A TR log linux port

Kevin E. Schmidt, W9CF
6510 S. Roosevelt St.
Tempe, AZ 85283 USA

Contents

1 Introduction

Read the standard legal disclaimer in section 25.

TR log or TR was written by Tree, Larry Tyree, N6TR. It was written for PCs running DOS. It continued to work fine running versions of Windows based on DOS, i.e. Windows 95, 98, ME etc., but does not run without modifications under modern multitasking multiuser versions of Windows that are based on Windows NT, and obviously will not run natively on unix based operating systems like linux or Mac OS X.

For many years I have run TRlog under linux using dosemu, a program that runs a version of DOS in linux. While this works for the most part, the timing for sending CW requires taking care that dosemu gets high priority, and even so occasionally poor cw is sent. Getting a sound card working is difficult, and the whole thing is ugly.

In November 2011, I asked Tree if he would be willing to let me try to port TR log to linux. He very generously sent me the complete source code to TR log version 6.76, along with the comment "I suppose I can zip up the source code – but I think you are nuts!"

TR log is written in Pascal. In particular, I learned that it is written in Borland Pascal 7.0. Happily, Free Pascal, available at `<http://freepascal.org>`, is an excellent multiplatform pascal compiler that supports nearly all the Borland Pascal 7.0 features. This made porting the code much less painful.

2 Goal

Please realize, that my goal was to port the code to run on modern hardware with a multitasking multiuser operating system while doing the least amount of work. Therefore I decided to run TR log in a terminal window that would look and work like the DOS version of TR.

I have added few new features. I have tried to remove as few features of the DOS based TR log as possible.

3 My tests

I have tested this code only on my linux systems which all run a recent version of Slackware, mostly Slackware 13.37.0, and all run the fairly lightweight xfce desktop environment. There are many features that I have not tested. Most of these should work since I have not changed Tree's code intentionally.

4 Quick start

1. Download the latest trlinux-rN.NN.txz, where the N.NN is the version number. The TR linux web site is `<http://www.kkn.net/trlinux/>`,

2. Untar this in a convenient location at the command line

   ```
   tar -xJf trlinux-rN.NN.txz
   ```

3. In the trlinux-rN.NN/files subdirectory find the Xresources file. Include these lines in the .Xresources file in your $HOME directory, (or create this file if it doesn't exist), for example type:

   ```
   cd trlinux_rN.NN/files
   cat Xresources >> $(HOME)/.Xresources
   ```

   To make these changes take effect during this X-windows session you can run at the command line

```
cd
xrdb -merge .Xresources
```

It will be automatically reloaded when you log in.

4. Test that this works with the commands

   ```
   xterm -class DosTerm
   ```

   and

   ```
   xterm -class BigDosTerm
   ```

   Feel free to change the font specified in the .Xresources file for these to get a convenient size. A list of font names you can try is given by typing

   ```
   fc-list :scalable=true:spacing=mono: family
   ```

   It is absolutely necessary that the DosTerm be 80 columns and 25 rows, and the BigDosTerm 80 columns and 50 rows. You can check this with the commands

   ```
   set |grep COLUMNS
   set |grep LINES
   ```

   the first should print COLUMNS=80 and the second LINES=25 or LINES=50 depending on whether you are in a DosTerm or a BigDosTerm. If these do not match, you likely have a font that is too big to allow the xterm its full size. In that case change the faceSize value in the .Xresources file.

5. All of the files that you need to run TRlog are in the subdirectory trlinux-rn.nn/log. Place these in any convenient directory (or leave them where they are). I will call this directory path-to-trlog. The executable name is then

   ```
   path-to-trlog/trlog
   ```

   since **tr** is already a system utility in linux/unix. Feel free to put the directory path-to-trlog in your PATH environment variable to reduce typing.

6. Start a DosTerm with

   ```
   xterm -class DosTerm
   ```

   Change to a convenient empty directory, and start Trlog, for example

   ```
   cd
   mkdir trtest1
   cd trtest1
   path-to-trlog/trlog
   ```

You should get the prompt

```
Do you want to step through setting up a new contest? (Y/N) :
```

answer Y, and choose a contest – try CQWW to test things out. Give your call when prompted, and follow the directions more or less as in DOS TR log to do a basic set up.

7. You should have a trlog screen in the xterm. Try logging a few contacts.

8. If this works you're essentially ready to go. See the later sections for more details, and how to set up a multi network if you want to connect more than one instance of trlog together.

5 Setting up TR log on linux

TR under linux should be run as a normal user. I run it under my usual login account. That user/account should have permission to read from and write to any devices you use to interface to your rig.

Since linux is case sensitive, Tree wrote the file names in the source in all capitals, and all the commands in the LOGCFG.DAT file are converted to uppercase before interpreting them, all file names used by TR should be in capital letters. So, for example, the default file names will be LOGCFG.DAT, LOG.DAT, LOG.TMP, RESTART.BIN, BANDMAP.BIN etc.

6 Window manager hot keys

Most window managers grab various function key presses before they are sent to applications. Since TRlog uses all the combinations of function, ctrl-function, alt-function, shift-function, you should turn off the window manager hot keys (or at least the ones you want to use in TRlog). How to do this varies according to the window manager you run. For example for xfce go to Settings->window manager->Keyboard, and clear the shortcuts that you want to use under TRlog. In most window managers, you can set up different profiles so that you can have one profile for normal computing when you want the hot keys to work, and another when you want to run TRlog.

7 PC speaker

Under linux, the PC speaker is accessed through /dev/console which, for security reasons, can only be read or written by root. If you want to use the PC speaker for beeps like in DOS, you must run the linux trlog executable as the root user. As an alternative, you can use the beep soundcard enable command described later. Nothing else should require root permissions!

8 New configuration commands

The main change to the LOGCFG.DAT is that since the program now goes through the standard drivers, the serial and parallel port devices must use linux device names. Whenever a port number was given in the DOS TR configuration command, a full device path needs to be given under linux. *These are case sensitive and must match the linux device names.* Here is an example of the lines that I use for radio control and keying of my Elecraft K2 using a usb serial port adapter. The adapter comes up under linux as the device /dev/ttyUSB0.

```
keyer output port = serial /dev/ttyUSB0
radio one type = k2
radio one control port = serial /dev/ttyUSB0
radio one baud rate = 4800
```

A similar set up on my shack computer which has hardware serial and parallel ports uses:

```
keyer output port = parallel /dev/parport0
radio one type = k2
radio one control port = serial /dev/ttyS0
radio one baud rate = 4800
paddle input = parallel /dev/parport0
```

The serial port device names are those set up by the kernel. The parallel port names are the device names reported by libieee1284. Usually these will be /dev/parport0, /dev/parport1, etc.

