



# The Open Traceable Time Platform

## User Manual

Version 1.0

Copyright 2016 E. Louis Marais, Michael Wouters

Generated March 3, 2020

This work is licensed under a Creative  
Commons Attribution 4.0 International License.

# Contents

<b>1</b>	<b>Introduction</b>	<b>7</b>
1.1	What is OpenTTP? . . . . .	7
1.2	The OpenTTP software suite . . . . .	8
1.2.1	Supported GNSS receivers . . . . .	8
1.2.2	Supported counters . . . . .	8
1.3	The OpenTTP reference platform . . . . .	8
1.3.1	Licenses . . . . .	9
<b>2</b>	<b>Getting started with the Reference Platform</b>	<b>10</b>
2.1	The front panel . . . . .	10
2.1.1	Using the front panel keypad . . . . .	11
2.1.2	Menus . . . . .	11
2.2	The rear panel . . . . .	14
2.3	Installation . . . . .	14
2.3.1	Operating environment . . . . .	14
2.3.2	Install the GPS antenna and cable . . . . .	14
2.3.3	Make other system connections . . . . .	15
2.4	Logging in . . . . .	16
2.5	The cvgps user . . . . .	16
2.6	Checking operation . . . . .	17
2.7	Local configuration . . . . .	17
2.8	Securing the system . . . . .	17
2.9	Maintenance . . . . .	17
2.9.1	Updating the software . . . . .	17
2.9.2	Replacing the SD card . . . . .	18
<b>3</b>	<b>The reference platform hardware</b>	<b>19</b>
3.1	Antenna . . . . .	19
3.2	Multi-channel counter/timer . . . . .	19
3.2.1	Counter delays . . . . .	21
3.3	Electrical specifications . . . . .	21
<b>4</b>	<b>Installing the software</b>	<b>23</b>

## CONTENTS

---

4.1	Installation requirements . . . . .	23
4.2	Obtaining, building and installing the software . . . . .	23
4.2.1	Building the documentation . . . . .	25
4.2.2	Installing the software . . . . .	25
4.3	A minimal software setup . . . . .	25
4.4	Common configuration problems . . . . .	26
4.4.1	UUCP lock file creation . . . . .	26
<b>5</b>	<b>GPSCV software</b> . . . . .	<b>27</b>
5.1	Software overview . . . . .	27
5.2	crontab . . . . .	27
5.3	Configuration file format . . . . .	29
5.3.1	Paths . . . . .	30
5.4	Data file formats . . . . .	30
5.4.1	GPS receiver . . . . .	30
5.4.2	Time-interval counter . . . . .	31
5.5	gpscv.conf - the core configuration file . . . . .	31
5.5.1	[Antenna] section . . . . .	32
5.5.2	[CGGTTS] section . . . . .	33
5.5.3	[Counter] section . . . . .	38
5.5.4	[Misc] section . . . . .	40
5.5.5	[Delays] section . . . . .	40
5.5.6	[Paths] section . . . . .	41
5.5.7	[Receiver] section . . . . .	42
5.5.8	[Reference] section . . . . .	45
5.5.9	[RINEX] section . . . . .	46
5.6	mktimetx . . . . .	47
5.6.1	usage . . . . .	47
5.6.2	configuration file . . . . .	48
5.6.3	log file . . . . .	48
5.7	runmktimetx.pl . . . . .	49
5.7.1	usage . . . . .	49
5.8	runr2cggtsv8.py . . . . .	49
5.8.1	usage . . . . .	50
5.9	cnt9xlog.py . . . . .	50
5.9.1	usage . . . . .	51
5.9.2	configuration file . . . . .	51
5.10	hp5313xlog.pl . . . . .	51
5.10.1	usage . . . . .	51
5.10.2	configuration file . . . . .	52
5.11	okxemlog.pl . . . . .	52
5.11.1	usage . . . . .	52
5.12	prs10log.pl . . . . .	53
5.12.1	usage . . . . .	53

## CONTENTS

---

5.13	ticclog.py . . . . .	53
5.13.1	usage . . . . .	53
5.14	Javad/Topcon receivers . . . . .	54
5.14.1	jnslog.pl . . . . .	54
5.14.2	jsextract.pl . . . . .	56
5.14.3	runrinexobstc.pl . . . . .	56
5.15	NVS NV08C receivers . . . . .	58
5.15.1	nv08log.pl . . . . .	58
5.15.2	nv08extract.pl . . . . .	58
5.15.3	nv08info.pl . . . . .	59
5.16	Trimble Resolution T receivers . . . . .	60
5.16.1	restlog.pl . . . . .	60
5.16.2	restextract.pl . . . . .	60
5.16.3	restinfo.pl . . . . .	61
5.16.4	restconfig.pl . . . . .	61
5.16.5	restplayer.pl . . . . .	62
5.17	ublox receivers . . . . .	62
5.17.1	ublox9log.py . . . . .	62
5.17.2	ubloxlog.pl . . . . .	63
5.17.3	ubloxextract.py . . . . .	63
5.17.4	ubloxmkdev.py . . . . .	64
5.18	Miscellaneous tools . . . . .	65
5.18.1	cggtsqc.py . . . . .	65
5.18.2	cmpecgts.py . . . . .	66
5.18.3	editcggts.py . . . . .	68
5.18.4	editrxnav.py . . . . .	68
5.18.5	editrxobs.py . . . . .	69
5.18.6	fetchigs.py . . . . .	69
5.18.7	ticqc.py . . . . .	71
<b>6</b>	<b>System software</b> . . . . .	<b>73</b>
6.1	dioctrl . . . . .	73
6.2	kickstart.pl . . . . .	73
6.2.1	usage . . . . .	73
6.2.2	configuration file . . . . .	74
6.3	mjd . . . . .	74
6.3.1	usage . . . . .	75
6.4	okcounterd . . . . .	75
6.4.1	usage . . . . .	75
6.5	okcounterdctl.pl . . . . .	76
6.5.1	usage . . . . .	76
6.6	okbloader . . . . .	77
6.6.1	usage . . . . .	77
6.7	lcdmonitor . . . . .	77

## CONTENTS

---

6.7.1	usage	77
6.7.2	configuration file	77
6.8	libraries	80
6.8.1	libconfigurator	80
6.8.2	TFLibrary.pm	81
6.8.3	OpenOK2	81
6.8.4	otplib.py	81
6.8.5	cggtslib.py	82
6.9	ppsd	83
6.9.1	usage	83
6.9.2	configuration file	84
6.10	sysmonitor.pl	84
6.10.1	usage	84
6.10.2	configuration file	85
6.10.3	log file	86
6.11	gziplogs.pl	86
6.11.1	usage	86
6.11.2	configuration file	87
<b>A</b>	<b>Software license</b>	<b>88</b>

# List of Figures

2.1	Front panel of the unit . . . . .	10
2.2	Menu structure . . . . .	12
2.3	Rear panel of the unit . . . . .	14
2.4	Directories in the <code>cvgps</code> account . . . . .	16
3.1	Selection of antenna voltage using JP1 . . . . .	20
3.2	The OpenTTP multi-channel counter. . . . .	21
4.1	Overview of the OpenTTP software source . . . . .	24

# List of Tables

1.1	Supported GPS/GNSS receivers. . . . .	8
1.2	Supported counters. . . . .	9
2.1	Rear panel electrical connections . . . . .	15
2.2	Checking NTP operation . . . . .	17
3.1	Status LEDs on Opal Kelly XEM6001 board . . . . .	22
3.2	XEM6001 counter delays . . . . .	22
3.3	Electrical signals and their characteristics . . . . .	22
5.1	GPSCV software overview . . . . .	28
5.2	Summary of <code>gpscv.conf</code> entries . . . . .	31
5.3	Correspondence of CGGTTS and RINEX signal names. . . . .	37
5.4	Summary of <code>gpscv.conf</code> entries used by <code>mktime tx</code> . Optional entries are italicised. . . . .	48
6.1	PPS OUT channels for the <code>-p</code> option . . . . .	76



# 1. Introduction

This chapter gives a short introduction to the Open Traceable Time Platform (OpenTTP), describing the Reference Platform and other supported hardware.

## 1.1 What is OpenTTP?

The Open Traceable Time Platform is an open-source solution for a timing system that can be made fully traceable to national standards. It achieves traceability using the GPS common-view technique, which allows distant clocks to be compared with an accuracy of a few ns. The reference platform is based on readily available, low-cost OEM modules and provides a full software and hardware solution.

The goals of the OpenTTP project are:

1. Fully open source hardware and software <sup>1</sup>
2. Easy customisation for specialised applications
3. Production of time-transfer files in the standard CGGTTS data format (currently for GPS only)
4. Easy extension to new receivers
5. Low cost
6. Provision of a convenient framework for research and development.

Applications currently include provision of traceable time of day and auditing of NTP-synchronized systems.

---

<sup>1</sup>The Reference Platform uses some black-box software modules supplied by the FPGA vendor which cannot be freely distributed.

## 1.2 The OpenTTP software suite

The OpenTTP software suite provides a full solution for automated logging and processing of time-transfer data. It is available via GitHub:

<https://github.com/openttp>

Users are invited to contribute to its development.

The software supports a number of GNSS receivers and counters. The supported GNSS receivers are mostly low-cost, single-frequency receivers since low cost is a key objective of the OpenTTP project. Low-cost dual frequency receivers have recently become available and one of these, the ublox ZED-F9, is now supported by OpenTTP.

### 1.2.1 Supported GNSS receivers

OpenTTP currently supports the receivers listed in table 1.1.

Manufacturer	models	notes
Javad	GRIL receivers	obsolete
NVS	NV08	
Trimble	Resolution T	obsolete
ublox	NEO-M8T,ZED-F9P,ZED-F9T	may work with earlier receivers

Table 1.1: Supported GPS/GNSS receivers.

OpenTTP uses a custom file format for logging GPS receiver data. It does not read native receiver binary-format files.

Guidance on testing a receiver for suitability for time-transfer, and writing software to process the receiver's data, is given in the OpenTTP Developer's Guide.

### 1.2.2 Supported counters

Counters supported by OpenTTP are listed in 1.2.

The file format used 5.4.2 is very simple and it should be easy to convert from another format, if needed.

## 1.3 The OpenTTP reference platform

The OpenTTP reference platform consists of:

## 1. Introduction

---

Manufacturer	models	notes
Agilent	5313x	needs IOtech GPIB to RS232 converter
OpenTTP	XEM6001	simple FPGA-based counter/timer
Pendulum	CNT-9x	
SRS	PRS10	uses the input 1 pps time-tagging function
TAPR	TICC	

Table 1.2: Supported counters.

- BeagleBone Black, an ARM-based single board computer
- NVS Technologies NV08-CSM GNSS receiver
- Opal Kelly XEM6001 FPGA development board
- Jackson Labs LTE Lite GPS-disciplined oscillator
- Solid-state disk for mass storage
- CrystalFontz LCD module

Custom circuits, printed circuit board designs and other hardware resources are all available via the GitHub repository.

### 1.3.1 Licenses

The software is available under the MIT license.

## 2. Getting started with the Reference Platform

This chapter describes basic setup of the OpenTTP reference platform, verification of its operation and logging in to the unit.

Warning ! The unit contains a computer. This must be shut down properly before power is removed from the unit. Failure to do this can result in inoperability of the system. The system can be shut down from the front panel (2.1.1) or by logging in (2.4).



### 2.1 The front panel

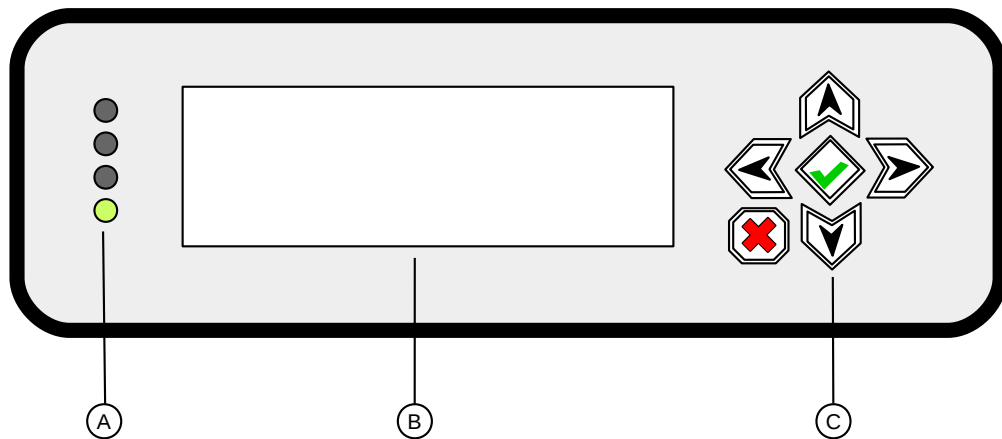


Figure 2.1: Front panel of the unit

- Ⓐ Status LEDs
- Ⓑ LCD display
- Ⓒ Keypad

## 2. Getting started with the Reference Platform

---

The top line of the LCD display shows UTC date and time. The date and time displayed will typically only be accurate to 1 s. The contents of the second and third lines of the display depend on the display mode [2.1.2](#).

The bottom line is reserved for notification of system alarms and either shows ‘System OK’ or ‘System Alarm’. The adjacent LED will be red if there is a running alarm. The other LEDs are currently unused.

### 2.1.1 Using the front panel keypad

The keypad provides access to status information and limited control and configuration of the unit. It can be used to cleanly shut down or reboot the computer without logging in.

