Callisto-Lx: Setup and Operation Guide

Whitham D. Reeve

1. Introduction

Callisto-Lx is a Raspberry Pi computer that uses the Linux operating system and application software for automatic and manual control of the Callisto instrument. This guide provides step-by-step procedures for building and operating the Callisto-Lx. A general description of Callisto-Lx is at {CLXDesc}. This project involves both hardware and software. The computer hardware is inexpensive and all software is freely available through internet download. I previously described related applications of the Raspberry Pi including Callisto-Pi and GpsNtp-Pi (See {ReeveCPi} and {ReeveGNPi}, respectively).

2. Hardware

2.1 Hardware Overview

The hardware used in Callisto-Lx consists of three basic components: The Raspberry Pi; EIA-232 serial interface circuit compatible with the RPi CMOS logic levels and a power supply. A USB-serial converter also may work in place of the EIA-232 level converter, but I made no attempt to implement or test it. An optional hardware real-time clock (RTC) with battery backup also may be equipped and is recommended.

RPi platform: There are many versions of the Raspberry Pi {RPi}, and I tested the RPi 1 model B+, RPi 2 model B and RPi 3 model B. The RPi model 1 B also may work but I did not test it. In this document I refer to all models simply as *RPi*. I built two prototypes, one with the RTC and one without. The only difference is how the system clock is set as described below. The serial interface and RTC can be shop-built or commercial Raspberry Pi Hats may be used.

The RPi may be connected to a local area network either through a wired Ethernet interface or wireless WiFi interface. The RPi 3 model B has built-in WiFi capability but it will not work if the RPi3 is installed in a metal enclosure. The other models require a wireless USB dongle for WiFi access. The computing hardware system less enclosure is quite compact and costs less than 60 USD (figure 1).

Figure 1 ~ Callisto-Lx hardware includes the Raspberry Pi platform (bottom printed circuit board), a battery backed real-time clock PCB (middle) and a serial interface level converter PCB (top), stacked together to form a compact unit. The two peripheral circuit boards are mounted on 11 mm standoffs. Both boards have unneeded prototyping space. The USB and Ethernet interface connectors on the main board are at upper-right. A 3-pin right-angle header is used in place of the DB-9M connector (compare to next figure).

Serial interface: The Callisto-Lx serial interface uses signals available on the RPi general-purpose input/output (GPIO) connector. The GPIO is based on 3.3 V CMOS logic and is not directly compatible with the EIA-232 interface required by the Callisto instrument. Therefore, it is necessary to use a voltage level converter transceiver. The

interface described here is the *Serial Pi Plus* available at {Serial} (figure 2), which is a commercial Raspberry Pi Hat based on the MAX3232 transceiver integrated circuit. Cost is about 9 USD. One advantage of this particular product is the signals are brought out to both a DB-9M connector and separate header pads.

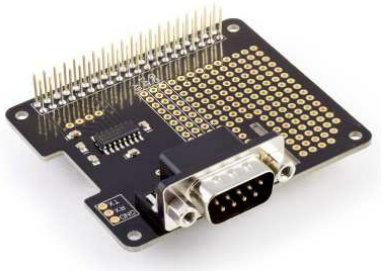


Figure 2 ~ *Serial Pi Plus* Hat used in Callisto-Lx includes a MAX3232 transceiver, a few charge pump and decoupling capacitors, a DB-9M connector and a 40-pin GPIO connector. The DB-9M connector is shown soldered to the PCB in this image but is removed to reduce space requirements when the RPi assembly is installed in an enclosure. Image source: AB Electronics.

Power Supply: In its basic configuration, the Callisto-Lx requires a 5 Vdc, 1 A (5 W) power supply but more current may be needed if the platform is used with a display. For example, a touchscreen display could be used that derives its power from the same bus as the RPi, in which case a 5 V power supply for the system should be rated at least 2 A (10 W). I installed Callisto-Lx in an enclosure. For better integration with the Callisto instrument, which uses nominal 12 Vdc for power, I used a 12 V to 5 V, 10 W step-down dc-dc converter and installed it in the enclosure with the RPi. This allows both the instrument and Callisto-Lx to be powered from the station 12 Vdc power supply.

Real-Time Clock: The Callisto instrument does not require internet (WAN) access to operate; however, data are time-stamped by the controlling computer and the basic RPi platform requires internet access to keep time. As originally supplied the RPi is not equipped with a battery backed hardware real-time clock (RTC). It keeps time by counting its on-board oscillator and implementing a software time-of-day clock (*system clock*). The RPi uses the Network Time Protocol (NTP) to set and regularly update the system clock.

