments that can be useful either if invoking from the command line, or in writing wrappers, or when configuring custom commands in Windows.

```
Usage:
        irmaster [-v|--verbose] [-d|--debug debugcode] [-p|--properties propertyfile]
 [--version|--help]
or
        irmaster IrpMaster <IrpMaster-options-and-arguments>
```

The options `--version` and `--help` work as they are expected to work in the GNU coding standards for command line interfaces: The `-v/--verbose` option set the verbose flag, causing commands like sending to IR hardware printing some messages in the console. The debug option `-d/--debug` takes an argument, and the result is stuffed into the debug parameter in the program. This is passed to invoked programs that are free to interpret it any way it likes. For example, here is how IrpMaster interprets the debug variable. This option is likely of interest only to developers, who have to look in the code to see what is sensible.

The second form invokes IrpMaster as a the command line program on the rest of the arguments. It is a convenience feature for just having one user entry point in the distributed jar file.

For automating tasks, or for integrating in build processes or Makefiles or the like, it is probably a better idea to use IrpMaster instead, which has a reasonably complete command line interface.

The program delivers well defined and sensible exit codes.

5 References

1. IrpMaster. Also a GPL3-project by myself. Much harder to read than the present document :-). See also this discussion thread in the JP1 forum.
2. The Harctoolbox project, also a GPL3-project by myself. It is used for the interface to GlobalCaché and IrTrans, as well as some minor support routines, that I did not feel for duplicating.
3. DecodeIR. This shared library tries to identify protocol name and parameters of an IR signal in raw form. Thus, it is in a sense, it implements the "inverse mapping" of IrpMaster.
4. GlobalCaché, a manufacturer of Ethernet connected IR hardware. Note that I have only tried with the GC-100 series, but the IR sending models of theiTach family are believed to work too. (Feel free to send me one :-).)
5. IRTrans, another manufacturer of Ethernet connected IR-hardware. The "IRTrans Ethernet" module, preferably with "IRDB Option" (internal flash memory), is directly supported by the current software.
6. LIRC, Linux InfraRed Control This project contain drivers for almost everything IR-related. The present project is able to use a modified LIRC-server for transmitting IR signals.