System status is normally displayed on the screen. The menus are accessed by pressing any key. Menus are navigated using the keypad:

- ⏏ Move to next menu item
- ⏪ Move to previous menu item
- ⏩, ⏹ Select menu item
- ⏪ Back to previous menu
- ⏏ Back to the status display

Escaping back to the status page after making a change will not undo the change. Where a sub-menu lists a number of options, the currently selected option is flagged with an asterisk.

Dialogs are navigated using the cursor keys. A dialog will typically consist of a number of input fields. Some of these work like buttons and are selected using the ⏹ key; others may require inputting a value and this is done by cycling through the possible values with the cursor keys.

If you move out of an input field, focus will pass to the next valid input field.

You can quit a dialog using the ⏏ key. Any changes made in a dialog will not be applied if you quit it.

If a menu or dialog has been inactive for more than 5 minutes, the display returns to showing system status.

### 2.1.2 Menus

The menu structure is shown below:

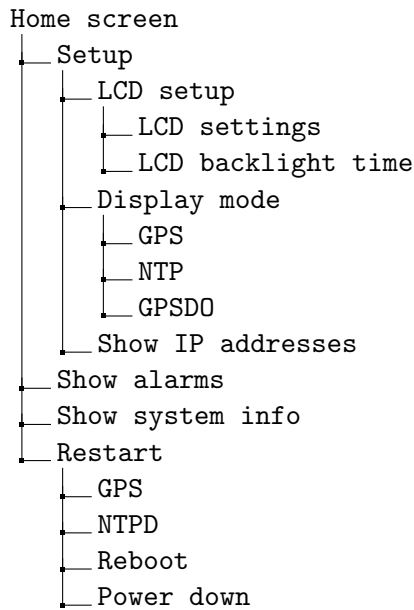


Figure 2.2: Menu structure

### LCD setup

The intensity and contrast of the LCD display can be set using this menu. A timeout can also be set on the backlight.

### Display mode

The status information displayed by the unit in the second and third lines of the LCD display can be selected as relating to either GPS, NTP or the GPSDO.

When in GPS mode, the identifiers of up to 10 currently visible GPS space vehicles are displayed.

When in NTP mode, the second line shows the number of NTP packets received per minute. The third line shows information about synchronization status, including leap second announcements.

In GPSDO mode, the second line shows whether the GPSDO is locked or not. The third line alternates between the ‘health’ byte (??), and the estimated fractional frequency error (reported by the GPSDO) and the electronic frequency control (EFC) voltage, normalized to a maximum of 100.

## *2. Getting started with the Reference Platform*

---

### **Show alarms**

This shows any currently running alarms as listed in `/home/cvgps/logs/alarms`.

### **Show system info**

This displays version and serial number information and the make of the installed oscillator. This information is read from the file `/usr/local/etc/sysinfo.conf` which has to be manually updated if the installed oscillator is changed.

### **Restart**

This allows the user to restart several processes (GPS common view logging and `ntpd`) as well as reboot or power down the unit. You will be asked to confirm power down or reboot.

## 2.2 The rear panel

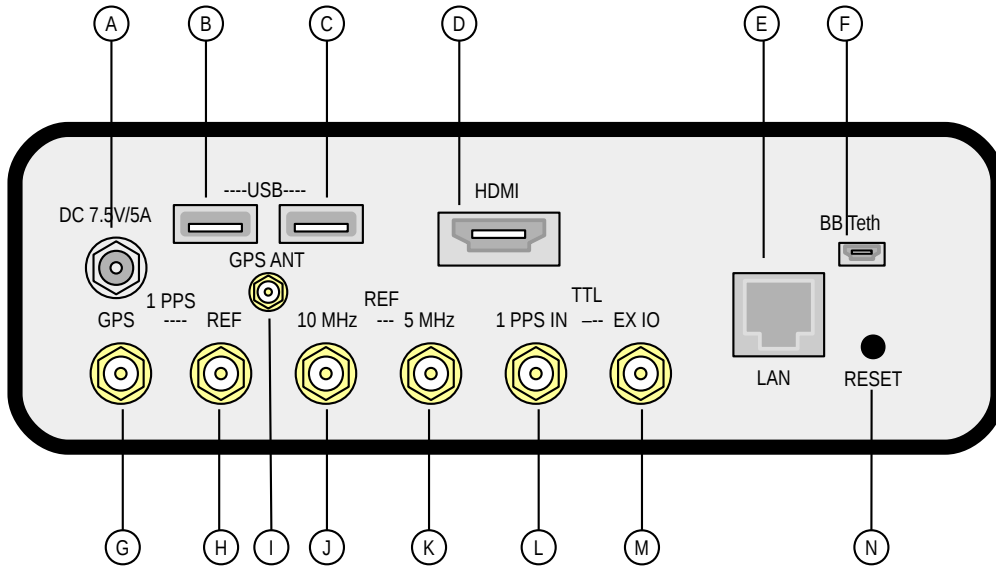


Figure 2.3: Rear panel of the unit

If connected to a computer via the USB connector (F), a network adapter should show up on the computer. The TTP will provide your computer with an IP address of either 192.168.7.1 or 192.168.6.1, depending on the type of USB network adapter supported by your computer's operating system. The TTP will reserve 192.168.7.2 or 192.168.6.2 for itself. Further information can be found in the BeagleBone Black documentation.

## 2.3 Installation

### 2.3.1 Operating environment

The TTP is designed for indoor use only and is neither water nor moisture-proof. It should not be subjected to large mechanical shocks or excessive heat and dust.

### 2.3.2 Install the GPS antenna and cable

The unit supplies 5 V DC to the antenna. This can be changed to 3.3 V via a jumper J1. More details are given in 3.1.



## 2. Getting started with the Reference Platform

---

	Function	Connector	Signal characteristics
Ⓐ	DC power input		7.5 V, 5A
Ⓑ	USB		
Ⓒ	USB		
Ⓓ	Video	HDMI	
Ⓔ	Network	RJ45	
Ⓕ	BB USB network		
Ⓖ	GPS 1 pps OUT	SMA	5V DC supplied
Ⓕ	Reference 1 pps OUT	SMA	
Ⓖ	GPS antenna	SMB	
Ⓖ	Reference 10 MHz OUT	SMA	
Ⓖ	Reference 5 MHz OUT	SMA	
Ⓖ	1 PPS IN	SMA	TTL
Ⓖ	General purpose I/O	SMA	TTL
Ⓖ	Beaglebone Black reset		

Table 2.1: Rear panel electrical connections

The GNSS antenna should be installed in a location with a clear view of the sky above 10°. All exposed connections should be weatherproofed. Make sure that all connections are tight but do not apply excessive torque as this can result in connectors detaching from the cable.

A strain relief bulkhead is used for making the connection to the short ‘N’-terminated cable.

### 2.3.3 Make other system connections

#### Network connection

A network connection is not required for operation as a time-transfer system but may be useful to the user for maintenance and downloading data files. Otherwise, a keyboard and monitor can be plugged in.

For operation as an NTP server, the unit will require a network connection. The unit can operate using DHCP or with a static IP address. The default is to use DHCP. The configured address can be conveniently found using the front panel menu.

A static IP can be set by using the utility `connmanctl` or by editing the file `/etc/network/interfaces`.

### Keyboard and monitor

A HDMI monitor and USB keyboard and mouse can be used with the TTP. However, this only allows a console login 2.4. The graphical login and desktop environment normally available have been disabled to reduce memory usage.

### Time and frequency signals

Specifications of the various output signals are given in 3.3.

## 2.4 Logging in

The TTP runs under Debian Linux. The desktop environment normally available has been disabled to reduce memory usage. Familiarity with a command-line Linux environment is therefore essential for operating and maintaining the TTP. It is beyond the scope of this manual to provide a Linux tutorial. The reader should consult one of the many online resources that exist.

Login is via the user `cvgps` (this name is used for historical reasons). The default password is supplied separately and you should change this after logging in.

## 2.5 The `cvgps` user

The `cvgps` user is the account used to log and process data.

It contains the following standard directories, some of which are mounted on the SSD.

```
/home/cvgps
├── bin ..... executables
├── etc ..... configuration files
├── cggts
├── logs
│   └── alarms
├── raw ..... data files from the receiver, counter, and GPSDO
├── rinex
└── tmp
```

Figure 2.4: Directories in the `cvgps` account

## 2. Getting started with the Reference Platform

---

remote refid st t when poll reach delay offset jitter

Table 2.2: Checking NTP operation

Automatic operation is managed through the `cvgps crontab` file [5.2](#).

### 2.6 Checking operation

Upon startup, a number of alarms may be produced. All alarms should clear within five minutes of startup. Information on alarms and troubleshooting hints are given in [??](#). Troubleshooting will require logging into the system ([2.4](#)).

You can check that logging of GNSS and counter data is occurring by looking in the directory `/home/cvgps/raw`.

Processing of data takes place at UTC0015 daily. CGGTTS files will be placed in the `/home/cvgps/cggtts` directory.

The TTP is NTP-synchronized using the GPS receiver's time of day messages and its 1 pps. The TTP operates on UTC. NTP operation can be checked using the command `ntpq -p`. This should display

### 2.7 Local configuration

To produce CGGTTS files, the processing software needs accurate antenna coordinates. These can be obtained from the TTP receiver, using `nv08extract`. More accurate coordinates can be obtained by submitting RINEX observation files to an online processing service. In the case of single frequency observations, the NRCAN PPP service can be used.

### 2.8 Securing the system

### 2.9 Maintenance

#### 2.9.1 Updating the software

Debian Linux is updated separately from the OpenTTP software.

Updating the kernel is a separate procedure.

## *2. Getting started with the Reference Platform*

---

Updating the OpenTTP software is most conveniently done via the git repository.

### **2.9.2 Replacing the SD card**

To replace the SD card, remove the screws from the front, remove the front panel and disconnect the cable connected to the LCD unit. Remove the screws at the rear and slide the electronics assembly out. You can remove the SD card without fully removing the electronics. Insert the new card and reassemble taking care to align the printed circuit boards with the slots in the enclosure.

## 3. The reference platform hardware

This chapter provides a more detailed description of the reference platform hardware.

### 3.1 Antenna

The DC voltage supplied to the antenna is selectable as either 3.3 or 5 V with the jumper J1. The default is 5 V. Short circuit protection is provided via the  $1\Omega$  fusible resistor R25.

The NV08C-CSM has two antenna inputs ANT-A (active) and ANT-P (passive). When the antenna is connected to ANT-P (default setting) the external DC power supply is used. If the antenna is connected to ANT-A, the NV08-CSM board supplies 3.3 V DC and short circuit protection is provided by the board.

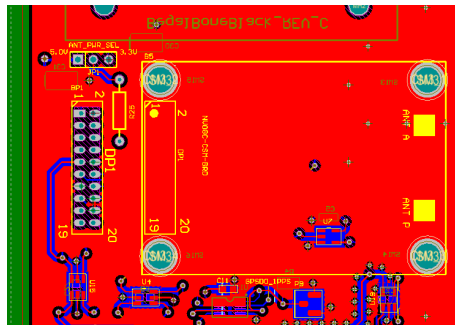
### 3.2 Multi-channel counter/timer

The OpenTTP reference platform uses an Opal Kelly XEM6001 FPGA (Xilinx Spartan 6 LX16) development board to provide a multi-channel counter. The counter is clocked at 200 MHz, so the full-range resolution is 5 ns. This is adequate for the applications envisaged for the OpenTTP system.

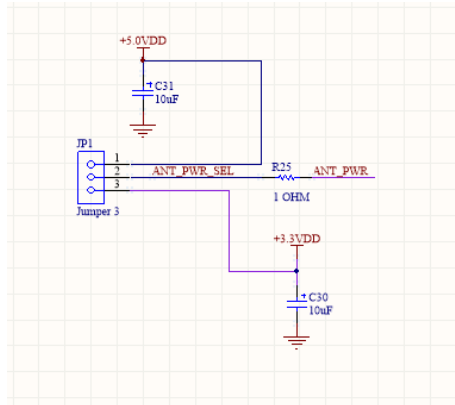
Although the counter has six channels, only three (FIXME four?) are connected in the current design 3.2. They are all configured with the system reference as START, with the exception of Channel 5. This is configured with the externally input 1 pps as START, and the time-transfer receiver's 1 pps as STOP. This is to allow time-transfer for an external reference.