9 Port permissions

You will need appropriate permission to read and write to the ports. The simplest way is to add the username of the account to the groups that own the ports. For example, when plugging in a USB to serial port adapter on my Slackware 13.37.0 system, the device is /dev/ttyUSB0 which has permissions from ls -l /dev/ttyUSB0

```
crw-rw---- 1 root dialout 188, 0 Feb 22 22:12 /dev/ttyUSB0
```

which is owned by the dialout group. I added my user name (w9cf) to that entry in /etc/group using the command

```
sudo gpasswd -a w9cf dialout
```

You must log out and log back in for this to take effect.

10 K1EL Winkeyer support

Initial limited support of the K1EL Winkeyer has been added. I have done limited testing with with my Winkeyer2 v2.3 chip. It should support two radios, both with ptt, and a single sidetone if desired.

Older version 1 chips are supported, but since they have only two output pins, I have chosen to use these for keying the two radios. No PTT or sidetone is available with these chips. Since you can order a WinKeyer2 chip from K1EL for 11 US dollars, I decided it was not worth more than a few minutes of my time to support the older chips.

There is a new configuration command. Add a line like:

```
winkeyer port = serial /dev/ttyUSB0
```

to your LOGCFG.DAT file, substituting your serial port device that is connected to the Winkeyer for /dev/ttyUSB0 above.

Most of the standard TR commands should still work. Some things that are currently different or should be mentioned (there are probably others)

- The Winkeyer paddle speed is locked to the trlog speed.
- The speed pot on the Winkeyer is ignored. (This is untested since I do not have a speed pot hooked to my Winkeyer2 v2.3 chip.)
- You cannot have a separate paddle sidetone frequency. (Actually, none of the Winkeyer sidetone stuff is tested since I do not have a speaker connected to my Winkeyer2 v2.3 chip.)
- You cannot change the weight in a message using ctrl-x, ctrl-y.
- Speed changes with ctrl-f and ctrl-s should work fine.
- None of the special length dots and dashes work. That is do not use ctrl-p, ctrl-q, ctrl-\, ctrl-v, ctrl-l, ctrl-e, ctrl-d, ctrl-k ctrl-n, ctrl-o, or - in a message.
- During CW, the Winkeyer handles PTT. It does not handle push to talk in other modes. This can be added if there is sufficient interest. The trlog PTT setting are approximately mapped to Winkeyer settings. So Paddle PTT Hold Count is converted to the nearest Winkeyer hang time. PTT Turn On Delay is converted into the nearest Winkeyer leadin.
- Footswitch CW grant does not work with the Winkeyer. This could be added if there is sufficent interest.
- Winkeyer PTT tail time has been fixed at 20 milliseconds.
- Choosing the simulator will disable the Winkeyer and use Tree's original CPU keyer to generate the simulated CW.

To add Winkeyer support, I encapsulated Tree's serial and parallel port keyer object as a class. This made adding Winkeyer support straightforward, however, be aware that this could easily have added bugs to the original serial and parallel port keyers.

11 Bug fixes and upgrades

As mentioned in the introduction, the code base is TRlog version 6.76. I have fixed one bug and added support for newer CTY.DAT with specific call signs.

- I have fixed the 160m bandmap bug – see `http://lists.contesting.com/_trlog/2003-10/msg00092.html`.

- You should use the CTY.DAT file for TRlog version 6.90 and later which flags individual calls with an equal sign. For example the entry =W9CF(3)[6] shows that I am in CQ zone 3 and ITU zone 6 rather than the default W9 zones 4 and 8. This same entry in the CTY.DAT file for earlier versions is W9CF(3)[6] without the equal sign. The problem with this older version is that W9CF is treated as a prefix and W9CFA through W9CFZ would be flagged as being in zones 3 and 6 too. With the equal sign in front of my call there has to be an exact match.

- The dvp commands can be used to play files. Currently these must be at 44100 Hz stereo, i.e. cd quality audio. They can be wave files, or, if your distribution has compiled ogg support into libsndfile, they can be ogg files. (You can take your wave file temp.wav and encode it with oggenc temp.wav then rename it to CQF1.DVP etc., or just rename temp.wav to CQF1.DVP.) The sound uses the ALSA (advanced linux sound architecture) libraries so the kernel must have ALSA support. By default it opens the hw:0,0 device so the first soundcard will be used. It must support 44100 Hz stereo. You can change the device with the configuration file command like

  ```
  SOUNDCARD DEVICE = plughw:0,0
  ```

  which would open the alsa software plugin for the same device. Push to talk should now work with the dvp commands. It is implemented in all current push to talk methods YCCC SO2R box, Winkeyer, and serial/parallel port push to talk.

- If you are not using the dvp files, you can give the command

  ```
  beep soundcard enable = true
  ```

  which will play the pc speaker beeps through the sound card. Again this requires ALSA and 44100 Hz stereo support. Note that, just like the original pc speaker, no shaping is done, so the clicks when using this as a sidetone or with the simulator get annoying quickly.

- Since parallel ports are getting rare, and USB to parallel adaptors don't work very well for non printer applications, I added paddle input on serial ports. This violates the RS-232 standard which defines the two logic levels as positive and negative voltages, while the paddle will only key a positive voltage to ground. But, according to the ARRL handbook, none of the serial port chips used actually require a negative voltage on their inputs to get the correct logic level. Therefore, I have set up the clear to send (CTS) input as the left paddle (normally dot) input, and data set ready (DSR) input as the right paddle (normally dash) input. I pull these inputs to +12 volts through a 10K ohm resistor (the

RS-232 standard requires between 3 and 25 volts), and the paddle keys those inputs to ground. On a DB9 connector they are pins 6 (DSR) and 8 (CTS). Ground is pin 5. It is then possible to have radio control, radio keying, push to talk, and paddle input on a single serial port. I use a USB serial port with the line

```
paddle port = serial /dev/ttyUSB0
```

in LOGCFG.DAT. Probably the two easiest places to get the pull-up voltage are from the rig power supply or use the +5 volt output of a USB port.