By default, the RPi accesses NTP time servers on the worldwide web. It may be reconfigured to access a local NTP server such as the GpsNtp-Pi {ReeveGNPi} through a LAN connection. However, if the LAN and WAN connections both fail or are not available throughout the observing period, the time-of-day system clock cannot be set or updated. Therefore, it is recommended to use a hardware real-time clock and associated battery. Although the hardware clock can be manually set to the correct time, it does drift. The drift is compensated by NTP when the RPi has access to an NTP server. Commercial RTC Hats are available; for example, see the *RTC Pi Plus* available at {RTC} (figure 3), which uses the Maxim DS1307 RTC integrated circuit. Cost is about 9 USD + CR2032 battery. Although the DS1307 is capable of being fine-tuned to improve its time accuracy, the Linux drivers presently available do not support that feature.

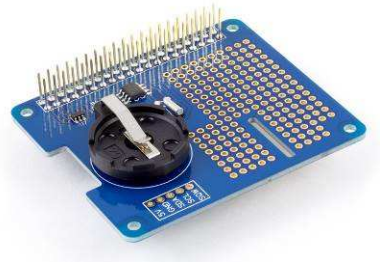


Figure 3 ~ *RTC Pi Plus* Hat used in Callisto-Lx includes a DS1307 integrated circuit, an I^2C bus interface IC, a few passive components, battery holder and a 40-pin GPIO connector. The battery is supplied by the user. Image source: AB Electronics.

Enclosure: I installed the RPi board with the two Hats in an extruded aluminum enclosure (figure 4) along with a small printed circuit board that contains the step-down dc-dc converter (figure 5). Power control and indication are installed on the rear panel and the EIA-232 interface connector is installed on the front panel (figure 6). The external appearance is identical to Callisto-Pi and GpsNtp-Pi except for the EIA-232 interface connector.

Figure 4 ~ Anodized extruded aluminum enclosure houses the RPi assembly and a power supply. The enclosure shown here with matching rubber bumpers on the end-plates is Box Enclosures BEX-series, and its dimensions are 108 W x 160 L x 50 H mm.

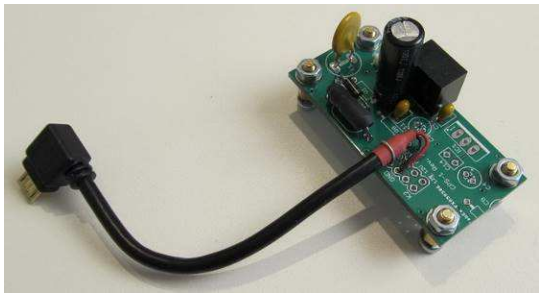


Figure 5 ~ Internal power supply uses the CPS-1-M1 printed circuit board (PCB). In the Callisto-Lx implementation, a 10 W step-down dc-dc converter is used along with input and output filter components, polarity guard diode and overcurrent and overvoltage protection. The input and output are connectorized so the power supply can be easily installed and removed. The cable seen in the image is terminated in a micro-USB connector, which is used by the RPi for power input. The Callisto-Pi and GpsNtp-Pi use the same power supply.

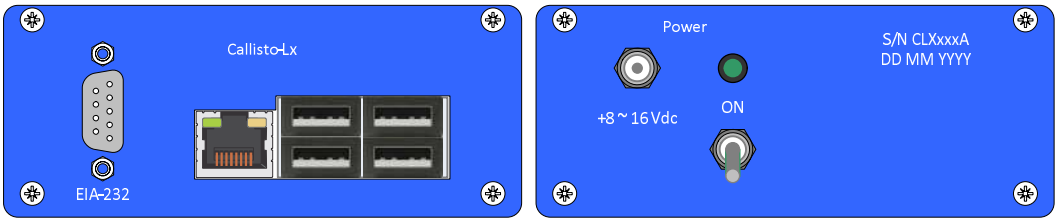


Figure 6 ~ Callisto-Lx front and back panels. Left: The front panel is cutout for flush-mounting the RPi USB and Ethernet interfaces and also includes a DB-9M connector for the serial interface, which is wired to the RPi through a wire-mount socket housing. Right: The power input jack, on-off switch and power indicating LED are on the back panel. The rear panel also is wired to the RPi through a wire-mount socket housing.