### 3. The reference platform hardware

---



(a) Location of JP1



(b) JP1 voltages

Figure 3.1: Selection of antenna voltage using JP1

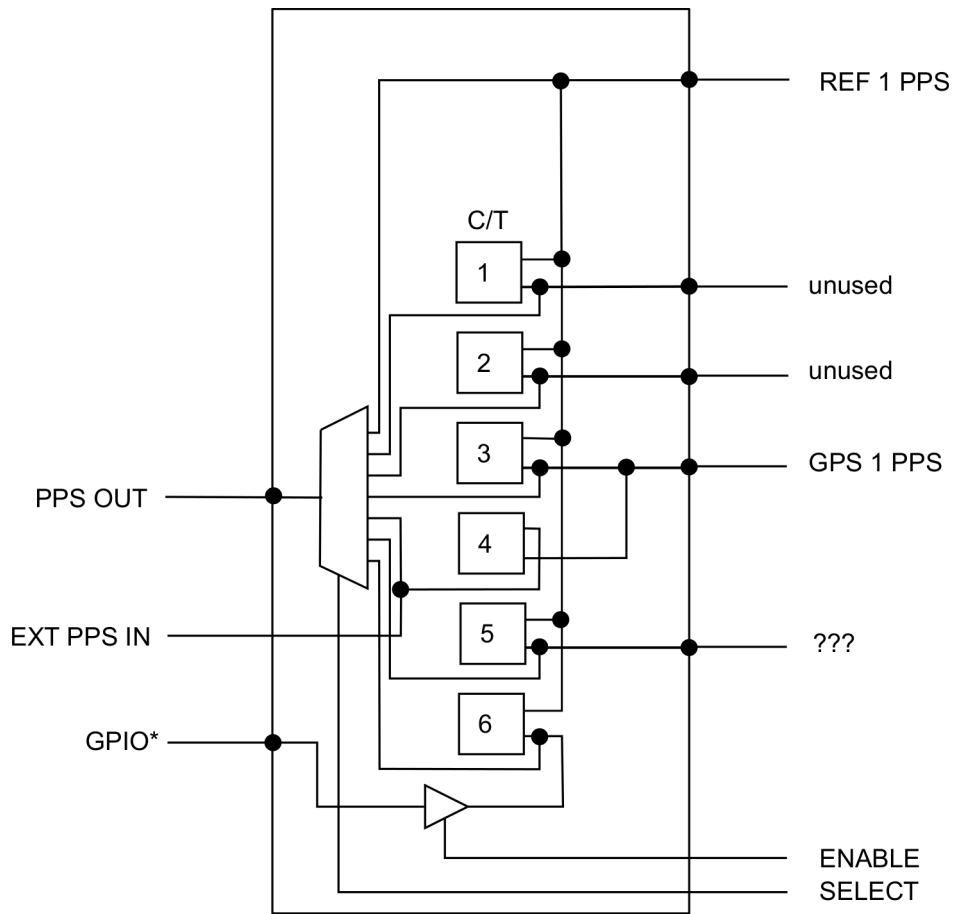


Figure 3.2: The OpenTTP multi-channel counter.

The XEM6001 has eight LEDs which may aid in fault-finding [3.1](#). In particular, the presence of 1 pps signals can be readily seen (the pulses are extended so that they are visible) and the lock state of the PLL producing the 200 MHz clock.

### 3.2.1 Counter delays

Differential delays, measured with respect to the system 1 pps, were measured by applying signals at the input of the FPGA.

## 3.3 Electrical specifications

### 3. The reference platform hardware

---

LED	function
1	channel 1 pps (unused)
2	channel 2 pps (unused)
3	channel 3 pps (GPS receiver)
4	channel 4 pps (external pps)
5	channel 5 pps (unused)
6	channel 6 pps (GPIO)
7	GPIO enabled
8	Digital Clock Manager PLL is locked

Table 3.1: Status LEDs on Opal Kelly XEM6001 board

channel	delay (ns)
1	TBA
2	TBA
3	TBA
4	TBA
5	TBA
6	TBA

Table 3.2: XEM6001 counter delays

GPS 1 pps out  
Reference 1 pps out  
Reference 10 MHz out  
Reference 5 MHz out  
1 pps in  
General purpose I/O

Table 3.3: Electrical signals and their characteristics



## 4. Installing the software

### 4.1 Installation requirements

You will need a basic Linux development environment, including C/C++ compilers, Perl and python2 and python3. You will likely need the development packages for:

- `boost` C++ libraries
- `libgs1` GNU scientific library

You may also need:

- `Time::HiRes` Perl library

The above requirements are by no means comprehensive. What you will need to install will depend very much on the Linux distribution you are using.

### 4.2 Obtaining, building and installing the software

The OpenTTP software is obtained from the git repository hosted by GitHub. There are two main branches, the ‘master’ and ‘develop’ branches. The ‘develop’ branch is generally stable, but may occasionally be broken.

Clone the repository

```
git clone https://github.com/openttp/openttp.git
```

change directory to the repository and then check out the branch you want to use

```
git checkout develop
```

#### 4. Installing the software

---

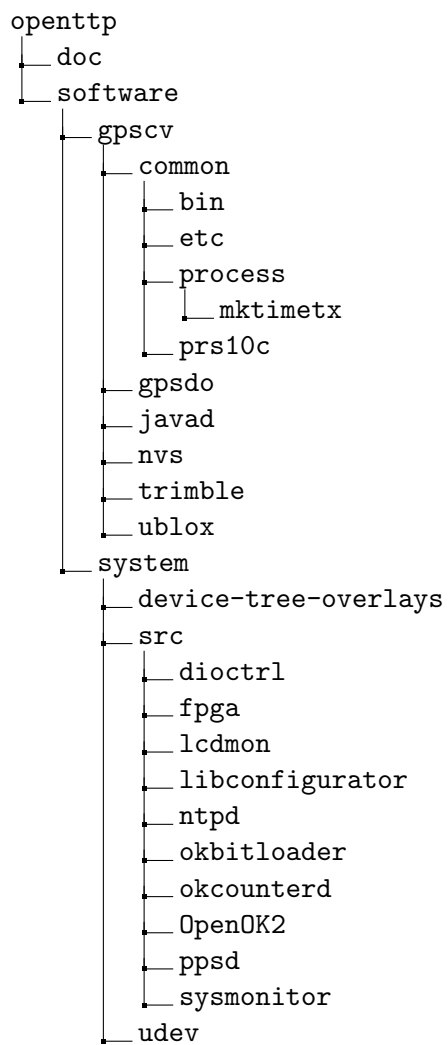


Figure 4.1: Overview of the OpenTTP software source

### 4.2.1 Building the documentation

The documentation is written in LaTeX and uses some packages and fonts that are generally packaged as ‘extras’ and will probably need to be installed. The various packages used can be inspected in the file `doc/manual/OpenTTPManual.tex`. To build the documentation as a PDF, change directory to `doc/manual` and run `make`.

### 4.2.2 Installing the software

Two scripts are provided for building and installing the software.

```
software/system/installsys.pl
software/gpscvcv/install.pl
```

The installer has been used with current versions of RedHat Enterprise Linux, CentOS, Ubuntu and Debian (on the BeagleBone Black).

`installsys.pl` must be run first (with superuser privileges), because it installs libraries which are needed by the GPSCV software.

`installsys.pl` can also be used to install various targets individually. Run `installsys.pl -h` to see the options available. This may be helpful when trying to resolve problems with building the software.

`install.pl` will archive any existing scripts and binaries, and create any directories it needs in the current user’s home directory. Run `install.pl -h` to see the options available.

## 4.3 A minimal software setup

For users who wish to use their own hardware, this section describes the minimum setup required for operation.

The OpenTTP software distribution is comprised of various C/C++ applications and Perl and python scripts. As a minimum, you will need:

- `TFLibrary.pm`, a Perl module for commonly used tasks, such as reading configuration files
- `otplib.py`, the python version of `TFLibrary.pm`
- `libconfigurator`, a C library for parsing configuration files
- `lockport`, a utility to create a UUCP lock file
- `mktimetx`, to create time-transfer files
- one of the OpenTTP-provided scripts to log your receiver
- one of the OpenTTP-provided scripts to log your counter/timer

You must use one of the OpenTTP receiver logging scripts with your GNSS receiver because `mktimetx` expects a custom file-format 5.4. In particular, the receiver's native binary formats are not readable by `mktimetx`. Similarly, OpenTTP uses a custom file format for the counter-timer measurement files, although in this case, conversion from another format will probably be straight-forward. Most likely, users will have to provide their own software for logging their counter/timer, given the large number of possible devices here, and the limited support within OpenTTP.

You may also find the following useful:

- `kickstart.pl` for automatic start of logging processes
- `runmktimetx.pl` for automated processing, and reprocessing
- `gziplogs.pl` for managing log file compression

## 4.4 Common configuration problems

### 4.4.1 UUCP lock file creation

UUCP lock files are used to prevent non-OTTP processes from opening serial ports while they are in use. You need to set the location for lock file creation specific to the operating system, and ensure that the OTTP user (`cvgps` typically) has the right permissions to write to this location. This is typically achieved by adding the user to the appropriate group. For example, on Ubuntu 18 the lock directory is `/var/lock`. In this case, the directory is world writeable, so the OTTP user does not need to belong to any supplementary groups.

## 5. GPSCV software

This chapter describes software related to GPSCV time-transfer.

### 5.1 Software overview

Time-transfer files are produced by `mktimetx` from the GNSS receiver logs and counter/timer measurements. The time-transfer files are in RINEX observation and CGGTTS format. Currently, CGGTTS-format files are only produced for GPS. Figure ?? illustrates the processing chain TODO.

The OpenTTP software suite is catalogued in Table 5.1

### 5.2 crontab

Automatic logging, processing and archival of data is co-ordinated via the user `cvgps`' crontab.

A minimal crontab for the user `cvgps` looks like this:

```
# Check that all logging is running every 5 minutes
*/5 * * * * /usr/local/bin/kickstart.pl # See ~/etc/
    ↪ kickstart.conf

# Run the processing of the data at 00:15
15 0 * * * nice $HOME/bin/runmktimetx.pl >/dev/null 2>&1

# Give the processing some time to complete, then zip the
    ↪ files at 00:45
45 0 * * * nice $HOME/bin/gziplogs.pl >/dev/null 2>&1

# Check the status of the system once a day, just before
    ↪ the day rollover
```

## 5. GPSCV software

---

	program	
GPSCV processing	<a href="#">mktimetx</a> <a href="#">runmktimetx.pl</a> <a href="#">runr2cggtsv8.py</a>	core program
TIC logging	<a href="#">hp5313xlog.pl</a> <a href="#">cnt9xlog.py</a> <a href="#">okxemlog.pl</a> <a href="#">prs10log.pl</a> <a href="#">ticclog.py</a>	
GNSS receiver logging	<a href="#">jnslog.pl</a> <a href="#">nv08log.pl</a> <a href="#">restlog.pl</a> <a href="#">ublox9log.py</a> <a href="#">ubloxlog.pl</a>	Javad NVS Trimble Resolution T ublox ublox
GNSS receiver utilities	<a href="#">jnsextract.pl</a> <a href="#">nv08extract.pl</a> <a href="#">nv08info.pl</a> <a href="#">restextract.pl</a> <a href="#">restinfo.pl</a> <a href="#">ubloxextract.py</a> <a href="#">ubloxmkdev.py</a>	
Analysis tools	<a href="#">cggtsqc.py</a> <a href="#">cmpcggts.py</a> <a href="#">editcggts.py</a> <a href="#">editrxnav.py</a> <a href="#">editrxobs.py</a> <a href="#">ticqc.py</a>	
Miscellaneous	<a href="#">log1Wtemp.pl</a>	

Table 5.1: GPSCV software overview

```
56 23 * * * $HOME/bin/checkstatus.pl >$HOME/  
    ↪ lockStatusCheck/status.dat
```

showing the three essential processes of logging, processing and archival of data.

The active crontab can be examined with the command `crontab -l`. A default crontab is saved in `/home/cvgps/etc/crontab`.

### 5.3 Configuration file format

Configuration files use a common format and are plain text files, designed to be easily edited via a command-line editor because in many applications, only shell access to the system will be available.

The file is usually divided into sections, with section names delimited by square brackets `[ ]`. Entries in each section are of the form:

```
key = value
```

For example,

```
[Receiver]  
manufacturer = Trimble  
model = Resolution T
```

defines a section `Receiver` and the two keys: `manufacturer` and `model`.

The notation `Section::Key` is used to fully specify keys. For example, `Receiver::model` and `Receiver::manufacturer` specify the two keys above.

Keys and section names are not case-sensitive. In particular, the python and Perl libraries which provide functions for reading the configuration files convert keys and section names to lower case. The case of key values is preserved, since this may be significant eg path names.

Leading and trailing whitespace is removed from keys, key values and section names.

Comments begin with a `#`.

Some keys define a list of sections. For example, the comma-separated list of values for `CGGTTS::outputs`

```
[CGGTTS]  
outputs = c1-code,p1-code,p2-code
```

defines three sections: `c1-code`, `p1-code`, and `p2-code`. This is a bodge which would be more elegantly addressed using an extensible format like XML, but it has proven to be sufficient for our needs.

### 5.3.1 Paths

Paths to files specified in a configuration file are constructed with the following precedence:

1. If the path begins with a leading slash, then it is interpreted as an absolute path
2. If a non-absolute path is specified, it is interpreted as being relative to the users's home directory (or where the configuration file allows specification of a different root path eg in [gpscv.conf](#), relative to that root path)
3. Otherwise, the default path is used.

Most software in OpenTTP follows these conventions.

## 5.4 Data file formats

### 5.4.1 GPS receiver

This text file records messages from the GPS receiver. The native format of the messages can be a mix of ASCII and binary messages. Binary messages are hexadecimal-encoded for saving in the log file. Some logging scripts record ancillary information, such as commands sent to the receiver. Comments in the log file are allowed, prefaced by a '#' character. The '@' character is used to tag lines containing special information that needs to be parsed by the processing software.

Messages are successive lines of the form:

```
<message_id> <time_stamp> <message>
```

*Example:*

```
TD 00:00:02 cdfbc75a9a8c353fc5
```

Hex encoding of binary messages results in much larger files but these compress to a size not much larger than the original binary data.