I have recently noticed that using the serial paddle input with USB to serial devices can affect the CW quality. On my brief experiments, a USB to serial converter using an FTDI chip slowed down the CW substantially above about 25 wpm. A different converter using a Prolific chip worked fine for CW up to about 35 wpm.

- Since the serial ports are using the linux drivers, you should be able to use any baud rate supported by your hardware (up to at least 115200 baud for modern serial ports). Since the linux kernel can buffer up to 4096 bytes, the only caveat is that trlog will have a maximum throughput on the serial ports set by the emulated DOS interrupt which occurs about every 1.7 milliseconds. I believe this should be adequate for any rig.

- Winkeyer support has been added.

- Ultimatic mode has been added. This is free with Winkeyer, it now also works with the original CPU keyer. Add

  ```
  curtis keyer mode = u
  ```

  to your LOGCFG.DAT or use the control-j menu.

- The shift key enable command can be true and false as in DOS TR. I have added

  ```
  SHIFT KEY ENABLE = ALTSHIFT
  ```

  In this mode, you need to first hold either alt key before pushing left or right shift to tune the radio or RIT. Shifted characters like the question mark then do not have to be remapped.

12 Things that no longer work

- DVP record/backtrace etc. These can be added, but they are low priority for me.

- DVP PATH has not been implemented. Place the audio files in the current directory with LOGCFG.DAT.

13 Packet using the shell device

The preferred way to use packet is now with the shell device. Use the lines

```
PACKET PORT = SERIAL shell
EIGHT BIT PACKET PORT = TRUE
```

If you now hit control-b, you will get the usual packet window, but it will now show a shell prompt. You can simply type a command like

```
telnet n7az.net
```

login with your call sign, and type sh/dx or just wait for spots. If you have other software that provides packet filtering, etc., you can use that.

14 Networking with the ncat device

My Slackware distribution comes with ncat installed as a part of the nmap package, <http://nmap.org/ncat/>. I assume, therefore, that most other distributions will have ncat available. The network is set up so each trlog instance talks to the next, forming a loop just like the original serial port network. That way most of Tree's code can be used without change.

Security disclaimer: I implemented this without any encryption. You should therefore trust the other users on your local network, and the port(s) you choose for the communication should be behind a firewall that does not expose them to the rest of the internet. If you have a real need to use secure sockets, because of networking between distant stations, let me know and I can try to get that set up.

To use ncat networking, the ncat command should be in your PATH. Add a line like:

```
MULTI PORT = SERIAL ncat;12002;shack.myhouse.org:12001
```

or

```
MULTI PORT = SERIAL ncat;12001;192.168.14.27:12001
```

The form is

```
MULTI PORT = SERIAL ncat;localport;nextmachine_ip_address:nextmachine_port
```

If all the trlog instances are running on different computers, then all the ports can be the same number. This should be an unused port number greater than 1024 (since ports below 1024 are privileged). You can check the file /etc/services for the well known port numbers and make sure you are not using any of those.

Trlog will listen on the localport and will send to nextmachine_port on the machine with ip address nextmachine_ip_address. Be sure to notice the use of both semicolons (to separate the different parts of the command string) and a colon (to separate the machine address from the port.)

As an example, if you have 3 computers with IP addresses, shack1.myhouse.org, shack2.myhouse.org, shack3.myhouse.org. You could decide to run ncat on port 12001 on all of them.

The trlog version on shack1.myhouse.org would have the line

```
MULTI PORT = SERIAL ncat;12001;shack2.myhouse.org:12001
```

The trlog version on shack2.myhouse.org would have the line

```
MULTI PORT = SERIAL ncat;12001;shack3.myhouse.org:12001
```

The trlog version on shack3.myhouse.org would have the line

```
MULTI PORT = SERIAL ncat;12001;shack1.myhouse.org:12001
```

15 xterm setup

The xterm terminal emulator for X is distributed with every linux system, so I decided to run TRlog inside an xterm. TRlog expects VGA sized screens, so you need to set up the xterm with the correct options. These are most easily set in the .Xresources file in your home directory (notice the "." at the beginning of the file name!). Here is an example which is included in the distribution in trlinux-rN.NN/files/Xresources:

```
!Use:
!xterm -class DosTerm
DosTerm*faceName: Lucida
DosTerm*faceSize: 14
DosTerm*renderFont: true
DosTerm*VT100*background: black
DosTerm*VT100*foreground: white
DosTerm*VT100*geometry: 80x25
DosTerm*eightBitInput: false
DosTerm*VT100.translations: #override \
  Ctrl<Key>minus: string(0x1f)


!Use:
!xterm -class BigDosTerm
BigDosTerm*faceName: Liberation Mono
BigDosTerm*faceSize: 12
BigDosTerm*VT100*background: black
BigDosTerm*VT100*foreground: white
BigDosTerm*VT100*geometry: 80x50
BigDosTerm*eightBitInput: false
BigDosTerm*VT100.translations: #override \
  Ctrl<Key>minus: string(0x1f)
```

You can then start these with the commands

```
xterm -class DosTerm
```

or

```
xterm -class BigDosTerm
```

You will want to use the BigDosTerm class if you have a band map or visible dupe sheet. You can use the command

```
fc-list :scalable=true:spacing=mono: family
```

to list font names. A name before the first colon (:) can be used in place of my choices above for the faceName values. The faceSize values can be changed to change the size of the font and the xterm window.

It is absolutely necessary that the DosTerm be 80 columns and 25 rows, and the BigDosTerm 80 columns and 50 rows. You can check this with the commands

```
set |grep LINES
set |grep COLUMNS
```

the first should print LINES=80 and the second LINES=25 or LINES=50 depending on whether you are in a DosTerm or a BigDosTerm. If these do not match, you likely have a font that is too big to allow the xterm its full size. In that case change the faceSize value in the .Xresources file.

All of this can be a annoying to get set up the way you like, but at least it only needs to be done once.

16 File location

Since under DOS everyone who touched a computer had complete control, DOS TR would write various files back to the directory which held the TR and POST executables. All modern operating systems are multiuser. Therefore, you should be able to have one copy of trlog installed on your computer, and all users of the computer have access to the executables, and contest data, but should not be able to change each other's files, and only the root user should be able to change the installation.