Display: The simplest way to operate Callisto-Lx is in the so-called “headless” mode in which no directly connected keyboard, mouse or display is needed. Alternately, an HDMI monitor or TV with an HDMI port may be connected to the RPi HDMI port and a keyboard and mouse may be connected through a wireless or wired connection to a USB port on the RPi. Another alternative uses a touchscreen, which provides a display and the equivalent of a keyboard and mouse all-in-one. I tested Callisto-Lx with a small touchscreen display {Display} but found the “headless” configuration far more convenient. Because my Callisto-Lx configurations use an enclosure, the RPi HDMI connector is not available for external connection anyway.

2.2 Hardware Installation

Two Hats are installed on an unmodified Raspberry Pi platform, one for the real-time clock and one for the serial interface level converter. The Hats can be obtained as kits that require a small amount of soldering. The Hat mounting hardware is purchased separately and consists of standoffs and fasteners (table 1).

Table 1 ~ Bill of material not including enclosure and associated power supply and components

Item	Qty	Description	Remarks
1	1	Real-time clock Hat for Raspberry Pi	Example, AB Electronics, RTC Pi Plus
2	1	Serial interface level converter Hat for Raspberry Pi	Example, AB Electronics, Serial Pi Plus
3	1	Battery, CR2032	
4	4	M2.5x6x6 mm M-F standoff, brass	
5	8	M2.5x6x11 mm M-F standoff, brass	
6	4	M2.5x6 mm machine screw, zinc plated steel	
7	4	M2.5 hex nut, zinc plated steel	
8	8	2.5 mm lock washer, zinc plated steel	
9	4	2.5 mm flat washer, zinc plated steel	
10	1	Header, 3-pin, right-angle, locking	Molex p/n 22-05-3031
11	1	Housing, wire receptacle, polarized locking, 3-pin	Molex p/n 22-01-3037
12	3	Terminal crimp contact, 22-30 AWG	Molex p/n 08-50-0114
13	1	Connector, DB-9M	
14	5	Wire, 24 AWG, 300 V hookup, 150 mm	Yellow, Brown, Green, Red, Black

If the two Hats are obtained as kits, solder the components according to the manufacturer's instructions. If the RPi is to be mounted in an enclosure, do not install the DB-9M connector; instead, use a 3-pin right-angle locking header wired to a panel mounted DB-9M connector (figure 7). Install a fresh battery in the RTC before mounting on the RPi. Install (4) 6 mm and (4) 11 mm standoffs on the RPi printed circuit board (PCB), with the 6 mm standoffs on the bottom. Place a lock washer on each 6 mm standoff post and insert in the holes in each corner of the PCB. Attach the 11 mm standoffs on the top of the PCB to these threaded posts and tighten. Line up the GPIO connectors and place the RTC PCB on the posts of the upper standoffs. Attach another set of (4) 11 mm standoffs to the top of the RTC PCB and tighten to secure the RTC. Line up the GPIO connectors and then place the serial PCB on the standoffs. Finally, place (4) flat washers, (4) lock washers and (4) hex nuts and tighten. The completed assembly may be installed in an extruded aluminum enclosure but is otherwise now ready for software installation.