## 5. GPSCV software

---

Section	Key
<a href="#">Antenna</a>	antenna number, antenna type, delta H, delta N, delta E, frame, marker name, marker number, X, Y, Z
<a href="#">CGGTTS</a>	BIPM cal id, comments, create, ephemeris, ephemeris file, ephemeris path, internal delay, lab id, maximum DSG, minimum elevation, minimum track length, naming convention, outputs, reference, receiver id, revision date, version, code, constellation, path
<a href="#">Delays</a>	antenna cable, reference cable
<a href="#">Counter</a>	file extension, GPIB address, header generator, lock file, logger, logger options, okxem channel, port
<a href="#">Misc</a>	gzip
<a href="#">Paths</a>	CGGTTS, counter data, processing log, receiver data, RINEX, tmp
<a href="#">Receiver</a>	configuration, elevation mask, logger, logger options, manufacturer, model, observations, port, pps offset, synchronization, pps synchronization delay, status file, timeout, version
<a href="#">Reference</a>	file extension, logging interval, log path, log status, oscillator, power flag, status file
<a href="#">RINEX</a>	agency, create, observer, version

Table 5.2: Summary of `gpscv.conf` entries

### 5.4.2 Time-interval counter

This text file records the difference between GNSS receiver and the Reference Oscillator 1 pps, measured each second. Entries are successive lines of the form:

```
<time_of_day> <time_difference>
```

where the time difference is in seconds.

*Example:*

```
00:00:04 +4.0821E-006
```

## 5.5 `gpscv.conf` - the core configuration file

A single configuration file, `gpscv.conf`, provides configuration information to most of the OpenTTP software. `gpscv.conf` is used by `mktimex`, receiver logging scripts, TIC logging scripts, receiver utilities and so on.

It uses the format described in [5.3](#).

### 5.5.1 [Antenna] section

Entries used to create the RINEX header are:

- antenna number
- antenna type
- delta H, delta E, delta N
- marker name
- marker number
- X,Y,Z

Entries used to create the CGGTTS header are:

- X,Y,Z

#### **antenna number**

This appears as ANTNUM in the RINEX header.

*Example:*

```
antenna number = A567456
```

#### **antenna type**

This appears as ANTTYPE in the RINEX header.

*Example:*

```
antenna type = Ashtec
```

#### **delta H**

This appears as DELTA H in the RINEX header.

*Example:*

```
delta H = 0.0
```

#### **delta E**

This appears as DELTA E in the RINEX header.

*Example:*

```
delta E = 0.0
```

#### **delta N**

This appears as DELTA N in the RINEX header.

*Example:*

```
delta N = 0.0
```

**frame**

This appears as FRAME in the CGGTTS header.

*Example:*

```
frame = ITRF2010
```

**marker name**

This appears as MARKER NAME in the RINEX header.

*Example:*

```
marker name =
```

**marker number**

This appears as MARKER NUMBER in the RINEX header.

*Example:*

```
marker number =
```

**X**

This appears as X in the CGGTTS header and APPROX POSITION XYZ in the RINEXheader.

*Example:*

```
X = +4567890.123
```

**Y**

This appears as Y in the CGGTTS header and APPROX POSITION XYZ in the RINEXheader.

*Example:*

```
Y = +2345678.90
```

**Z**

This appears as Z in the CGGTTS header and APPROX POSITION XYZ in the RINEXheader.

*Example:*

```
Z = -1234567.890
```

### 5.5.2 [CGGTTS] section

Entries in this section control the format and content of CGGTTS files and filtering applied to CGGTTS tracks in the final output.

To create CGGTTS output you must enable it:

**outputs**

This defines a list of sections which in turn define the desired CGGTTS

outputs.

*Example:*

```
outputs = CGGTTS-GPS-C1,CGGTTS-GPS-P1,CGGTTS-GPS-P2
```

A CGGTTS v2E header looks like:

```
CGGTTS GENERIC DATA FORMAT VERSION = 2E
REV DATE = 2018-03-20
RCVR = NVS NV08 undefined 1999 mktimetx,v0.1.4
CH = 32
IMS = 99999
LAB = NMIA
X = -4648239.852 m
Y = +2560635.623 m
Z = -3526317.023 m
FRAME = ITRF2008
COMMENTS = none
INT DLY = 0.0 ns (GPS C1) CAL_ID = none
CAB DLY = 0.0 ns
REF DLY = 0.0 ns
REF = UTC(AUS)
CKSUM = 1A
```

Entries in the header than can be defined in the CGGTTS section are as below.

#### **comments**

This defines a single COMMENTS line in the CGGTTS header. The output line will be truncated at 128 characters.

*Example:*

```
comments = none
```

#### **lab**

This defines the LAB line.

*Example:*

```
lab = NMI
```

#### **reference**

This defines REF in the CGGTTS header.

*Example:*

```
reference = UTC(XXX)
```

#### **revision date**

This defines REV DATE in the CGGTTS header. It must be in the format

YYYY-MM-DD.

*Example:*

```
revision date = 2015-12-31
```

#### **create**

This defines whether or not CGGTTS files will be generated by `mktimetx`.

*Example:*

```
create = yes
```

#### **ephemeris**

This defines whether to use the receiver-provided ephemeris or a user-provided ephemeris (via a RINEX navigation file). If a user-provided ephemeris is specified then `ephemeris path` and `ephemeris file` must also be specified.

*Example:*

```
ephemeris = receiver
```

#### **ephemeris file**

This specifies a pattern for user-provided RINEX navigation files. Currently, only patterns of the form `XXXXddd0.yyn` are recognized.

*Example:*

```
ephemeris file = brdcddd0.yyn
```

#### **ephemeris path**

This specifies the path to user-provided RINEX navigation files.

*Example:*

```
ephemeris path = igsproducts
```

#### **lab id**

This defines the two-character lab code used for creating BIPM-style file names, as per the V2E specification.

*Example:*

```
lab id = AU
```

#### **maximum DSG**

CGGTTS tracks with DSG lower than this will be filtered out. The units are ns.

*Example:*

```
maximum DSG = 10.0
```

#### **minimum elevation**

CGGTTS tracks with elevations lower than this will be filtered out. The

units are degrees.

*Example:*

```
minimum elevation = 10
```

#### **minimum track length**

CGGTTS tracks shorter than this will be filtered out. Tracks meeting the criterion are not necessarily contiguous. The units are seconds.

*Example:*

```
minimum track length = 390
```

#### **maximum URA**

GPS ephemerides with URA greater than this will not be used. They will still be written to RINEX navigation files, however.

*Example:*

```
maximum URA = 3.0
```

#### **naming convention**

Defines the CGGTTS file naming convention. Valid options are ‘plain’ (MJD.ctf) and ‘BIPM’ styles. The `lab id` and `receiver id` should be defined in conjunction with BIPM-style filenames.

*Example:*

```
naming convention = BIPM
```

#### **receiver id**

This defines the two-character receiver code used for creating BIPM-style file names, as per the V2E specification.

*Example:*

```
receiver id = 01
```

#### **version**

This defines the version of CGGTTS output. Valid versions are v1 and v2E. The `lab id` and `receiver id` should be defined in conjunction with v2E output

*Example:*

```
version = v2E
```

### **CGGTTS output sections**

Multiple CGGTTS outputs can be defined, allowing for different constellation and signal combinations. An example of a CGGTTS output section is as follows:

CGGTTS name	RINEX name
C1	C1C
P1	C1P
E1	C1C
B1	C2I
C2	C2C
P2	C2P
B2	C7I

Table 5.3: Correspondence of CGGTTS and RINEX signal names.

```
[CGGTTS-GPS-C1]
constellation=GPS
code=C1
path=cggtts
BIPM cal id = none
internal delay = 11.0
```

#### **BIPM cal id**

This defines CAL\_ID for the internal delay, as used in v2E CGGTTS headers.

*Example:*

```
BIPM cal id = none
```

#### **code**

CGGTTS codes can be specified using either the two letter code, for example 'C1' as described in the CGGTTS V2E specification, or three letter RINEX v3.03 observation codes. *Example:*

```
code = P1+P3
```

#### **constellation**

This defines the GNSS constellation. Only GPS is supported currently for CGGTTS generation.

*Example:*

```
constellation = GPS
```

#### **internal delay**

This defines INT\_DLY in the CGGTTS header. The units are ns.

*Example:*

```
INT DLY = 0.0
```

**path**

This defines the directory in which output files are placed.

*Example:*

```
path = cggttts
```

In v1 CGGTTS files, the delays are specified via INT DLY, CAB DLY and ANT DLY. For v2E files, the delays may be specified via the ‘system delay’ and ‘total delay’. If multiple delays (eg both internal and system delay are present) are defined in `gpscv.conf`, the precedence order is internal, system and then total delay.

The entries used to specify system and total delay are:

**system delay**

This defines SYS DLY in the CGGTTS header. The units are ns.

*Example:*

```
SYS DLY = 0.0
```

**total delay**

This defines TOT DLY in the CGGTTS header. The units are ns.

*Example:*

```
TOT DLY = 0.0
```

### 5.5.3 [Counter] section

**file extension**

This defines the extension used for time interval measurement files. The default is ‘tic’.

*Example:*

```
file extension = tic
```

**configuration**

This sets a configuration file for the counter. Its contents are defined by the calling script. In the case of HP5313x counters for example, it lists SCPI commands sent to initialize the counter.

*Example:*

```
configuration = etc/hp53131.conf
```

**flip sign**

The processing software eg `mktimetx` assumes that TIC measurements are started by the reference and stopped by the GNSS receiver. If you need



to invert the sign of the measurements, set the option 'flip sign' to 'yes',  
*Example:*

```
flip sign = no
```

### **GPIB address**

For GPIB devices, the GPIB address must be defined.

*Example:*

```
GPIB address = 3
```

### **GPIB converter**

This defines the GPIB interface converter (eg RS232 to GPIB). Currently the only valid values is 'Micro488'.

*Example:*

```
GPIB converter = Micro488
```

### **header generator**

A header for the log file can be optionally added to the log file, using the output of a user provided script. Output should be to `stdout`. Each line will have a “#” automatically prepended to it.

*Example:*

```
header generator = bin/myticheader.pl
```

### **lock file**

This defines the device lock file, used to prevent multiple instances of the logger from being started.

*Example:*

```
lockfile = okxem.gpsc.v.lock
```

### **logger**

This defines the counter logging script.

*Example:*

```
logger = okxemlog.pl
```

### **logger options**

This defines options passed to the counter logging script.

*Example:*

```
logger options =
```

### **mode**

This defines the operating mode of the counter. Valid values are `timestamp` and `time interval`. Currently, this is only used by the TAPR TICC.

*Example:*

```
mode = timestamp
```

### **okxem channel**

The OpenTTP counter is multi-channel so the channel to use (1-6) must be specified.

*Example:*

```
okxem channel=3
```

### **port**

This defines the port used to communicate with the counter. It's value depends on the type of counter. For the XEM6001, it's a Unix socket. For serial devices, it's a device name like `/dev/ttyUSB0`.

*Example:*

```
# this is the port used by okcounterd  
port = 21577
```

### **timestamp format**

This controls the format of timestamps used in the counter log file. Valid values are `unix` and `time of day`. Currently, this is only used by the TAPR TICC.

*Example:*

```
timestamp format = time of day
```

## 5.5.4 [Misc] section

### **gzip**

Defines the compression/decompression program used in conjunction with counter and receiver log files.

*Example:*

```
gzip = /bin/gzip
```

## 5.5.5 [Delays] section

### **antenna cable**

This is ANT DLY as used in the CGGTTS header. Units are ns.

*Example:*

```
antenna cable = 0.0
```

### **reference cable**

This is REF DLY as used in the CGGTTS header. Units are ns.

*Example:*

```
reference cable = 0.0
```

### 5.5.6 [Paths] section

Paths follow the rules described in .

#### **root**

Defines the root path to be used with all other paths, unless they are specified as absolute paths. As with other paths specified in `gpscv.conf`, it is interpreted as being relative to the user's home directory, unless prefaced with a `'/'`.

*Example:*

```
root = test/newreceiver
```

#### **CGGTTS**

Defines the default directory used for CGGTTS files. This is typically overridden by the output directory that can be defined in each CGGTTS output section.

*Example:*

```
CGGTTS = cggotts
```

#### **counter data**

Defines the directory used for TIC data files.

*Example:*

```
counter data = raw
```

#### **processing log**

Defines the directory where the `mktimetx` processing log is written.

*Example:*

```
processing log = logs
```

#### **receiver data**

Defines the directory used for GNSS receiver raw data files.

*Example:*

```
receiver data = raw
```

#### **RINEX**

Defines the directory used for RINEX files.

*Example:*

```
RINEX = rinex
```

### **tmp**

Defines the directory used for intermediate and debugging files.

*Example:*

```
tmp = tmp
```

### **onewire temp data**

Defines the directory used to write temperature logs.

*Example:*

```
onewiretemp data = raw/onewire
```

### **uucp lock**

Sets the directory used to write UUCP lock files. UUCP lock files are used with serial devices to prevent other processes (eg `minicom`) opening the serial port while it is in use. The default is `/var/lock` but this varies with the operating system and its version so you will need to check this.

*Example:*

```
# This is for Debian
uucp lock = /var/run/lock
```

### **rinex 1112**

Defines the directory used to write RINEX files with all observations, typically so that precise antenna coordinates can be obtained. This is only used with dual frequency Javad receivers.