The linux version starting with version 0.16, will create a .trlog subdirectory in your home directory if needed, i.e. it will create $HOME/.trlog if needed. This will be where changes to the files CABNAME.DAT, ADDRESS.DAT, TRMASTER.DTA will be stored for each user. This means that if you have a guest operator, you can make a new user account for that operator, and changes made by that user should not affect any of your configuration files. You should be able to install trlinux in any convenient directory e.g. /usr/local/trlinux which can be globally readable and executable, but not writeable.

17 YCCC SO2R box

Information about the YCCC SO2R box, its design, and how to obtain one, can be found at <http://k1xm.org/SO2R/index.html>.

After assembling your SO2R box, it is highly recommended that you first test that it functions correctly on a Windows computer, so that you know the SO2R box is working.

17.1 Setting permissions

On linux the first thing to do is to set up permissions for the YCCC SO2R box. Trlog must have have read/write permission to the SO2R box hardware device. The standard way to set this up is to add a udev rule, so when the operating system detects the YCCC SO2R box, it will set the desired owner, group, and permissions for the YCCC SO2R box device.

- As superuser, use a text editor to create a new rule file. I use

  `/etc/udev/rules.d/50-usb-so2r.rules`

- The contents of this file should tell the operating system what you want to do. Here are two examples:

  - With contents

    ```
    SUBSYSTEM=="usb", ATTR{idVendor}=="16c0", ATTR{idProduct}=="065e", MODE="0666"
    ```

    all users will have read/write permission. If your computer is secure, this should work fine.

  - A more secure solution uses instead a rules file with contents

    ```
    SUBSYSTEM=="usb", ATTR{idVendor}=="16c0", ATTR{idProduct}=="065e", MODE="0660"
    ,GROUP="so2rbox"
    ```

    all on one line. Only members of the so2rbox group will have read/write permission. You will have to create this group. You can do this using the command groupadd. That is on Slackware, with my user name w9cf I can add the group with

    `sudo groupadd so2rbox`

    and then add my account to that group using

    `sudo gpasswd -a w9cf so2rbox`

    You need to log out, and log back in, in order to have your login session reread the group file. Make sure that typing

    `groups`

    at the command line returns the name so2rbox (or whatever group name you used.)

In the udev rules above, the hex vendor and product IDs are those reported by the firmware of the YCCC SO2R box. All YCCC SO2R boxes use these same values.

17.2 Optional: Checking YCCC SO2R box permissions

You can double check that the udev rule is working and giving the proper permissions. If your linux works like mine:

- Type

  `lsusb -d 16c0:065e`

You can also type lsusb without options, but you will get a listing of all your usb devices not just the YCCC SO2R box.

- You should see something like

  ```
  Bus 002 Device 003: ID 16c0:065e VOTI
  ```

  where VOTI is the manufacturer ID of the YCCC box. Your bus and device numbers will likely be different.

- Check the permissions on this device by using a command like

  ```
  ls -l /dev/bus/usb/002/003
  ```

  where you should use the Bus and Device numbers from your lsusb command output instead of 002 and 003 above. The first example udev rule above will show something like

  ```
  crw-rw-rw- 1 root root 189, 1 Sep  4 21:44 /dev/bus/usb/002/003
  ```

  where the crw-rw-rw shows a character device owned by root with group root but all users have read/write permission. The second example udev rule above will show something like

  ```
  crw-rw---- 1 root so2rbox 189, 1 Sep  4 21:44 /dev/bus/usb/002/003
  ```

  which has owner and group read/write permission and the group is set to so2rbox. Users not in the so2rbox group do not have permission to communicate with the YCCC SO2R box.

17.3 Using the YCCC SO2R box

To use the box, include the configuration line

```
yccc so2r box enable = true
```

in LOGCFG.DAT

Other LOGCFG.DAT commands that can be used are

```
so2r microphone relay enable = true
```

with value true or false. This turns the microphone relays on or off in the YCCC SO2R box. Default is true.

```
so2r blend enable = true
```

with value true or false. Default is false. This turns on blending (i.e. some of the left radio in the right ear, and some of the right radio in the left ear).

```
so2r blend = 50
```

where the value should be between 0 (for minimal blending) to 255 (for maximal blending).

```
so2r headphone mode = normal
```

The other values are spatial and symmetric. See the YCCC SO2R box documentation for details. Default is normal.

In latch mode the transmitting rig is normally muted in the headphones. If the headphone mode is normal (stereo), the nontransmitting rig will be in both ears. On cw, if the paddle or keyboard cw is used, the transmitting rig is not muted. You can turn on latch mode with

```
so2r latch = true
```

```
so2r rig1 map = 1
so2r rig2 map = 2
```

These set the mapping from rig1 and rig2 to the output 1 through 4 on the back of the YCCC box. A value of 0 will use whatever the box is set up for currently. Default is 0. You can also use the values -1, -2, -3, -4. These will save the mapping in the YCCC box's EEPROM. The EEPROM values are loaded on power up.

You can change all of these value using the control-J menu.

The second way to send commands to the SO2R box is to include commands as messages analogously to the way that raw rig control commands can be used as messages, i.e. ⟨03⟩SO2R=Command⟨04⟩. The ⟨03⟩ and ⟨04⟩ are CTRL-C and CTRL-D. The current commands are:

```
LATCHON
LATCHOFF
LATCHTOGGLE
RX1
RX2
STEREO
RXA
RXI
```

LATCHON turns on latch mode, LATCHOFF turns off latch mode, while LATCHTOGGLE toggles the latch mode. RX1 puts receive focus on Rig 1, RX2 puts receive focus on Rig 2. RXA puts receive focus on the active radio. RXI puts receive focus on the inactive radio. STEREO puts the SO2R box in Stereo mode.

TR puts the band data on the auxiliary connector (the DB25 connector) of the SO2R box. Pins 1-4 are the band data for rig 1, Pins 5-8 are the band data for rig 2. Pin 9 of the axiliary port is the common ground. These set such that Aux pins 1, 2, 3, 4, are equivalent to the parallel port pins 2, 7, 8, 9, respectively, of DOS TR, and Aux pins 5, 6, 7, 8, are equivalent to pins 2, 7, 8, 9 of rig 2's band output parallel port.