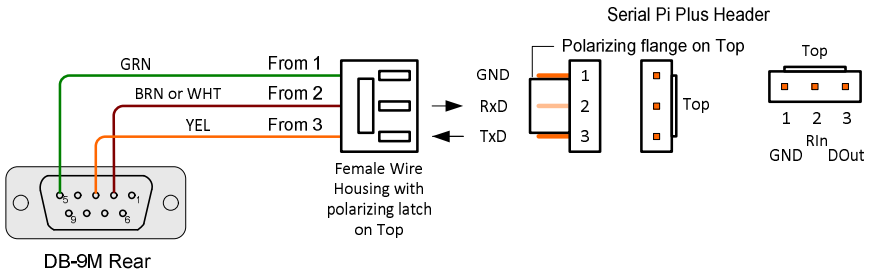


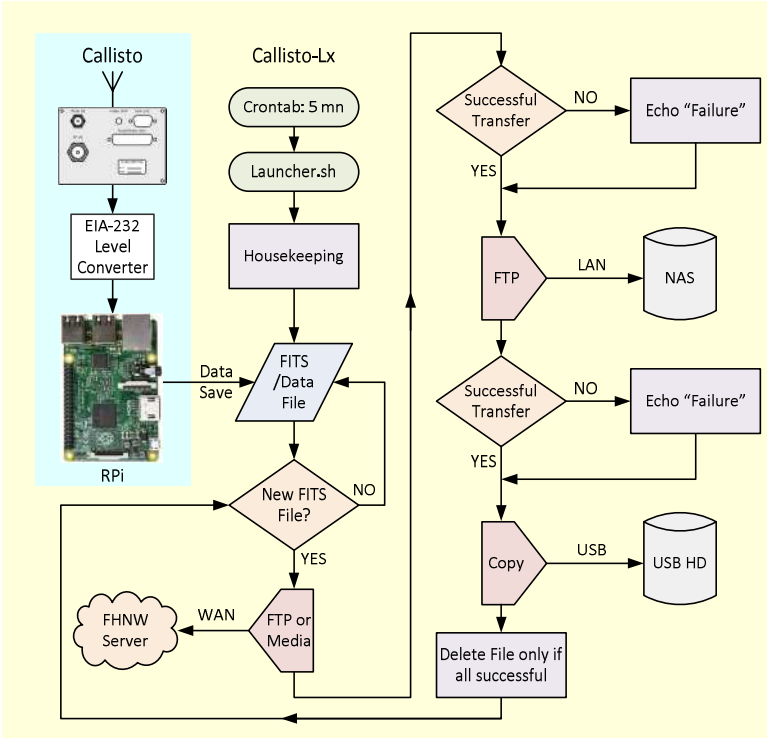
Figure 7 ~ Wiring diagram for serial interface based on the Serial Pi Plus Hat. A 3-pin right-angle header is used to connect a panel-mounted DB-9M connector instead of the on-board connector. The DB-9M is wired as EIA-232 DTE. This drawing applies to s/n CLX002 and later.

3. Software

3.1. Software Overview

The main software components in Callisto-Lx are the Linux operating system and the Callisto software application program. A number of software tools and applications also are installed to support testing and operating the hardware and software. As data files are produced by the Callisto software application, a launcher shell (figure 8) runs in the background to manage these files including sending them to the e-Callisto data server and archiving to a Network Attached Storage (NAS) server and a USB hard drive or USB flash drive.

Figure 8 ~ Launcher shell flow chart. After a data file is produced by the Callisto application software, the launcher shell manages its disposition, which includes copying to FTP servers and to an optional local USB drive.



Generally, all software installations are configured from a Windows PC using PuTTY or Tera Term terminal emulator as the primary command line interface. The command line interface provides a secure shell (SSH) connection to the RPi through the LAN. The PC also uses WinSCP as the RPi file explorer (WinSCP stands for Windows Secure Copy). The advantage of WinSCP is that directory structure is shown in a more obvious way than is possible with a command line interface. Also, it allows files to be conveniently dragged and dropped between the Windows PC and the RPi.

Operating system: The latest RPi operating system at the time of writing is the Raspbian *Jessie* implementation of Debian Linux {Raspbian}. I installed the *Jessie Lite* version because it uses much less space than the full version on the memory card with no loss of features relevant to Callisto-Lx. The operating system software is easily downloaded and installed on the micro-SD memory card used in the RPi. Before the application software and tools are installed, the RPi operating system is brought up-to-date. Depending on the amount of updating required, this may require several hours (one of my systems ran overnight to upgrade).

The serial interface needs to be configured in software. If installed, the hardware real-time clock uses the RPi's built-in I^2C bus and requires installation of additional software drivers and configuration for operation. A separate terminal emulator program called *Minicom* is installed on the RPi for testing the serial interface connection with the Callisto instrument. After the interfaces are setup and tested, the application software is installed and configured. Software installation and configuration requires at most a couple hours and involves following a step-by-step guide, which includes tests at each major step {CLXGuide}.

Callisto application: The application software is provided as *.tar.gz file packages that are first extracted and then compiled on the RPi to build the executable binary files. The result is a set of directories and the main callisto program. The program requires the function library for input/output of FITS formatted data files, which is installed using the Advanced Packaging Tool (APT), a normal Linux installation process. I found the installation to be relatively easy and trouble-free, requiring only a few minutes. The callisto application is installed as a *service*, which can be started, stopped and restarted as needed.

3.2. Software Operating System Installation and Setup

3.2.1. Basic setup: Install the Raspbian *Jessie Lite* operating system on a memory card and complete the steps described in {RPiSetup}. Change the RPi default `hosts` and `hostname` to *Callisto-Lx* and be sure to change the password. If the RPi is to be used in a wireless LAN environment, it can be setup according to {RPiWLAN}.