*Example:*

```
rinex 1112 = rinex/1112
```

## **5.5.7 [Receiver] section**

### **configuration**

This specifies a file to be used to configure the receiver. Currently, it is only used with Javad receivers.

*Example:*

```
configuration = rx.cfg
```

### **elevation mask**

This sets an elevation mask for tracking os satellites - below this elevation, satellites are ignored. The units are degrees. This may not be implemented for all receivers.

*Example:*

```
elevation mask = 0
```

### **file format**

Data logged by the receiver can be logged in either OpenTTP format ('ottp') or its native binary format ('native'). However, data logged in native format cannot presently be used by `mktime`.

*Example:*

```
file format =  ottp
```

### **logger**

This is the script used to configure and log messages from the GNSS receiver.

*Example:*

```
logger =  jnslog.pl
```

### **logger options**

These are options passed to the receiver logging script.

*Example:*

```
logger options =
```

### **manufacturer**

This defines the manufacturer of the receiver. Together with the model and version, this sets how data from the receiver is processed. For a list of supported receivers see XX.

*Example:*

```
manufacturer =  Javad
```

### **model**

This is the receiver model. For a list of supported receivers see XX.

*Example:*

```
model =  HE_GGD
```

### **version**

This can be used to identify the firmware version in use, for example.

*Example:*

```
version =  2.6.1
```

### **observations**

This is a list of GNSS systems which will be tracked by the receiver. In some applications where a multi-GNSS receiver is used, it may be desirable to track only one GNSS system so this sets which system is used. More generally, low-end multi-GNSS receivers are typically only capable of tracking certain combinations of GNSS systems so this is used to select the required combination. *Example:*

```
observations = GPS
```

### **port**

This is the serial port used for communication with the receiver.

*Example:*

```
port = /dev/ttyS0
```

### **pps offset**

This is an offset programmed into the GNSS receiver. Its purpose is to ensure that the counter triggers correctly, particularly HP5313x counters, which will only trigger every two seconds if the start trigger slips slightly behind the stop trigger. Suitable values are determined by the long-term stability of the reference, compared with GPS. The units are ns.

*Example:*

```
pps offset = 3500
```

### **pps synchronization**

This is a Javad-specific option. The logging script will force a synchronization of the receiver's internal time scale with the input 1 pps.

*Example:*

```
pps synchronization = no
```

### **pps synchronization delay**

This is a Javad-specific option. Synchronization of the receiver's internal time scale with the input 1 pps is delayed for this time after reset of the receiver. The units are seconds.

*Example:*

```
pps synchronization delay = 300
```

### **status file**

The status file contains a summary of the current state of the receiver, typically at least the currently visible GNSS satellites. Other software, for example `lcdmonitor`, uses this information.

*Example:*

```
status file = logs/rx.status
```

### **timeout**

The logging script will time out and exit if no messages are received for this period.

*Example:*

```
timeout = 60
```

### **time-transfer**

Receivers can be configured for non-time transfer operation. Typically, in this mode fewer messages are enabled and logged.

*Example:*

```
time-transfer = yes
```

### **sawtooth phase**

This defines which pps the sawtooth correction is to be applied to. Valid values are 'current second', 'next second' and 'receiver specified'. 'Current' means for the pps just generated.

*Example:*

```
sawtooth phase = current second
```

### **year commissioned**

The year the receiver was commissioned. This is used in the CGGTTS header.

*Example:*

```
year commissioned = 1999
```

## **5.5.8 [Reference] section**

### **file extension**

This defines the extension used for Reference status logs.

*Example:*

```
file extension = .rb
```

### **logging interval**

This defines the interval between status file updates. The units are seconds.

*Example:*

```
logging interval = 60
```

### **log path**

This defines where status logs are written to.

*Example:*

```
log path = raw
```

### **log status**

This enables status logging of the Reference.

*Example:*

```
log status = yes
```

**oscillator**

This identifies the installed oscillator, so that device-specific handling can be implemented.

*Example:*

```
oscillator = PRS10
```

**power flag**

This defines the file used to flag that the Reference has lost power, and needs rephasing. Currently, this only has meaning for the PRS10. It is used ntpd to disable the refclock corresponding to the PRs10's 1 pps.

*Example:*

```
power flag = logs/prs10.pwr
```

**status file**

In the case of the PRS10, this consists of the six status bytes and sixteen ADC values.

*Example:*

```
status file = logs/prs10.status
```

### 5.5.9 [RINEX] section

Entries in this section control the format and content of RINEX files. RINEX observation and navigation files in version 2 and version 3 formats can be produced. RINEX observations

**agency**

This specifies the value of the AGENCY field which appears in RINEX observation file headers.

*Example:*

```
agency = MY AGENCY
```

**create**

This defines whether or not RINEX files will be generated.

*Example:*

```
create = yes
```

**force v2 name**

This forces a V2 name for V3 RINEX output.

*Example:*

```
force v2 name = no
```



### observations

Normally, only code observations are output by `mktimetx`. To output phase observations, set this to 'all'.

*Example:*

```
observations = code
```

### observer

This specifies the value of the `OBSERVER` field which appears in RINEX observation file headers. If the observer is specified as 'user' then the environment variable `USER` is used.

*Example:*

```
observer = user
```

### version

This specifies the version of the RINEX output. Valid versions are 2 and 3.

*Example:*

```
version = 2
```

## 5.6 mktimetx

`mktimetx` is the core OpenTTP application. It creates CGGTTS and RINEX-format time-transfer files.

In the RINEX files, the code measurements have been corrected for any offsets between the raw measurements and the output 1 pps, and then the difference between the external clock and output pps (obtained from the TIC measurements) is applied ie the raw code measurements are reported with respect to the external clock.

### 5.6.1 usage

When run with no arguments, `mktimetx` uses the default GPSCV processing configuration file `~/gpscv.conf` and processes data from the preceding day.

The command line options are

- configuration** `<file>` specify the configuration file
- debug** `<file>` turn on debugging to file. To debug to `stderr`, just use 'stderr'.
- disable-tic** disable the use of sawtooth-corrected counter/timer measurements
- help** show help

Section	Key
Antenna	antenna number, antenna type, <i>delta H</i> , <i>delta N</i> , <i>delta E</i> , frame, marker name, marker number, marker type, x, y, z
CGGTTS	<i>comments</i> , <i>create</i> , lab, lab id, <i>maximum dsq</i> , <i>minimum track length</i> , <i>naming convention</i> , outputs, receiver id, reference, revision date, version
Counter	<i>file extension</i> , <i>flip sign</i>
Delays	antenna cable, reference cable
Misc	<i>gzip</i>
Paths	cggfts, counter data, receiver data, <i>processing log</i> , rinex, <i>root</i> , tmp
Receiver	<i>file extension</i> , manufacturer, model, <i>observations</i> , pps offset, <i>sawtooth phase</i> , <i>version</i>
RINEX	agency, <i>create</i> , observer, version

Table 5.4: Summary of `gpscv.conf` entries used by `mktime tx`. Optional entries are italicised.

- m <MJD>** specify the mjd
- start <hh:hh:ss/hhmmss>** set the start time
- stop <hh:hh:ss/hhmmss>** set the stop time
- short-debug-message** print out shorter debugging messages
- sv-diagnostics** save raw measurements for each SV in a file. Each SV is in a separate file.
- timing-diagnostics** save timing diagnostics in a file
- verbosity <n>** set the level of debugging verbosity. Valid values are 1 to 4. A verbosity of 4 will create a file several hundred MB in size.
- version** print version information and exit

Example:

```
mktime tx --configuration test.conf -m 57803 --debug stderr
↪ --verbosity 1
```

runs `mktime tx` in debugging mode, writing to `stderr` using the configuration file `test.conf` and processing data for MJD 57803.

### 5.6.2 configuration file

`mktime tx` uses `gpscv.conf`. Keys used by it are listed in table 5.4.

### 5.6.3 log file

## 5.7 runmktimetx.pl

runmktimetx.pl provides a convenient way to process multiple days of data and to run any missed processing.

runmktimetx.pl uses gpsc.v.conf. There are no entries in gpsc.v.conf specific to runmktimetx.pl

runmktimetx.pl doesn't produce a log file.

### 5.7.1 usage

runmktimetx.pl is normally run as a cron job.

To run runmktimetx.pl on the command line, use

```
runmktimetx.pl [option] ... [Start MJD [Stop MJD]]
```

Start MJD and Stop MJD specify the range of MJDs to process. If a single MJD is specified, then data for that day is processed. If no MJD is specified, the previous day's data is processed.

The options are:

- a <file> extend check for missed processing back n days (the default is 7)
- c <file> use the specified configuration file
- d run in debugging mode
- h print help and exit
- x run missed processing
- v print version information and exit

## 5.8 runr2cggttsv8.py

runr2cggttsv8.py can be used to automate the running of r2cggttts a program commonly used for generating CGGTTS files from RINEX files. In particular, given an MJD, it will copy and rename RINEX observation and navigation files to the files needed by r2cggttts. CGGTTS output can be optionally renamed and moved. The current number of leap seconds is automatically set from the navigation file if this information is present. Otherwise, it can be specified on the command line.

Current limitations are that V3 file names assume both mixed observation and navigation files. There is also no control over renaming of CGGTTS files.

### 5.8.1 usage

```
runr2cggttsv8.py [option] ... obsSTA MJD obsPath navSTA
    ↪ navPath paramsFile cggttspath
```

The required arguments are:

**obsSTA** the station name for RINEX observation files  
**MJD** the MJD to process  
**obsPath** the path to RINEX observation files  
**navSTA** the station name for RINEX navigation files  
**navPath** the path to RINEX navigation files  
**paramsFile** the paramCGGTTS file needed by `r2cggttsv8`  
**cggttspath** the output path for CGGTTS files

The command line options are

**--debug, -d** enable debugging  
**--help, -h** show help  
**--leapsecs <n >** set the number of leap seconds  
**--rename, -r** rename CGGTTS output  
**--rinexversion <2,3 >** set the RINEX version for constructing file names  
**--version, -v** print version information and exit

Example:

```
runr2cggttsv8 --version 3 CEDU00AUS_R 58545 ../RINEX
    ↪ CEDU00AUS_R ../RINEX CEDU.paramCGGTTS.dat cggttspath/
    ↪ CEDU
```

will attempt to use the RINEX files

```
../RINEX/CEDU00AUS_R_20190620000_01D_30S_MO.rnx
../RINEX/CEDU00AUS_R_20190630000_01D_30S_MO.rnx
../RINEX/CEDU00AUS_R_20190620000_01D_MN.rnx
../RINEX/CEDU00AUS_R_20190630000_01D_MN.rnx
```

If RINEX files for the day after the specified MJD are not present, processing will continue.

## 5.9 cnt9xlog.py

Pendulum CNT-90 and CNT-91 counters are supported via USBTMC (in particular, the Python `usbtmc` module).

### 5.9.1 usage

```
cnt9xlog.py [option] ...
```

The command line options are:

- c <file> use the specified configuration file
- d run in debugging mode
- h print help and exit
- v print version information and exit

### 5.9.2 configuration file

There is an optional file `cnt9x.cmds` which lists SCPI commands used to configure the counter. This file overrides the default configuration:

```
:SENS:TINT:AUTO OFF
:SENS:FUNC 'TINT 1,2'
:INP1:COUP DC # coupling DC
:INP2:COUP DC
:INP1:IMP 50 # impedance 50 ohms
:INP2:IMP 50
:INP1:LEVEL 1.0
:INP2:LEVEL 1.0
:INP1:SLOPE POS
:INP2:SLOPE POS
```

## 5.10 hp5313xlog.pl

HP and Agilent 53131 and 53132 counters are supported, in combination with the IOtech GPIB to RS232 converter.

### 5.10.1 usage

```
hp5313xlog.pl [option] ...
```

The command line options are:

- c <file> use the specified configuration file
- d run in debugging mode
- h print help and exit
- v print version information and exit

## 5.10.2 configuration file

There is a file `hp5313x.cmds` which lists the SCPI commands used to configure the counter. For example:

```
:FUNC 'TINT 1,2' # time interval
:SENS:EVENT1:LEVEL:ABS 1.0 # trigger level 1 volt
:SENS:EVENT2:LEVEL:ABS 1.0 #
:SENS:EVENT1:SLOP POS # trigger on positive slope
:SENS:EVENT2:SLOP POS
:INP1:ATT 1 # input attenuation x1
:INP2:ATT 1
:INP1:COUP DC # coupling DC
:INP2:COUP DC
:INP1:IMP 50 # impedance 50 ohms
:INP2:IMP 50
```

It has the following specific configuration file entries in `gpscv.conf`:

- [counter:configuration](#)
- [counter:gpib address](#)
- [counter:gpib converter](#)

## 5.11 okxemlog.pl

The OpenTTP reference platform includes a multi-channel TIC and this script communicates with the daemon `okcounterd` via a local TCP/IP socket. The script will exit if no data are returned for more than two minutes.

### 5.11.1 usage

```
okxemlog.pl [option] ...
```

The command line options are:

- c** `<file>` use the specified configuration file
- d** run in debugging mode
- h** print help and exit
- v** print version information and exit

It has the following specific configuration file entries in `gpscv.conf`:

- [counter::okxem channel.](#)

## 5.12 prs10log.pl

The Stanford PRS10 rubidium standard has a 1 pps input port that is typically used to lock the PRS10 to a GNSS receiver. The PRS10 timetags each input 1 pps with respect to its own 1 pps and can report these measurements. It can thus be also be used as a time-interval counter. In this application, the lock to the input 1 pps is disabled (ie the PRS10 is left free-running), and the 1 pps measurements are used for time-transfer.