You can plug a footswitch into the PTT connector of the YCCC SO2R box. It then becomes the footswitch with operation as explained in the DOS TR manual. Except,

- The footswitch will always toggle the radio PTT line on the rig connectors of the YCCC box, in all modes. This is the way the hardware is set up, and cannot be changed (except by modifying the YCCC SO2R box firmware.) This means that you can only use the SO2R box footswitch for other operations if you do not connect the connector PTT line to the rig.
- CW grant mode does not work. Footswitch mode Normal is untested.

17.4 Controlling a digital voice keyer

Pins 18-24 on the YCCC SO2R box DB-25 can be used in place of a parallel port to control a digital voice keyer. Pin 18 is DVK0 and is pulsed to abort the voice keyer. Pins 19-24 are connected to DVK1-DVK6. To turn this on use

```
DVK PORT = YCCC
```

The operation is the same as the W9XT DVK as explained in the DOS TR manual. Recording from TR commands is not supported.

Normally TR is set up to trigger recording on a W9XT DVK card when the corresponding control keys are pressed. You can disable this behavior, so that control-f1 through control-f10 can be used for other messages with the command

```
DVK CONTROL KEY RECORD = FALSE
```

This can also be changed in the control-j menu.

18 Using a rig's internal voice keyer

You can use a rig's internal voice as the DVK by first setting

```
dvk radio enable = true
```

You must also set up the strings to send as rig commands to trigger the DVK messages using

```
dvk radio1 stop cmd =
dvk radio1 play dvk1 cmd =
dvk radio1 play dvk2 cmd =
dvk radio1 play dvk3 cmd =
dvk radio1 play dvk4 cmd =
dvk radio1 play dvk5 cmd =
dvk radio1 play dvk6 cmd =

dvk radio2 stop cmd =
dvk radio2 play dvk1 cmd =
dvk radio2 play dvk2 cmd =
dvk radio2 play dvk3 cmd =
dvk radio2 play dvk4 cmd =
dvk radio2 play dvk5 cmd =
dvk radio2 play dvk6 cmd =
```

These are then accessed in the funciton key messages as for a external DVK with something like

```
cq ssb memory f1 = dvk1
```

19 Using rigctld

Elecraft/Kenwood and Icom rigs are supported natively. For other rigs, you need to install the hamlib library which comes with the rigctld daemon. You can then specify the rig with a configuration command like:

```
radio one type = rigctld;4532;/dev/ttyUSB0;1608;57600;38;
radio two type = rigctld;4534;/dev/ttyS0;221;4800;38;
```

Starting the radio type with rigctld tells TR to start the rigctld program for that radio. The semicolon separated values following are the port to use (these need to be different if more than one rigctld is started), the serial port that connects to the radio, the hamlib radio model number, the baud rate, and finally the CIV address. A list of radio model numbers supported by hamlib can be found by typing

```
rigctl -l
```

at the linux command-line prompt in a terminal window. You can usually find the model number quickly with something like the examples below

```
rigctl -l 2>/dev/null |grep -i k2
rigctl -l 2>/dev/null |grep -i orion
rigctl -l 2>/dev/null |grep -i kx3
rigctl -l 2>/dev/null |grep -i ic-7700
```

The CIV address only matters for Icom rigs, and should match the rig's CIV address. For other brands, any value will work.

The hamlib recommended rigctld port are the even numbered ports 4532, 4534, 4536, etc. You can, of course, use any unused nonprivileged port.

If you want TR to connect to a rigctld that is or will be started by another process, then leave out everything excecpt the port number, i.e.

```
radio one type = rigctld;4532;
```

will set up TR to communicate with rigctld commands on port 4532 but will not launch rigctld.

20 Real time contest score reporting

TR can report contest scores and band breakdowns to one of the contest scoreboards if desired. Currently, this is not user friendly at all. It must be set up in the LOGCFG.DAT or equivalent file at program start up. Equivalent control-J commands have not been implemented. *Currently no input checking is enforced, and correct input defaults are not set up for any contests. Everything must be set manually.*

The commands are described in the next sections:

20.1 Enabling and specifying the score reporting web site

These commands choose the score reporting web site, and sets your credentials, if any.

- `score report enable =`

  Setting this to true turns on score reporting. False is the default.

- `score report post url =`

  This is the url for the web site used for score reporting. This should normally be set to the score distributor `http://scoredistributor.net` which will then send the results to the other score boards. If you want to send to a specific site only, the current scoreboard sites are `https://contestonlinescore.com/post/` and `https://cqcontest.net/postscore.php`. There are also corresponding unencrypted http sites `http://post.contestonlinescore.com/` and `http://cqcontest.net/postscore.php`.

  The code also recognizes a udp: url so, for example, setting the url to `udp://192.168.73.255:12060`, will properly configure TR for WRTC 2018 score reporting. This will send a UDP packet with the score xml on port 12060 on the local network (in this case 192.168.73.xxx) broadcast address.

- `score report username =`

  The user name used for authentication at the score reporting web site. This is typically your call sign. This is only used if the url begins with "https:". The default is the null string.

- `score report password =`

  The password used for authentication at the score reporting web site. This is only used if the url begins with "https:". The default is the null string. Please note that since your username and password will be stored *in the clear* in the TR configuration file, that this is not secure.

  You should make the file containing your password readable only by you, i.e. with a command like

  ```
  chmod 600 LOGCFG.DAT
  ```

  However, any administrator on your computer, or any malware that gains root permission would be able to read this trivially. You should think of this username password as locking your front door and then leaving a key under the door mat or under a nearby flower pot. It tells people that you don't want them to barge in on you, but you know a burglar will be able to get in.

  Making this secure does not seem like a good use of developer time, since the only consequences of stolen credentials is that someone could post wrong scores from you until you notified the score reporting site to reset your account. If this bothers you, use the unsecured web sites that do not require a username and password.

  Be sure that the password you use here is not used for any important account.

20.2 Operator entry class

The following commands speficy your entry class. The default for all of these is a null string.

- `score report class ops =`

  The choices are SINGLE-OP, MULTI-OP, or CHECKLOG.

- `score report class bands =`

  The choices are ALL, 160M, 80M, 40M, 20M, 15M, 10M, 6M, 2M.

- `score report class power =`

  The choices are HIGH, LOW, QRP.

- `score report class transmitter =`

  The choices are ONE, TWO, LIMITED, UNLIMITED, SWL.