3.2.2 FTP: It is necessary to install and configure the File Transfer Protocol (FTP) so that Callisto FITS files can be uploaded to the FHNW server and also to move the FITS files to a Network Access Storage (NAS) server or local drive for archiving. Note that a NAS can be easily accessed from both the Windows and Linux PCs. The code necessary to use FTP is placed in a launcher shell file (launcher.sh) and it is executed every 5 minutes under control of the Cron process in Linux; both are described in the following subsections.

Install FTP (no configuration is necessary)

```
sudo apt-get install ftp
```

3.2.3. Configure optional USB drive: A USB drive, such as a flash ("thumb") drive or hard drive, may be used for long term data storage.

```
sudo apt-get install usbmount
sudo apt-get install ntfs-3g
sudo mkdir /media/Callisto-Lx
```

Reboot the RPi.

```
sudo reboot
```

Login using PuTTY and change permissions for the new directory

```
cd /media/
sudo chmod a+rwx Callisto-Lx
```

Commands that may be useful if problems are experienced with the USB drive:

```
sudo umount /dev/sda1
sudo mount –a
sudo fdisk -l
ls -l /media/Calliisto-Lx/
sudo chown pi:pi /media/Callisto-Lx
```

3.2.4. Prepare the launcher shell script: The launcher shell script manages all data file transfers, first by determining if a new FITS file has been placed in the Data directory and then copying it to various destinations including the FHNW server in Switzerland, an optional local NAS or USB drive and finally to an Archive directory. After these actions successfully complete, the launcher.sh deletes the file from the Data directory. If an FTP transfer is not successful, the file remains in the Data directory for attempted transfer in the next data cycle.

At the PuTTY command line prompt enter

```
sudo nano launcher.sh
```

When the nano editor opens, copy the text from **Appendix I** including indents to the editor. Be sure to change the usernames, passwords and paths to correspond with actual requirements. If data files are to be copied to a

NAS server, be sure a directory is setup on the NAS to receive the files. When finished, save the file and exit the editor (CTRL-X, Y, Enter).

Another way to prepare the file is to copy/paste the text from the Appendix to a new file in NotePad++. Be sure the indentation is correct and passwords, usernames and paths correspond to actual requirements. Save the file as `launcher.sh` (not launcher.txt) on the Windows Desktop. Now, using WinSCP drag the file from the Desktop to the `/home/pi/` directory in Callisto-Lx. The launcher.sh file may be viewed later by double clicking it in WinSCP or running the nano editor again, as when it was built (figure 9).

Figure 9 ~ Launcher shell in the nano editor (only part of the file is visible). Note that nano color codes the text to aid in reading. The nano editor has limited features but is entirely adequate for the purposes here.

After placing the launcher.sh file in the proper directory, change its Execution permission

```
chmod +x launcher.sh
```

3.2.5. Setup the cron table: The cron table is very simple in that it simply runs the launcher.sh every 5 minutes. Edit crontab using its built-in editor

```
crontab -e
```

At the bottom of the file add the lines immediately after the comment that shows the format, as in

```
# m h  dom mon dow   command
SHELL=/bin/bash
*/5 *  *   *   *     /home/pi/launcher.sh
```

3.2.6. Install Postfix mail transfer agent (MTA) and associated mail handler MailUtils: These progams aid in troubleshooting. Postfix places a message file in `/root/var/mail` labeled `pi` if a program such as FTP, cron or launcher.sh encounter problems. Mailutils allows the mail to be read.

```
sudo apt-get install postfix
```

When PostFix is installed, a configuration window will open. Select `Local Only` in the configuration dialog and use `Callisto-Lx` as hostname. If necessary, the PostFix configuration can be changed later by entering

```
sudo nano /etc/postfix/main.cf
```

Install the MailUtils mail handler (no configuration is needed)

```
sudo apt-get install mailutils
```

3.2.7. Serial interface setup: It is necessary to modify some of the system configuration files to make the serial port compatible with Callisto-Lx. All of the changes indicated below apply to the RPi3 but only the

enable_uart needs to be changed in the RPi2. However, all modifications are compatible with and may be installed on the RPi2.

At the PuTTY command line interface, disable BlueTooth and remap the UART to ttyAMA0 as follows:

```
sudo nano /boot/config.txt
```

Add two lines at the bottom (if the **enable_uart** line already exists, change its value if necessary)

```
enable_uart=1
dtoverlay=pi3-disable-bt
```

Save and exit the editor (CTRL-X, Y, Enter). Reboot the RPi.

```
sudo reboot
```

List the devices

```
ls -l /dev
```

Scroll through the devices to confirm that serial0 is mapped to ttyAMA0, as in **serial0 -> ttyAMA0**. Also, there should be a separate, stand-alone entry for **ttyAMA0**.