### 5.12.1 usage

```
prs10log.pl [option] ...
```

The command line options are:

- c <file> use the specified configuration file
- d run in debugging mode
- h print help and exit
- v print version information and exit

The PRS10 is also used as the system time reference so it has entries in `gpscv.conf` associated with this:

- `reference::log status`
- `reference::logging interval`
- `reference::log path`
- `reference::file extension`
- `reference::power flag`
- `reference::status file`

## 5.13 ticclog.py

`ticclog.py` is used to log time interval measurements from a TAPR Time-stamping/Time Interval Counter (TICC). See [https://www.tapr.org/kits\\_ticc.html](https://www.tapr.org/kits_ticc.html) for more information about this counter.

### 5.13.1 usage

```
ticclog.py [option] ...
```

The command line options are:

- help, -h** print help and exit
- config, -c <file>** use the specified configuration file
- debug, -d** run in debugging mode
- settings, -s** print the counter settings
- version, -v** print version information and exit

It has the following specific configuration file entries in `gpscv.conf`:

- `counter::mode`
- `counter::timestamp format`

When the TICC is in timestamp mode, measurements are split into two files.

## 5.14 Javad/Topcon receivers

### 5.14.1 jnslog.pl

`jnslog.pl` is used to configure and log Javad GPS Receiver Interface Language (GRIL) receivers.

It doesn't produce a log file.

GRIL receivers are obsolete and further development of the software described here has ceased.

#### usage

```
jnslog.pl [option] ... configurationFile
```

The command line options are:

- h** print help and exit
- d** run in debugging mode
- r** suppress the reset of the receiver on startup (NOTE! automatic reset)
- v** print version information and exit

#### configuration

The file `receiver.conf` lists the commands used to configure the receiver. For example, to configure a single frequency receiver:



```

# GRIL commands
set,out/elm/dev/ser/a,$Init{receiver:elevation mask}
set,pos/iono,off
set,pos/tropo,off
set,dev/pps/a/time,gps
set,dev/pps/a/offns,$Init{receiver:pps offset}
set,dev/pps/a/per/ms,1000
set,dev/pps/a/out,on

# Receiver messages to turn on
RT # Receiver time
TO # Receiver base time to receiver time offset
ZA # PPS offset (sawtooth)
YA # Extra time offset
SI # Satellite index
EL # Satellite elevation
AZ # Satellite azimuth
SS # Satellite navigation status
FC # C/A lock loop status bits
F1
F2
RC # Full pseudorange C/A
R1 # Full pseudorange P/L1
R2 # Full pseudorange P/L2
P1 # For RINEX Obs
P2 # For RINEX Obs
RD # For RINEX
NP # Navigation Position text message
UO:{3600,14400,0,2} # UTC parameters, when changes or
    ↔ every 4 hrs
IO:{3600,14400,0,2} # Ionospheric parameters
GE:{1,3600,0,2} # GPS ephemeris data
```

The GRIL commands in the first part of the file are executed verbatim, with substitution of values from the configuration file, written as Perl hash table lookups. Messages to be enabled are then listed, one per line. The message rate can be specified in the GRIL syntax.

Configuration commands in `gpscv.conf` specific to this receiver are:

JNS receivers have a number of specific configuration entries in `gpscv.conf`:

- [receiver::configuration](#)
- [receiver::pps synchronization](#)

- [receiver::pps synchronization delay](#)
- [paths::rinex l1l2](#)

### 5.14.2 `jnsextract.pl`

`jnsextract.pl` is used to decode and extract information from Javda receiver log files. It will decompress the file if necessary. It doesn't use `gpscv.conf`.

#### usage

```
jnsextract.pl [option] ... file
```

The command line options are:

- b** <value> start time (s or hh:mm, default 0)
- c** compress (eg skip repeat values) if possible
- e** <value> stop time (s or hh:mm, default 24:00)
- g** select GPS only
- i** <n> decimation interval (in seconds, default=1)
- k** keep the uncompressed file, if created
- o** <mode> output mode:
  - bp** receiver sync to reference time (BP message)
  - cn** carrier-to-noise vs elevation
  - do** DO derivative of time offset vs time
  - si** visibility vs time (SI message)
  - np** visibility vs time (NP message)
  - vt** visibility vs time (list time for each SV)
  - tv** visibility vs time (list SV for each time)
  - tr** satellite tracks (PRN, azimuth, elevation)
  - tn** satellite tracks (PRN, azimuth, elevation, CN)
  - to** TO time offset vs time
  - ya** YA time offset vs time
  - za** ZA time offset vs time
  - st** ST solution time tag vs time
  - uo** UO UTC parameters
- r** select GLONASS only

### 5.14.3 `runrinexobstc.pl`

`runrinexobstc.pl` runs the now deprecated `rinexobstc`, which produces a RINEX observation file suitable for precise positioning when used with a

suitable Javad receiver. `mktimetx` can now be configured to produce suitable output.

`runrinexobstc.pl` doesn't produce a log file.

### usage

`runrinexobstc.pl` is normally run as a cron job.

To run `runrinexobstc.pl` on the command line, use

```
runrinexobstc.pl [option] ... [Start MJD [Stop MJD]]
```

`Start MJD` and `Stop MJD` specify the range of MJDs to process. If a single MJD is specified, then data for that day is processed. If no MJD is specified, the previous day's data is processed.

The options are:

- `-a <file>` extend check for missed processing back `n` days (the default is 7)
- `-c <file>` use the specified configuration file
- `-d` run in debugging mode
- `-h` print help and exit
- `-x` run missed processing
- `-v` print version information and exit

### configuration

`runrinexobstc.pl` uses `gpscv.conf`.

It uses the following specific configuration file entries in `gpscv.conf`; most relate to writing the RINEX observation file header:

- `antenna::x`
- `antenna::y`
- `antenna::z`
- `antenna::marker name`
- `antenna::marker number`
- `antenna::delta e`
- `antenna::delta h`
- `antenna::delta n`

- [antenna::antenna number](#)
- [antenna::antenna type](#)
- [rinex::agency](#)
- [rinex::observer](#)
- [paths::rinex l1l2](#)

## 5.15 NVS NV08C receivers

### 5.15.1 nv08log.pl

`nv08log.pl` is used to configure and log NVS NV08 receivers. It needs the Perl library `NV08C`.

It doesn't produce a log file.

There are no NV08-specific configuration commands in `gpscv.conf`

#### usage

```
nv08log.pl [option] ...
```

The command line options are:

- c <file> use the specified configuration file
- h print help and exit
- d run in debugging mode
- r reset the receiver on startup
- v print version information and exit

### 5.15.2 nv08extract.pl

`nv08extract.pl` is used to decode and extract information from NVS NV08 receiver log files. It will decompress the file if necessary.

It uses `gpscv.conf` to construct receiver log file names if the file is not explicitly given.

#### usage

```
nv08extract.pl [option] ... [file]
```

Given an MJD via a command line option, it will construct a file name using the information in `gpscv.conf`. If no MJD or file is given, it assumes the current MJD for the file.

The command line options are:

- c **<file>** use the specified configuration file
- h print help and exit
- d run in debugging mode
- v print version information and exit
- m **<MJD>** MJD of the file to process
- t extract Time, Date and Time Zone offset
- o extract Receiver Operating Parameters
- s extract Visible Satellites
- n extract Number of Satellites and Dilution Of Precision (DOP)
- w extract Software Version, Device ID and Number of Channels
- f extract 1 PPS 'sawtooth' correction
- T extract Time Synchronisation Operating Mode (antenna cable delay, averaging time)
- p extract PVT Vectors and associated quality factors (including TDOP)
- a extract Additional Operating Parameters
- P extract Port status messages
- e extract Satellite ephemeris
- l extract Time scale parameters
- i extract Ionosphere parameters
- g extract GPS, GLONASS and UTC time scale parameters
- r extract Raw data (pseudoranges, etc)
- G extract Geocentric antenna coordinates in WGS-84 system
- u extract Unknown message (garbage data)
- z less verbose output

### 5.15.3 `nv08info.pl`

This actually does nothing useful. If it did, it would query the receiver for its serial number and so on.

#### usage

```
nv08info.pl [option] ...
```

The command line options are:

- c **<file>** use the specified configuration file
- h print help and exit
- d run in debugging mode

**-v** print version information and exit

## 5.16 Trimble Resolution T receivers

Trimble Resolution T receivers are obsolete and further development of the software described here has ceased. The successor to the Resolution T, the SMT 360, cannot currently be used for time-transfer because of changes in the message set. It is possible though that a future TSIP-based receiver may be suitable for time-transfer.

### 5.16.1 restlog.pl

#### usage

```
restlog.pl [option] ...
```

The command line options are:

- c** <file> use the specified configuration file
- h** print help and exit
- d** run in debugging mode
- r** reset the receiver on startup
- v** print version information and exit

#### configuration

restlog.pl respects the `receiver::model` configuration option. The valid values are:

1. Resolution T
2. Resolution SMT 360

### 5.16.2 restextract.pl

restextract.pl is used to decode and extract information from Trimble receiver log files. It will decompress the file if necessary.

#### usage

```
restextract.pl [option] ...
```

Given an MJD via a command line option, it will construct a file name using the information in `gpscv.conf`. If no MJD is given, it assumes the current MJD for the file.

The command line options are:

- c <file> use the specified configuration file
- h print help and exit
- v print version information and exit
- a extract S/N for visible satellites
- f show firmware version
- l leap second warning
- L leap second info )
- m <mjd> MJD of the file to process
- p position fix message
- r <svn> pseudoranges for satellite with given SVN (svn=999 reports all satellites)
- s number of visible satellites
- t temperature
- u UTC offset
- x alarms and gps decoding status

### 5.16.3 restinfo.pl

`restinfo.pl` communicates with the receiver configured in `gpscv.conf`, polling it for information such as hardware, software and firmware versions. The serial port communication speed must be 115200 baud.

#### usage

```
restinfo.pl [option] ...
```

The command line options are:

- c <file> use the specified configuration file
- h print help and exit
- d run in debugging mode
- v print version information and exit

### 5.16.4 restconfig.pl

`restconfig.pl` is used to configure the receiver serial port for 115200 baud and no parity bit. The latter is necessary for using the SMT 360 with `ntpd`. The new configuration is written to flash memory.

### usage

```
restconfig.pl [option] ...
```

The command line options are:

- c <file> use the specified configuration file
- h print help and exit
- d run in debugging mode
- v print version information and exit

The serial port device name is read from `gpscv.conf` or the specified file.

### 5.16.5 restplayer.pl

This is used to replay raw data files through a serial port, simulating the operation of the GNSS receiver. This is useful for testing the operation of `ntpd`, for example. This script will have to be modified for individual use because it currently hard codes paths and device names.

## 5.17 ublox receivers

### 5.17.1 ublox9log.py

This python3 script is used with ublox series 9 receivers like the ZED-F9P and ZED-F9T. The following messages are enabled for 1 Hz output:

- RXM-RAWX
- TIM-TP
- NAV-SAT (not logged)
- NAV-TIMEUTC
- NAV-CLOCK

A status file is written once per minute, containing the SV identifiers of tracked satellites with code and time synchronization flags set.

### usage

```
ublox9log.py [option] ...
```

The command line options are:

- help, -h print help and exit



**--config, -c <file>** use the specified configuration file  
**--debug, -d** run in debugging mode  
**--reset, -r** reset the receiver before configuration  
**--version, -v** print version information and exit

### configuration

`ublox9log.pl` respects the `receiver::model` configuration option. The valid values are:

1. ZED-F9

### 5.17.2 ubloxlog.pl

This script used to configure and log series 8 receivers like the NEO-M8T.

#### usage

```
ubloxlog.pl [option] ...
```

The command line options are:

**-h** print help and exit  
**-c <file>** use the specified configuration file  
**-d** run in debugging mode  
**-r** reset the receiver before configuration  
**-v** print version information and exit

### 5.17.3 ubloxextract.py

`ubloxextract.py` is used to decode and extract information from ublox receiver log files written in OpenTTP format. It will decompress the file if necessary.

It uses `gpscv.conf` to construct receiver log file names if the file is not explicitly given.

#### usage

```
ubloxextract.py [option] ... [file]
```

Given an MJD via a command line option, it will construct a file name using the information in `gpscv.conf`. If no MJD or file is given, it assumes the current MJD for the file.

The command line options are:

- help, -h** print help and exit
- config, -c <file>** use the specified configuration file
- debug, -d** run in debugging mode
- version, -v** print version information and exit
- mjd, -m <MJD>** MJD of the file to process
- uniqid** show chip id
- monver** show hardware and software versions
- navclock** extract nav-clock message
- navsat** extract nav-sat message
- navtimeutc** extract nav-timeutc message
- rawx** extract raw measurement data
- timtp** extract sawtooth correction

### 5.17.4 ubloxmkdev.py

`ubloxmkdev.py` is used for creating a device name for a connected receiver. It is useful, for example, when multiple receivers are connected and unique USB device identification is not possible. It works by connecting to the receiver and querying its serial number. A configuration file defines the device name for that serial number.

The script is typically called by `udev`.