- `score report class mode =`

  The choices are MIXED, CW, SSB, RTTY

  `score report class assisted =`

  The choices are ASSISTED, and NON-ASSISTED.

20.3 Contest specifics

- `score report breakdown enable =`

  Setting this to true uploads band and mode breakdowns for QSOs and multipliers, otherwise just the total QSOs, multipliers, QSO points and score are reported. False is the default.

- `score report contest =`

  The is the contest name as specified by the score reporting web site. This does not appear to be standardized. It is usually the same string that is used by the Cabrillo file, but not always. One list can be found here `https://contestonlinescore.com/settings/`.

20.4 Multipliers

Assigning the 4 kinds of TR multipliers to the score reporting names must also be done manually. These are done using

```
score report dx multipliers =
score report zone multipliers =
score report prefix multipliers =
score report domestic multipliers =
```

Only the multipliers used for your particular contest must be set. If the same name is used for more than one type of multiplier (for example in NA Sprint, I think both **score report domestic multipliers** and **score report dx multipliers** should be set to **state**), the multipliers are summed in the upload report. Multipliers not used by TR for the contest are ignored. The typical values from `https://contestonlinescore.com/settings/`. are **country**, **state**, **zone**, **hq**, **gridsquare**.

So for CQ WW you would use

```
score report dx multipliers = country
score report zone multipliers = zone
```

while for IARU you would use

```
score report domestic multipliers = hq
score report zone multipliers = zone
```

20.5 Miscellaneous score reporting commands

- `score report club =`

  Set the name of your club in your score report. The default is null.

- `score report interval minutes =`

  Any positive integer. This gives the approximate time between score uploads in minutes. The default is 2. The first score report is sent at about 1/20 of this interval after the program starts. So after 6 seconds for the default.

- `score report debug enable =`

  Setting this to true turns on verbose debugging output which is sent to the file curl.dbg in the start up directory. This can be useful if something doesn't work. It shows the xml data that will be uploaded, followed by libcurl verbose responses. False is the default.

  If you want to make some quick tests without uploading data, simply redirect the uploading. For example, set

  ```
  score report post url = http://localhost:5555
  ```

  which sends the output to port 5555 on your computer. Then, in a terminal window start:

  ```
  ncat -l -p 5555
  ```

  and you will see the transmitted output of TR. Note in this case since ncat is not a real http server, it will not give a response so the transaction will not complete.

20.6 Examples

These were two contests that allowed uploading of data while the code was being developed. The Russian world-wide multimode contest:

```
score report enable = true
score report debug enable = false
score report breakdown enable = true
score report contest = RUS-WW-MM
score report class ops = SINGLE-OP
score report class bands = ALL
score report class power = LOW
score report class transmitter = ONE
score report class mode = CW
score report class assisted = NON-ASSISTED
score report post url = https://contestonlinescore.com/post/
score report username = w9cf
score report password = secretpassword
score report club = Arizona Outlaws Contest Club
score report interval minutes = 1
score report domestic multipliers = none
score report dx multipliers = country
score report zone multipliers = state
score report prefix multipliers = none
```

and the Scandinavian activity contest SSB

```
score report enable = true
score report debug enable = false
score report breakdown enable = true
score report contest = SAC-SSB
score report class ops = SINGLE-OP
score report class bands = ALL
score report class power = LOW
score report class transmitter = ONE
score report class mode = SSB
score report class assisted = NON-ASSISTED
score report post url = https://cqcontest.net/postscore.php
score report username = w9cf
score report password = secretpassword
score report club = Arizona Outlaws Contest Club
score report interval minutes = 1
score report domestic multipliers = none
score report dx multipliers = country
score report zone multipliers = none
score report prefix multipliers = country
```

21 Troubleshooting

21.1 Some keys or key combinations don't work

This is likely either some program like a window manager or helper is intercepting the key presses before reaching the xterm, or your xterm version is not sending the standard function key strings.

You can test these as follows:

- It will be convenient to copy the Xresources and xinitrc files from the files subdirectory of the trlog distribution to you home directory.
- Exit X. If you are running a display manager (i.e. you logon through a pretty X display rather than the console login prompt:

  ```
  login:
  ```

  you need to do something like:

  ```
  telinit 3
  ```

  as root to enter run level 3 which often is the console login. However, some distributions make run levels 3 through 5 identical – see your distributions documentation and/or check the comments in /etc/inittab if this doesn't work.
- Log in at the console prompt. Type

  ```
  ls -l Xresources
  ls -l xinitrc
  ```

  to make sure those files are available. Now run a minimal X.

  ```
  xinit xinitrc
  ```

  which is X with no window manager and only a single xterm. In this xterm you can try running trlog and see if the function keys work.
- You can also simply type

  ```
  cat
  ```

  in the xterm and press the key combinations that aren't working in trlog. The strings that the xterm sends will be displayed (escape is ^[ ). Compare these to the expected strings in xterm.txt in the files subdirectory.
- If trlog works in this minimal xterm, then some part of your X installation is capturing the keystrokes before handing them to the xterm. You will have to figure out what this is, and turn it off when running trlog. Usually it is the window manager, but other helper programs can capture keystrokes too.
- If your version of xterm is not sending the correct strings expected by trlog, you can use the xterm translations property to change them.

21.2 Shared libraries not found

Since this is Tree's source code, he makes the rules. I therefore am only distributing a binary version.

To legally distribute a binary only version of trlog, some libraries (e.g. libsndfile) can only be dynamically linked. Here is the output of ldd trlog on my system:

```
linux-gate.so.1 =>  (0xffffe000)
libX11.so.6 => /usr/lib/libX11.so.6 (0xb75e2000)
libasound.so.2 => /usr/lib/libasound.so.2 (0xb7515000)
libieee1284.so.3 => /usr/lib/libieee1284.so.3 (0xb750b000)
libpthread.so.0 => /lib/libpthread.so.0 (0xb74f1000)
libsndfile.so.1 => /usr/lib/libsndfile.so.1 (0xb748b000)
libdl.so.2 => /lib/libdl.so.2 (0xb7487000)
libusb-1.0.so.0 => /usr/local/lib/libusb-1.0.so.0 (0xb747b000)
libc.so.6 => /lib/libc.so.6 (0xb7318000)
librt.so.1 => /lib/librt.so.1 (0xb730f000)
libxcb.so.1 => /usr/lib/libxcb.so.1 (0xb72f6000)
libXau.so.6 => /usr/lib/libXau.so.6 (0xb72f3000)
libXdmcp.so.6 => /usr/lib/libXdmcp.so.6 (0xb72ee000)
libm.so.6 => /lib/libm.so.6 (0xb72c8000)
/lib/ld-linux.so.2 (0xb772c000)
libFLAC.so.8 => /usr/lib/libFLAC.so.8 (0xb7279000)
libvorbisenc.so.2 => /usr/lib/libvorbisenc.so.2 (0xb7102000)
libvorbis.so.0 => /usr/lib/libvorbis.so.0 (0xb70db000)
libogg.so.0 => /usr/lib/libogg.so.0 (0xb70d5000)
```