Disable internal console access to the serial port

```
sudo systemctl stop serial-getty@ttyS0.service
sudo systemctl disable serial-getty@ttyS0.service
```

Edit the internal command line configuration

```
sudo nano /boot/cmdline.txt
```

Remove the line:

```
console=serial0,115200
```

Save the changes and exit the editor (CTRL-X, Y, Enter).

For both the RPi2 and RPi3, setup a Callisto-Lx directory (this is required for testing the serial interface and also is used for the data subdirectories, which are setup later)

```
mkdir /home/pi/Callisto-Lx
```

Reboot the RPi.

```
sudo reboot
```

The RPi serial interface is now ready for testing. To test the interface it is necessary to install the Minicom terminal emulator program as described at {Minicom}. When finished, return to this document to install the real-time clock and then the Callisto Linux software application described below.

3.2.8. Real-time clock: The real-time clock software installation discussed here applies to the Maxim (Dallas Semiconductor) DS1307 RTC integrated circuit and presumes the *RTC Plus* Hat mentioned previously is used. It also presumes the RPi operating system already has been updated to the latest distribution. The updates should have been installed during Basic Setup. The real-time clock requires the I^2C bus and instructions for it are broken

into three parts. The first part applies to both the Raspbian Jessie and Wheezy operating systems; the second part applies only to Jessie and the third part applies only to Wheezy.

3.2.9. Perform the following procedures for both Raspbian Wheezy and Jessie operating systems:

3.2.9.1. Enable the I^2C bus

```
sudo raspi-config
```

Select **8 Advanced Options** and then **A7 I2C - Enable/Disable automatic loading**. A prompt will appear: **Would you like the ARM I2C interface to be enabled?** Select Yes. Exit and reboot the RPi as instructed by raspi-config.

3.2.9.2. Install Python support (Note: this step may not be required but it allows for future changes)

```
sudo apt-get install python-smbus python3-smbus python-dev python3-dev
```

3.2.9.3. Install I^2C Tools

```
sudo apt-get install i2c-tools
```

3.2.9.4. Enable I^2C and SPI protocols

```
sudo nano /etc/modprobe.d/raspi-blacklist.conf
```

If the file does not contain the following two lines or they are commented out, exit the editor (no changes are needed)

```
blacklist spi-bcm2708
blacklist i2c-bcm2708
```

If the two lines exist, comment them out with the hash character #

```
# blacklist spi-bcm2708
# blacklist i2c-bcm2708
```

Save the changes and exit the editor (CTRL-X, Y, Enter).

3.2.9.5. Update the /boot/config.txt file

```
sudo nano /boot/config.txt
```

If the two lines below already exist, exit the editor (no changes are needed); otherwise, add the two lines to the end of the file

```
dtparam=i2c1=on
dtparam=i2c_arm=on
```

Save the changes and exit the editor (CTRL-X, Y, Enter).

3.2.9.6. Set the I^2C bus device driver to start automatically at boot

```
sudo nano /etc/modules
```

Add a new line at the bottom of the file

i2c-dev

Save the changes and exit the editor (CTRL-X, Y, Enter).

3.2.9.7. To avoid having to run the I²C tools as root add the 'pi' user to the I²C group

sudo adduser pi i2c

3.2.9.8. Reboot the RPi

sudo reboot

3.2.9.9. Check that the real-time clock has been detected.

For the RPi 1 model B+ enter

sudo i2cdetect -y 0

For the RPi 2 model B and RPi 3 model B enter

sudo i2cdetect -y 1

An entry should appear on channel 68, indicating the real-time clock has been detected on the bus (figure 10). If it does not appear, check for correct battery installation or a dead battery. Note: If "UU" is shown at channel 68, the driver already is installed and detected on the channel.