#### usage

```
ubloxmkdev.py [option] ... dev
```

The command line options are:

- help, -h** print help and exit
- debug, -d** run in debugging mode
- version, -v** print version information and exit

#### configuration

The configuration file is `/usr/local/etc/ublox.conf`. It looks like:

```
ce92fa1422 ublox1
77435b1763 ublox2
```

Each line consists of the receiver's serial number and the device name to be associated with it.

## 5.18 Miscellaneous tools

### 5.18.1 cggttsc.py

cggttsc.py performs various checks on CGGTTS files. Running it on a single CGGTTS file with no options will produce output like:

File	Tracks	Short	min SV	max SV	DSG	elv
57401.cctf	706	71	3	11	0	0

where:

'Tracks' is the total number of tracks

'Short' is the number of tracks with length less than a specified threshold (default 780 s)

'min SV' is the minimum number of SVs visible

'max SV' is the maximum number of SVs visible

'DSG' is the number of tracks with DSG above a specified threshold (default 20 ns)

'elv' is the number of tracks with elevation below a specified threshold (default 10 degrees)

#### usage

```
cggttsc.py [OPTION] ... file [file ldots]
```

The command line options are:

**--help,-h** print the help information and exit

**--debug,-d** print debugging information to stderr

**--nowarn** suppress warnings (eg about missing files, bad formatting in CGGTTS files)

**--dsg <value>** set the upper limit for acceptable DSG. The units are ns.

**--elevation <value>** set the lower limit for acceptable elevation. The units are degrees.

**--tracklength <value>** set the lower limit for acceptable track length. The units are seconds.

**--checkheader** show when significant fields in the header change, for example the delays.

**--nosequence** do not interpret (two) input filenames as a sequence.

**--version,-v** show the version information and exit

### examples

Multiple files can be checked and a sequence can be specified with two file names. For example:

```
cggttsqc.py --dsg 5 GZAA0157.834 GZAA0157.876
```

will report on all files between MJDs 57834 and 57876, indicating in the DSG field the number of tracks with DSG greater than 5 ns. For the two file names to be interpreted as a sequence, they must:

1. have names in v2E CGGTTS recommended format or in the format MJD.xxx
2. be in the same directory
3. have the same extension, if in the format MJD.xxx

### 5.18.2 cmpcggtts.py

cmpcggtts.py matches tracks in paired CGGTTS files. It can be used for time transfer and delay calibration.

#### usage

```
cmpcggtts.py [option] ... refDir calDir firstMJD lastMJD
```

The command line options are:

- help** show this help message and exit
- starttime** <time> time of day in HH:MM:SS format to start processing (default 00:00:00)
- stoptime** <time> time of day in HH:MM:SS format to stop processing (default 23:59:00)
- elevationmask** <value> elevation mask (in degrees, default 0.0)
- mintracklength** <value> minimum track length (in s, default 750)
- maxdsg** <value> maximum DSG (in ns, default 20.0)
- matchephemeris** match on ephemeris (default no)
- cv** compare in common view (default)
- aiv** compare in all-in-view
- acceptdelays** accept the delays (no prompts in delay calibration mode)
- delaycal** delay calibration mode
- timetransfer** time-transfer mode (default)

**--ionosphere** use the ionosphere in delay calibration mode (default = not used)  
**--checksum** exit on bad checksum (default = warn only)  
**--refprefix <value>** file prefix for reference receiver (default = MJD)  
**--calprefix <value>** file prefix for calibration receiver (default = MJD)  
**--refext <value>** file extension for reference receiver (default = cctf)  
**--calext <value>** file extension for calibration receiver (default = cctf)  
**--debug, -d** debug (to stderr)  
**--nowarn** suppress warnings  
**--quiet** suppress all output to the terminal  
**--keepall** keep tracks after the end of the day  
**--version, -v** show version and exit

The default mode is time-transfer. In this mode, a linear fit to the time-transfer data is made and the fractional frequency error and REFSYS (evaluated at the midpoint of the data set) are outputted. The uncertainties are estimated from the linear fit.

In delay calibration mode, the presumption is that the data are for two receivers sharing a common clock and operating on a short baseline. The ionosphere correction is removed by default but can be retained via a command line option. Delays as reported in the CGGTTS header can be changed interactively; prompting for the new delays can be skipped with a command line option.

Data can be filtered by elevation, track length and DSG.

Matching on ephemeris (IODE) can also be enforced. This can reduce time-transfer noise in some circumstances.

Three text files are produced:

### examples

Basic common-view time transfer with CGGTTS files in the folders **refrx** and **remrx**:

```
cmpcggtts.py refrx remrx 57555 57556
```

Delay calibration with filenames according to the CGGTTS V2E specification:

```
cmpcggtts.py --delaycal --refprefix GZAU01 --calprefix  
↪ GZAU99 refrx remrx 57555 57556
```

s

### 5.18.3 editcggts.py

editcggts.py is used to edit CGGTTS navigation files. The header checksum and the modification time are updated after editing,

#### usage

```
editcggts.py [option] ... filename [filename ...]
```

The command line options are:

- help,-h** print the help information and exit
- debug,-d** print debugging information to stderr
- comments <value>** set comment
- output,-o <value>** output to file/directory
- tmp** output is to file(s) with .tmp added to name
- replace, -r** replace the edited file(s)
- nosequence** do not interpret (two) input file names as a sequence
- nowarn** suppress warnings
- version,-v** show the version information and exit

The **--output**, **--tmp** and **--replace** options are mutually exclusive. If none of these options are used, output will be to **stdout**.

If two filenames are given without the **--nosequence** option, they will be interpreted as specifying a sequence of files to be processed.

### 5.18.4 editrxnav.py

editrxnav.py is used to edit RINEX navigation files. Currently, it doesn't do much but it will eventually do much more.

#### usage

```
editrxnav.py [option] ... file
```

The command line options are:

- help,-h** print the help information and exit
- debug,-d** print debugging information to stderr
- output <value>** output to file/directory
- replace, -r** replace edited file
- ura, -u <value>** remove entries with URA greater than this
- version,-v** show the version information and exit

### 5.18.5 editrxobs.py

`editrxobs.py` is used to edit RINEX observation files.

#### usage

```
editrxobs.py [option] ... file [file ...]
```

The command line options are:

- help, -h** print the help information and exit
- debug, -d** print debugging information to stderr
- output, -o <file>** output to file/directory
- keep, -k** keep intermediate files
- replace, -r** replace edited file
- system <system>** gnss system (BeiDou, Galileo, GPS, GLONASS)
- obstype <obstype>** observation type (C2I, L2I, ...)
- fixms** fix ms ambiguities (ref RINEX file required)
- fixmissing** add observations missing at the beginning of the day
- sequence, -s** interpret input files as a sequence
- refrinex <file>** reference RINEX file for fixing ms ambiguities (name of first file if multiple input files are specified)
- version, -v** show the version information and exit

### 5.18.6 fetchigs.py

`fetchigs.py` uses the Python library `urllib` to download GNSS products and station observation files from IGS data centres. You will need a configuration file that sets up downloads from at least one IGS data centre. The sample configuration file should be sufficient for most uses.

#### usage

```
fetchigs.py [option] ... start [stop]
```

The `start` and `stop` times can be in the format:

- MJD** Modified Julian Date
- yyyy-doy** year and day of year (1 ...)
- yyyy-mm-dd** year, month (1-12) and day (1 ...)

The command line options are:

- help, -h** print help and exit
- config <file>** , -c use this configuration file

**--debug, -d** print debugging output to stderr  
**--outputdir <dir>** set the output directory  
**--ephemeris** get broadcast ephemeris  
**--clocks** get clock products (.clk)  
**--orbits** get orbit products (.sp3)  
**--rapid** fetch rapid products  
**--final** fetch final products  
**--centre <centre>** set data centres  
**--listcentres, -l** list available data centres  
**--observations** get station observations (only mixed observations for V3)  
**--statid <statid>** station identifier (eg V3 SYDN00AUS, V2 sydn)  
**--rinexversion <2,3>** rinex version of station observation files  
**--system <system>** gnss system (GLONASS,BEIDOU,GPS,GALILEO,MIXED)  
**--noproxy** disable use of proxy server  
**--proxy <proxy>** set your proxy server (server:port)  
**--version, -v** print version information and exit

### examples

This downloads V3 mixed observation files from the IGS station CEDU, with the identifier CEDU00AUS for days 10 to 12 in 2020.

```
fetchigs.py --centre CDDIS --observations --statid  
↳ CEDU00AUS --version 3 --system MIXED 2010-10  
↳ 2020-12
```

This downloads final IGS clock and orbit products for MJD 58606 from the CDDIS data centre.

```
fetchigs.py --centre CDDIS --clocks --orbits --final 58606
```

This downloads a brdc broadcast ephemeris file.

```
fetchigs.py --centre GSSC --ephemeris --system MIXED --  
↳ rinexversion 2 58606
```

### configuration file

The [Main] section has two keys.

#### Data centres

This is a comma-separated list of sections which define IGS data centres



which can be used to download data.

*Example:*

```
Data centres = CDDIS,GSSC
```

### **Proxy server**

This sets a proxy server (and port) to be used for downloads.

*Example:*

```
Proxy server = someproxy.in.megacorp.com:8080
```

Each defined IGS data centre has the following keys, defining various paths.

### **Base URL**

This sets the base URL for downloading files.

*Example:*

```
Base URL = ftp://cddis.gsfc.nasa.gov
```

### **Broadcast ephemeris**

This sets the path relative to the base URL for downloading broadcast ephemeris files.

*Example:*

```
Broadcast ephemeris = gnss/data/daily
```

### **Products**

This sets the path relative to the base URL for downloading IGS products.

*Example:*

```
Products = gnss/products
```

### **Station data**

This sets the path relative to the base URL for downloading IGS station RINEX files.

*Example:*

```
Station Data = gnss/data/daily
```

## **5.18.7 ticqc.py**

`ticqc` checks TIC files. It currently reports the total number of measurements, duplicates and gaps in the file, and the data range.

**usage**

ticqc.py [option] file

The command line options are:

**--help, -h** print help and exit

**--verbose** verbose output eg duplicate measurements are printed out

**--version, -v** print version information and exit

## 6. System software

### 6.1 dioctrl

`dioctrl` is used to provide access to a digital I/O port. On Intel platforms, the currently supported hardware includes the 8255 parallel port and the SIOF8186x. `dioctrl` is not currently functional on the ARM platform.

### 6.2 kickstart.pl

`kickstart.pl` is used to check that required processes are running, and restart them if necessary. It is used to start the receiver and counter logging processes, for example.

The lock file for each process (`target`) contains the running process ID; this is used by `kickstart.pl` to test whether the target is running.

`kickstart.pl` produces a log file in the user's home directory, `logs/kickstart.log`. Each time a process is checked, it touches the file `logs/kickstart.target.check`, where `target` is specified in the configuration file.

#### 6.2.1 usage

To run `kickstart.pl` on the command line, use:

```
kickstart.pl [OPTION] ...
```

The command line options are:

- c **FILE** use the specified configuration file
- d run in debugging mode
- h print help and exit
- v print version information and exit

### 6.2.2 configuration file

An example configuration file is:

```
targets = okxem
[okxem]
target = okxem
command = bin/okxemlog.pl
lock file = logs/okxem.gpscv.lock
```

#### **targets**

This defines a list of sections, each of which corresponds to a process to monitor.

*Example:*

```
targets = restlog,okxem
```

#### **target**

This defines a target identifier. It is used to construct filenames and as an identifier in logged messages.

*Example:*

```
target = okxem
```

#### **command**

This defines the command used to start the target. Options can be used.

*Example:*

```
command = bin/okxemlog.pl
```

#### **lock file**

This defines the lock file associated with a target.

*Example:*

```
lock file = logs/okxem.gpscv.lock
```

Paths specified in the configuration file are constructed using the usual rules.

## 6.3 mjd

mjd provides conversion between a calendar date and MJD and shows the current MJD.

### 6.3.1 usage

```
mjd [option]
```

The command line options are:

- d <DD MM YYYY> convert date to MJD
- h print help and exit
- m <MJD> convert MJD to date
- t print today's MJD and exit
- v print version information and exit

## 6.4 okcounterd

`okcounterd` provides the interface to the Opal Kelly FPGA development board when configured as a multi-channel counter. It communicates with user processes via port 21577 (it's the date "Star Wars" premiered). Multiple processes can communicate with the daemon so that logging processes can be separated.

`okcounterd` does not use a configuration file and does not produce a log file.

`okcounterd` recognizes the following commands, sent as plain text:

CONFIGURE GPIO 0|1 enables/disables the system GPIO.

CONFIGURE PPSSOURCE n selects the input channel of the counter which is routed to the output 1 pps.

QUERY CONFIGURATION reads the device configuration register. `okcounterd` sends a plain text response.

LISTEN registers a process to receive counter-timer readings.

`okcounterdctrl.pl` provides a convenient way to send commands.

Counter/timer data sent by `okcounterd` is in the following format:

```
channel_number timestamp (s) timestamp ( $\mu$ s) reading (ns)
```

Channel numbers are indexed from 1.

### 6.4.1 usage

`okcounterd` is automatically started by the system's init system. On Debian, this is `systemd`. It can be run manually for debugging purposes. Use:

```
okcounterd [OPTION] ...
```

The command line options are:

- b <file>load the specified bitfile (the full path is needed)
- d run in debugging mode
- h print help and exit
- v print version information and exit

To manually run `okcounterd`, you may need to disable the system service and kill any running `okcounterd` process.