Check your versions of these libraries if there are problems on your system. These libraries are mostly used for parallel port, usb, and sound card access. The trlog executable is linked with an RPATH that points to the lib subdirectory in the same directory as the trlog executable. If your system uses default libraries that differ from the ones above and do not work, you can put alternative libraries in the lib subdirectory. You should then see them dynamically linked if you run ldd trlog. The Slackware 13.37.0 libraries work on my system.

22 Reporting a program crash

Nothing that you as a user do should cause the program to crash. The most likely cause of a program crash is a bug in the program.

Hardware problems (like computer memory that is failing) are a possible, but much less likely reason for a program crash. Usually, if you are having hardware problems, multiple programs will crash intermittently.

Starting with version 0.32, the linux version of trlog will try to produce a core dump file if it crashes with an illegal instruction, floating point exception, or segmentation fault. Most linux distributions are set to not allow core dumps by default. If you are experiencing program crashes, and are willing to help me find the problem, please do the following:

- Start the xterm you use for trlog.

- In this xterm, before starting trlog, type the command

  ```
  ulimit -c
  ```

  If this reports anything but "unlimited", type

  ```
  ulimit -c unlimited
  ```

- Optional: you can check that trlog can now produce a core file. Start trlog, and in another terminal type

  ```
  killall -s 11 trlog
  ```

  This should cause trlog to crash and create a core file (usually named "core") in the current directory. Now remove this file

  ```
  rm core
  ```

  If the core file is not produced, it is possible that you have a newer linux kernel (3.7 or later) that has been configured without the CONFIG_COREDUMP option. If this is the case, you will not be able to produce core dumps unless you recompile the kernel with this option turned on.

- When trlog crashes, send me an email with the following attachments
  - The core file.
  - Your configuration file(s) LOGCFG.DAT or equivalent.
  - Any information on what action you think might have led to the crash.

23 Using pseudoterminals – Deprecated

This section is now deprecated. You should use the shell device for packet and the ncat device for the multi-op network. The pseudoterminal support does still work.

You can set up packet, mulit-op networking etc. exactly as under DOS trlog using serial ports.

An alternative that allows you to connect using standard networking connections – so, for example, you can connect to a packet cluster over telnet, is to use pseudoterminals. You can use either the legacy berkeley pseudoterminals if you have support compiled in your kernel or as of version 0.04, standard pseudottys using /dev/ptmx etc. Note the legacy Berkeley psudottys have been deprecated, and may not be compiled into your kernal by default. Now that standard pseudottys are supported in trlog, you should probably use them.

23.1 Telnet packet with standard pseudottys

Here's a minimal LOGCFG.DAT to use a pseudotty for telnet packet:

```
MY CALL = W9CF
CONTEST = CQ WW
DISPLAY MODE = COLOR
PACKET PORT = SERIAL ptypacket
EIGHT BIT PACKET PORT = TRUE
BAND MAP ENABLE = TRUE
```

You can choose any unique name that starts with pty or PTY. I chose the name ptypacket above. Trlog will open the next available pseudo tty master and use it for the serial port. It then opens the file given by the name you used (here ptypacket) and write the slave device name to that file.

I use the following script to fire up trlog (trlog must be in your path)

```
#!/bin/bash
xterm -class BigDosTerm -e trlog &
sleep 2
ptyslave=‘cat ptypacket‘
stty icrnl erase ’^h’ < /dev/ttyqa
telnet n7az.net < $ptyslave > $ptyslave
```

The first line creates an xterminal of the BigDosTerm class as described above, and runs trlog putting the process in the background. It then sleeps 2 seconds to allow trlog to set up the master end of the pseudotty. The next line reads the slave device from the file ptypacket that trlog creates. Then it sets up the slave end translating carriage returns to line feeds and using ctrl-h as the erase character. The last line starts a telnet connection to the n7az.net telnet cluster.

When trlog opens, you type ctrl-b to get to the packet window, and login to n7az.net. You can type sh/dx to get some spots to start. Close the packet window with another ctrl-b and it should act exactly as a serial port connection to a packet radio cluster.

23.2 Telnet packet with legacy Berkeley pseudotty

Here's a minimal LOGCFG.DAT to use a legacy pseudotty for telnet packet:

```
MY CALL = W9CF
CONTEST = CQ WW
DISPLAY MODE = COLOR
PACKET PORT = SERIAL /dev/ptyqa
EIGHT BIT PACKET PORT = TRUE
BAND MAP ENABLE = TRUE
```

where I have assumed that no other process is using the master /dev/ptyqa. Normally this will be true since most linux codes will use the more modern /dev/ptmx method, but if not, look in /dev and choose a different pty device. Note, you need to have the required permissions to

open the device. On my systems, they are owned by root and in the tty group, so including my username in /etc/group for the tty group gives me permission. You can do this with the command

```
sudo gpasswd -a w9cf tty
```

You must log out and log back in for this to take effect. You can check the groups you are a member of with the command

```
groups
```

I use the following script to fire up trlog (trlog must be in your path)

```
#!/bin/bash
xterm -class BigDosTerm -e trlog &
sleep 2
stty icrnl erase '^h' < /dev/ttyqa
telnet n7az.net < /dev/ttyqa > /dev/ttyqa
```

The first line creates an xterminal of the BigDosTerm class as described above, and runs trlog putting the process in the background. It then sleeps 2 seconds to allow trlog to set up the master end of the pseudotty. The next line says to set up the slave end translating carriage returns to line feeds and using ctrl-h as the erase character. The last line starts a telnet connection to the n7az.net telnet cluster.