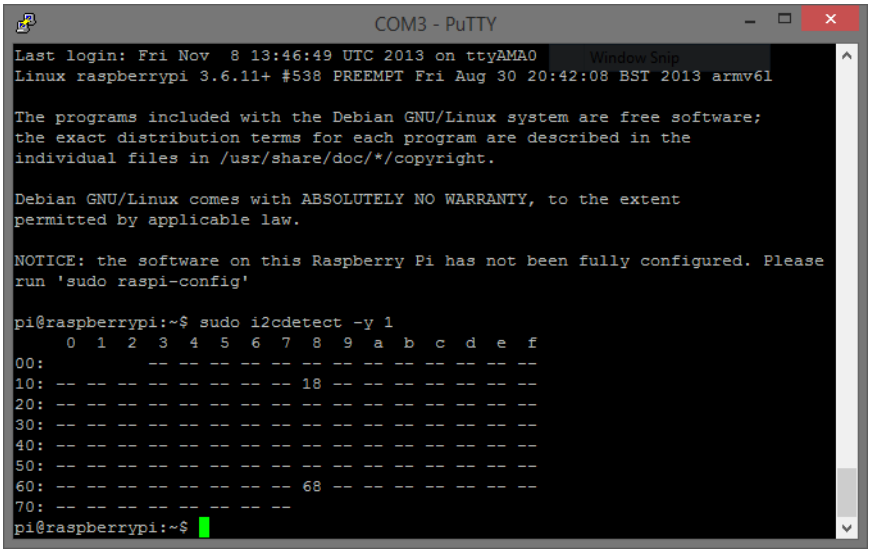


Figure 10 ~ Detection of the real-time clock on the I²C bus is indicated by an entry is channel 68 of the channel matrix.

3.2.10. <u>Perform the following steps only on the Raspbian Jessie operating system (skip to the next subsection for Wheezy)</u>:

3.2.10.1. Add the DS1307 real-time clock to the RPi configuration

sudo nano /boot/config.txt

At the end of the file add the line

dtoverlay=i2c-rtc,ds1307

When finished save the changes and exit the editor (CTRL-X, Y, Enter).

3.2.10.2. Add the real-time clock module

```
sudo nano /etc/modules
```

At the end of the file add the line
```
rtc-ds1307
```

When finished save the changes and exit the editor (CTRL-X, Y, Enter).

3.2.10.3. Edit hardware clock device manager
```
sudo nano /lib/udev/hwclock-set
```

Locate the following lines
```
if [ -e /run/systemd/system ] ; then
exit 0
fi
```

Comment out the three lines with the hash character #. The changes should look like
```
# if [ -e /run/systemd/system ] ; then
# exit 0
# fi
```

When finished save the changes and exit the editor (CTRL-X, Y, Enter). Reboot the Raspberry Pi.
```
sudo reboot
```

3.2.10.4. If this is the first time the RTC has been run it will display a date of January 1, 2000. The time will depend on how long the RTC has been running since powered up. Check the date and time by entering
```
sudo hwclock -r
```

Check the current date and time in the software system clock with the command
```
date
```

The hardware clock date and time can be set manually or by using the software system clock; the latter should be correct if the RPi is connected to the internet. To set the software system date and time manually enter
```
date -s "DD MMM YYYY HH:MM:SS"
```

To save the software system clock time and date onto the hardware clock enter
```
sudo hwclock -w
```

Verify the date has been saved in the hardware clock by entering
```
sudo hwclock -r
```

If everything worked correctly the RTC will be initialized on boot and the current date and time will be loaded into the software clock. Go to the next section.

3.2.11. Perform the following steps only on the Raspbian Wheezy operating system:

3.2.11.1. Enable the I^2C bus by editing the blacklist

```
sudo nano /etc/modprobe.d/raspi-blacklist.conf
```

Find the line

```
blacklist i2c-bcm2708
```

If the line does not exist, exit the editor (no changes are required). If the line does exist, comment it out with the hash character #. The changes should look like

```
# blacklist i2c-bcm2708
```

When finished save the changes and exit the editor (CTRL-X, Y, Enter).

3.2.11.2. Invoke the bash shell as root

```
sudo bash
```

For the RPi 1 model B+ enter

```
echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-0/new_device
```

For the RPi 2 model B and RPi 3 model B enter

```
echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
```

Exit the bash shell as root

```
exit
```

3.2.11.3. If this is the first time the RTC has been run it will display a date of January 1, 2000. The time will depend on how long the RTC has been running since powered up. Check the current time and date on the hardware real-time clock

```
sudo hwclock -r
```

3.2.11.4. The date and time can be set manually or by using the Linux software clock, which should be correct if the RPi is connected to the internet. To set the date and time manually enter

```
date -s "DD MMM YYYY HH:MM:SS"
```

To set the date and time using the software clock, first check to make sure it is correct by entering

```
date
```

Save the date onto the real-time clock

```
sudo hwclock -w
```

Verify the date has been saved onto the real-time clock

```
sudo hwclock -r
```

3.2.11.5. Edit the modules to load the real-time clock when the RPi boots.

```
sudo nano /etc/modules
```

Add at the end of the file

```
rtc-ds1307
```

When finished save the changes and exit the editor (CTRL-X, Y, Enter).