## 6.5 `okcounterctl.pl`

`okcounterctl.pl` provides a convenient way to send commands to `okcounterd`.

`okcounterctl.pl` doesn't have a configuration file. It uses `gpscv.conf` to determine the port used by `okcounterd`.

`okcounterctl.pl` doesn't produce a log file.

### 6.5.1 usage

To run `okcounterctl.pl` on the command line, use:

```
okcounterctl.pl [OPTION] ...
```

The command line options are:

- c <file> use the specified `gpscv` configuration file
- d run in debugging mode
- g <0|1> disable/enable the system GPIO
- h print help and exit
- p <1...6> set the PPS OUT source
- q query the counter configuration
- v print version information and exit

channel	PPS signal
1	NavSpark
2	SMT360
3	NV08C
4	external pps
5	system time pps
6	GPIO
default	GPSDO

Table 6.1: PPS OUT channels for the `-p` option

Note that the default state is to output the GPSDO 1 pps.

## 6.6 okbloader

`okbloader` is used to load a bitfile to the XEM6001. Normally, `okcounterd` will load the bitfile it needs on startup. `okbloader` is mainly used during development and testing.

### 6.6.1 usage

```
okbloader [option] ... file
```

The command line options are:

- h print help and exit
- v print version information and exit

## 6.7 lcdmonitor

`lcdmonitor` runs the lcd display on the front panel of the unit.

`lcdmonitor` produces a log file `/usr/local/log/lcdmonitor.log` that records significant commands run from the front panel, for example reboots.

A lock file `/usr/local/log/lcdmonitor.lock` is used to prevent duplicate processes from running.

### 6.7.1 usage

`lcdmonitor` is automatically started by the system's init system. On Debian, this is `systemd`. It can be run manually for debugging purposes.

The command line options are:

- d run in debugging mode
- h print help and exit
- v print version information and exit

To run `lcdmonitor` on the command line, you will need to disable the entry in `/etc/inittab`, reread the `inittab` and kill any running `lcdmonitor` process.

### 6.7.2 configuration file

The configuration file for `lcdmonitor` is `/usr/local/etc/lcdmonitor.conf`. This file is only modifiable by the super-user. The file is divided into sec-

## 6. System software

---

tions, with section names delimited by square brackets [ ].

*Example:*

```
token = value
```

Comments begin with a # character. Entries in the various sections of the configuration file are given below.

### [General] section

#### **NTP user**

This defines the name of the user associated with NTP functions.

*Example:*

```
NTP user = ntp-admin
```

#### **Squealer config**

This specifies the location of the configuration file used by `squealer`, a program used to detect system problems.

*Example:*

```
Squealer config = /home/cvgps/etc/squealer.conf
```

### [UI] section

#### **Show PRNs**

This specifies whether or not to show the PRNs (or Space Vehicle identifiers) of GPS satellites being tracked by the receiver. If this is set to zero, then only the number of satellites tracked is displayed.

*Example:*

```
Show PRNs=1
```

#### **LCD intensity**

This sets the intensity of the LCD display. Valid values are 0 to 100.

*Example:*

```
LCD intensity=90
```

#### **LCD contrast**

This sets the contrast of the LCD display. Valid values are 0 to 100.

*Example:*

```
LCD contrast=95
```



## [GPSCV] section

### GPSCV user

This defines the name of the user associated with GPSCV functions.

*Example:*

```
GPSCV user = cvgps
```

### gpscv config

This defines the location of the file `gpscv.conf`.

*Example:*

```
GPSCV config = /home/cvgps/etc/gpscv.conf
```

### GPS restart command

This specifies the command used to restart the GPS receiver. Note that since `lcdmonitor` runs as `root`, the restart must be explicitly done as `cvgps`.

*Example:*

```
GPS restart command =su - cvgps -c '/home/cvgps/bin/  
↪ check_rx'
```

### GPS logger lock file

This specifies location of the lock file used by the GPS logging process. It is used to determine which process needs to be killed before a restart.

*Example:*

```
GPS logger lock file=/home/cvgps/logs/rx.lock
```

## [OS] section

### Reboot command

This specifies the command used to reboot the PC.

*Example:*

```
Reboot command = /sbin/shutdown -r now
```

### Poweroff command

This specifies the command used to shut down the PC.

*Example:*

```
Poweroff command = /sbin/shutdown -t 3 -h
```

### ntpd restart command

This specifies the command used to restart `ntpd`.

*Example:*

```
ntpd restart command = /sbin/service ntpd restart
```

## [Network] section

### DNS

*Example:*

```
DNS = /etc/resolv.conf
```

### Network

*Example:*

```
Network = /etc/sysconfig/network
```

### Eth0

*Example:*

```
Eth0 = /etc/sysconfig/network-scripts/ifcfg-eth0
```

## 6.8 libraries

### 6.8.1 libconfigurator

This is a C library used for parsing of configuration files. It is mainly used by `mktimetx`.

The configuration file is read using

```
int configfile_parse_as_list(ListEntry **first, const char
    ↪ *filename)
```

This returns a pointer to the first item in a linked list of `ListEntry`, which describe entries in the configuration file.

Functions are provided for searching the list for an entry by section and key (token) name, returning a value of the required type.

```
int list_get_int(ListEntry *first, const char *section,
    ↪ const char * token, int *value);
int list_get_float(ListEntry *first, const char *section,
    ↪ const char * token, float *value);
int list_get_double(ListEntry *first, const char *section,
    ↪ const char * token, double *value);
int list_get_string(ListEntry *first, const char *section,
    ↪ const char * token, char **value);
```

The return value of the function flags whether or not an error occurred. The last error to occur is determined by calling:

```
config_file_get_last_error()
```

The error codes are:

```
NoError  
FileNotFound  
SectionNotFound  
InternalError  
ParseFailed  
TokenNotFound
```

`list.clear()` can be used to delete the list.

This should be replaced one day by something based on the C++ STL!

### 6.8.2 TFLibrary.pm

This library is effectively deprecated because all new development is now in Python.

### 6.8.3 OpenOK2

This is a C++ library that can be used for communicating with the Opal Kelly FPGA board. It is a fork of a library originally written by Jennifer Holt and now maintained by Jorge Francisco. Opal Kelly provides a similar library for a number of platforms. When OpenTTP development began, ARM support was not available hence the use of OpenOK2. One significant limitation of the library is that it does not provide support for USB block transfers.

### 6.8.4 ottplib.py

This library replaces `TFLibrary.pm` and should be used for all new development.

```
ottplib.LoadConfig(path, options={})
```

Loads the configuration file specified by `path`, returning key/value pairs as a dictionary. Two options can be supplied:

- tolower** all keys are converted to lower case if this is `true`
- defaults** a default configuration file is loaded first

Example:

```
cfg=ottplib.LoadConfig(configFile,{'tolower':True})
```

`ottplib.MJD(time)`

Returns the Modified Julian Date, given the time in seconds since the epoch. It is usually called with `time.time()`

`ottplib.MakeAbsolutePath(path,root)`

Returns a path constructed relative to `root`, unless `path` is already absolute. A trailing slash is added if it is absent.

`ottplib.MakeAbsolutePath(file,home,defaultPath)`

Returns an absolute path to `file` according to the following rules

1. If `path` begins with a slash then it is returned unchanged.
2. If `path` ends with a slash, then `home` is prepended.
3. Otherwise, `defaultPath` is prepended.

`ottplib.CreateProcessLock(file)`

Creates a lock file named `file`. The contents of the file are the process name and its PID. When attempting to create a lock, the PID is checked in `proc`. Returns `True` if the lock was successfully created.

`ottplib.RemoveProcessLock(file)`

Removes the lock file named `file`. It returns nothing.

### 6.8.5 `cggttslib.py`

This library provides some functions for reading and manipulating CGGTTS files.

`cggttslib.CheckSum(string)`

This computes the checksum of `string` using the algorithm described in the CGGTTS specification. The checksum is returned as an integer.

`cggttslib.ReadHeader(file)`

This reads the CGGTTS header of `file` and returns the fields in a Python dictionary. Dictionary keys are the same as in the header. For example, `REV DATE` has the key `'rev date'` associated with it. The exceptions to this are the delays `INT DLY`, `SYS DLY` and `TOT DLY`. For convenience, these are further broken down.

The function takes one argument, a file name and returns a list containing the dictionary and any error messages as a single string. An empty dictionary is returned if:

1. the file cannot be opened

2. there is a formatting error in the header

If the calculated checksum does not match the value in the header, an error message is returned.

`cggttplib.MakeFileSequence(file1,file2)`

`MakeFileSequence` interpolates a sequence of file names between `file1` and `file2`. For example, if the two input file names are `GZAU0158.532` and `GZAU0158.537`, file names with MJDs ranging from 58532 through to 58537 are generated. The generated file names are returned as a Python list.

File names in two formats can be interpolated:

1. names in the format `MJD.ext` where `MJD` is a five digit MJD and `ext`, the file extension, is arbitrary.
2. the format recommended in the CGGTTS v2E specification eg `GZAU0158.532`

For a sequence to be generated, the two file names must:

1. have the same path
2. have the same extension

## 6.9 ppsd

`ppsd` produces a digital pulse on an I/O port, aligned with the system time. It works by sleeping until just before the second (plus any programmed delay) rolls over, and then going into a hard loop, polling the time until rollover.

Two I/O ports are presently supported: the standard PC parallel port, and SIO8186x devices. The latter are found on some single board computers. For parallel port output, all 8 bits of the data port (pins 2 to 9 on a DB37) are written to.

`ppsd` doesn't produce a log file.

### 6.9.1 usage

To run `ppsd` on the command line, use:

```
ppsd [option] ...
```

The command line options are:

- d run in debugging mode
- h print help and exit

- o <delay> set the PPS delay, in microseconds.
- v print version information and exit

### 6.9.2 configuration file

The configuration file, `ppsd.conf` contains a single number, an offset for the output 1 pps, in microseconds.

## 6.10 sysmonitor.pl

`sysmonitor.pl` monitors the system status and provides notification of alarm conditions via files written to a specified directory, and via calling the alarm delivery system. In particular, `lcdmonitor` reads this directory to pick up current alarms.

Some of the conditions currently monitored include:

- TIC logging running (via it's status file)
- reference oscillator logging running
- reference is locked
- reference has lost power (PRS10 only)
- GPS logging running
- GPS receiver is tracking sufficient satellites
- RAID status (where RAID is used)
- NTP reference clocks are healthy

The run time for an alarm must integrate up to the configured threshold before an alarm is issued. Similarly, the run time for a clearing alarm must integrate to zero before a clear is issued.

### 6.10.1 usage

`sysmonitor.pl` is normally started by the init system, for example by `systemd` on Debian.

To run `sysmonitor.pl` on the command line, use:

```
sysmonitor.pl [OPTION]
```

The command line options are:

- c <file> use the specified configuration file
- d run in debugging mode
- h print help and exit
- v print version information and exit

To manually run `okcounterd`, you may need to disable the system service and kill any running `okcounterd` process.

### 6.10.2 configuration file

The configuration file uses the format described in 5.3.

#### **alarm path**

This defines the directory to which alarm notifications are written.

*Example:*

```
alarm path = /usr/local/log/alarms
```

#### **alarm threshold**

This defines the threshold at which alarms are raised. The units are seconds.

*Example:*

```
alarm threshold = 60
```

#### **alerter queue**

Alarms can be delivered by other methods using `alerter`. This entry defines the queue used by `alerter`. *Example:*

```
alerter queue = /usr/local/log/alert.log
```

#### **gpscv account**

This defines the account used for GPSCV processing (and implicitly, the location of `gpscv.conf`).

*Example:*

```
gpscv account = cvgps
```

#### **log file**

This defines the file used for logging of `sysmonitor`'s operation and alarm events.

*Example:*

```
log file = /usr/local/log/sysmonitor.log
```

#### **ntp account**

This defines the account used for NTP-related logging and processing.

*Example:*

```
ntp account = ntp-admin
```

### **ntpd refclocks**

This specifies a list of sections, each of which defines an `ntpd` refclock that is to be monitored.

*Example:*

```
ntpd refclocks = PPS,NMEA
```

An `ntpd` refclock section looks like:

```
[NMEA]
refid = 127.127.20.0
name = NMEA
```

### **6.10.3 log file**

`sysmonitor.pl` creates a log file. The default file is `/usr/local/log/sysmonitor.log`

## **6.11 gziplogs.pl**

`gziplogs.pl` is used to manage compression of log files. Typically, it will be run once per day, after UTC0. It requires a configuration file, `gziplogs.conf`, which is expected to be in the user's `etc` directory.

`gziplogs.pl` doesn't produce a log file.

### **6.11.1 usage**

`gziplogs.pl` is normally run as a `cron` job. To run it on the command line, use:

```
gziplogs.pl [OPTION] ...
```

The command line options are:

- c <file> use the specified configuration file
- d run in debugging mode
- h print help and exit
- m <MJD> compress files for the given MJD
- v print version information and exit



### 6.11.2 configuration file

The configuration file has a single entry.

#### **files**

This entry defines a comma-separated list of files to compress. Two date specifications, delimited by parentheses, are recognized: YYYYMMDD; and MJD.

*Example:*

```
files = raw/{MJD}.rx, raw/{MJD}.tic, raw/{YYYYMMDD}.dat
```

## A. Software license

The OpenTTP software is offered under the MIT license, as given below. A copy of this license is present in each software source.

The MIT License (MIT)

Copyright (c) 2018 The Authors

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.