When trlog opens, you type ctrl-b to get to the packet window, and login to n7az.net. You can type sh/dx to get some spots to start. Close the packet window with another ctrl-b and it should act exactly as a serial port connection to a packet radio cluster.

23.3 Multi networking with standard pseudottys

The N6TR multi networking can be set up the same way on two machines with the configuration files below. Assume the first machine has ip address 192.168.0.1 with TRlog configuration file

```
MY CALL = W9CF
CONTEST = CQ WW
DISPLAY MODE = COLOR
MULTI PORT = SERIAL ptymulti
COMPUTER ID = X
```

and the second with ip address 192.168.0.2 and configuration file

```
MY CALL = W9CF
CONTEST = CQ WW
DISPLAY MODE = COLOR
MULTI PORT = SERIAL ptymulti
COMPUTER ID = Y
```

It is convenient (but not necessary) to use the same pty name (here I chose ptymulti) for all of the machines.

Start trlog on each machine. Note the current directory name where the LOGCFG.DAT file. Below I will assume this is $HOME/radio/multi. In this example, trlog will connect will write out the name of the slave device in the file $HOME/radio/multi/ptymulti on each machine. Once all the trlog instances are running, you can connect the slave devices together. On machine 192.168.0.1, execute the script in another terminal window

```
#!/bin/sh
nextcomputer=192.168.0.2
nextptyfile=radio/multi/ptypacket
mypty=‘cat ptypacket‘
ssh $nextcomputer "cat > \‘cat $nextptyfile\‘" < $mypty
```

and similarly on 192.168.0.2 execute

```
nextcomputer=192.168.0.1
nextptyfile=radio/multi/ptypacket
mypty=‘cat ptypacket‘
ssh $nextcomputer "cat > \‘cat $nextptyfile\‘" < $mypty
```

For more computers, each will need to execute such a command to connect to the next machine in the circular N6TR multi network simulating a serial port connections around the loop. That is for four computers, computer 1 connects to computer 2 which connects to computer 3, which connects to computer 4, which connects back to computer 1.

23.4 Multi networking with legacy pseudottys

The N6TR multi networking can be set up the same way on two machines with the configuration files below. Assume the first machine has ip address 192.168.0.1 with TRlog configuration file

```
MY CALL = W9CF
CONTEST = CQ WW
DISPLAY MODE = COLOR
MULTI PORT = SERIAL /dev/ptyqb
COMPUTER ID = X
```

and the second with ip address 192.168.0.2 and configuration file

```
MY CALL = W9CF
CONTEST = CQ WW
DISPLAY MODE = COLOR
MULTI PORT = SERIAL /dev/ptyqb
COMPUTER ID = Y
```

It is convenient (but not necessary) to use the same master pseudotty device name for all of the machines.

Start trlog on each machine. This will open all the master pseudottys. Now on machine 192.168.0.1, execute the command in another terminal window

```
ssh 192.168.0.2 "cat > /dev/ttyqb" < /dev/ttyqb
```

and similarly on 192.168.0.2 execute

```
ssh 192.168.0.1 "cat > /dev/ttyqb" < /dev/ttyqb
```

For more computers, each will need to execute such a command to connect to the next machine in the circular N6TR multi network simulating a serial port connections around the loop. That is for four computers, computer 1 connects to computer 2 which connects to computer 3, which connects to computer 4, which connects back to computer 1.

24 Additional parallel port information

Unless you want to control old hardware that requires a parallel port to function, you shouldn't need a parallel port on your shack computer. TR can use USB and/or serial ports instead. In order to test TR, I have installed parallel ports on my shack computer – I don't use them in my normal operating. If you still want to add a parallel port, read on.

First, as far as I am aware, none of the available USB to printer adapters will work as a parallel port. Don't waste your money. This pretty much eliminates adding a parallel port to a modern laptop.

My current shack computer is a refurbished HP8300 (8GB memory, quad core Intel I5 processor, one serial port as delivered). There seem to be a large number of these available currently (November 2017) for around $120 or so (without a monitor). I think they are off lease. Mine looks to have been built in 2013. While the BIOS did not report a parallel port, the motherboard has a parallel port header. For about $5.00 on Ebay I bought an HP 462537-001, 462537-002, low profile parallel port adapter. It is just a cable from the motherboard header to a DB-25 that mounts in a spare card slot spot. The motherboard connector has a jumper to tell the BIOS it is installed. With this installed the HP8300 parallel port comes up and is recognized. It works fine with TR. (There is also a second serial port connector on the motherboard, but I haven't tried to install the equivalent rear panel connector since USB serial ports work fine.)

Most computers won't have a parallel port hidden on the motherboard. Before realizing mine did, I bought a PCI express parallel port card. It also works fine. The one I bought is a SYBA SD-PEX10005 PCI-Express Card 1x Parallel Port, MCS9900 Chipset, and costs around $15.00. It also works just fine. I did have to tell the parport_pc kernel module its address manually. That is after installing the card and booting up, I typed as root

```
lspci -v
```

This showed the parallel port card as

```
02:00.0 Parallel controller: MosChip Semiconductor Technology Ltd. Device 9900 (prog-if 0
        Subsystem: Device a000:2000
        Flags: bus master, fast devsel, latency 0, IRQ 11
        I/O ports at e010 [size=8]
        I/O ports at e000 [size=8]
        Memory at f7c01000 (32-bit, non-prefetchable) [size=4K]
```

```
Memory at f7c00000 (32-bit, non-prefetchable) [size=4K]
Capabilities: [50] MSI: Enable- Count=1/1 Maskable- 64bit+
Capabilities: [78] Power Management version 3
Capabilities: [80] Express Legacy Endpoint, MSI 00
Capabilities: [100] Virtual Channel
Capabilities: [800] Advanced Error Reporting
```

From that, the possible I/O port addresses are 0xe010 and 0xe000. By trying both addresses in turn via commands like

```
modprobe -r parport_pc
modprobe parport_pc io=0xe010
```

and then trying TR, I found that the 0xe010 address was correct. I then created the file /etc/modprobe.d/parport_pc.conf with the contents

```
options parport_pc io=0x378,0xe010 irq=7,none
```

where my 0x378 entry is for the motherboard port. If you have just the PCI express card, this file would contain something like:

```
options parport_pc io=0xe010 irq=7
```

The irq can also be set to none without affecting anything.

25 Disclaimer

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Index