3.2.11.6. Edit the real-time hardware clock services so that it automatically sets the system software clock

```
sudo nano /etc/rc.local
```

Add the following at the bottom of the file above the line **`exit 0`**:

For the RPi 1 model B+ enter

```
echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-0/new_device
sudo hwclock -s
```

For the RPi 2 model B and RPi 3 model B enter

```
echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
sudo hwclock -s
```

When finished save the changes and exit the editor (CTRL-X, Y, Enter). Reboot the RPi

```
sudo reboot
```

If the above changes provide the correct responses, the RTC will be initialized on boot and the current date and time will be loaded into the software clock. Go to the next section.

3.3. Callisto Linux Application Software Installation

Download the files associated witih the application software.

```
sudo apt-get install libcfitsio3 libcfitsio3-dev
wget http://www.metsahovi.fi/callisto/e-Callisto_for_Unix/callisto-1.1.0.tar.gz
wget http://www.metsahovi.fi/callisto/e-Callisto_for_Unix/Debian.init.d
wget http://www.metsahovi.fi/callisto/e-Callisto_for_Unix/Debian.cron.daily
tar xvf callisto-1.1.0.tar.gz
```

Install the primary callisto application

```
cd callisto-1.1.0/
./configure --prefix=/usr --sysconfdir=/etc
make
sudo make install
```

Prepare a directory for the configuration files.

```
cd ..
sudo mkdir /etc/callisto
```

Using a text editor in Windows (for example, NotePad or NotePad++) or the RPi nano editor, prepare the Callisto configuration files callisto.cfg, frqxxxxx.cfg and scheduler.cfg. See the next section for information and limitations for these files. The files from an existing Windows installation may be used, but they may need editing to conform to the requirements of the Callisto-Lx.

If the files are prepared on a Windows PC, use WinSCP to drag/drop the configuration files to Callisto-Lx directory `/home/pi/Callisto-Lx/`. Now, use the PuTTY command line to copy the files to `/etc/callisto/`. After copying, change permissions so the files can be edited and then reboot.

```
sudo cp /home/pi/Callisto-Lx/*.cfg /etc/callisto/
cd /etc/callisto/
ls
sudo chmod a+rwx *.cfg

sudo reboot
```

Copy the Callisto `Debian.init.d` to the proper directory

```
sudo cp Debian.init.d /etc/init.d/Callisto
```

Copy the `Debian.cron.daily` to the proper directory

```
sudo cp Debian.cron.daily /etc/cron.daily/Callisto
```

Make the output directory structure the same as in the callisto.cfg configuration file. Alternately, use WinSCP to make the directory structure (Note that the Linux version of callisto does not use the /LC or /Log directories but they must be setup).

```
mkdir /home/pi/Callisto-Lx/Data
mkdir /home/pi/Callisto-Lx/LC
mkdir /home/pi/Callisto-Lx/Log
mkdir /home/pi/Callisto-Lx/Ovs
```

Modify permissions for init.d and reload

```
cd /etc/init.d
sudo chmod +x Callisto

sudo systemctl daemon-reload
```

Be sure the Callisto instrument is turned on and its serial port is connected to the RPi serial interface. Now, start the callisto service

```
sudo service callisto start
```

The console will appear to pause of a few seconds as the service is started. If service start fails, view the error by looking at:

```
sudo journalctl –xn
```

Also, look in `/root/var/daemon.log` for a log of activities including failures.

If there are no errors, check that Callisto service is active by querying system control:

```
systemctl is-active callisto.service
```

Example, if active:

```
pi@Callisto-Lx:~ $ systemctl is-active callisto.service
active
```

Example, if inactive:

```
pi@Callisto-Lx:~ $ systemctl is-active callisto.service
unknown
```

3.4. Basic Operation of Callisto Linux Application Software

The callisto service is started and stopped by running:

```
sudo service callisto start
sudo service callisto stop
```

The Callisto file in `/etc/init.d` can be used to invoke options. For example, to load the frequency configuration file at every boot, add the –L option to `init.d`. In this example, open the file in nano, as in

```
sudo nano /etc/init.d/Callisto
```

Look for the following lines and edit to include the option –L as shown

```
# Default options, these can be overriden by the information
# at /etc/default/$NAME
DAEMON_OPTS="-L"          # Additional options given to the server
```

Note to minimize wear and tear on the Callisto instrument MPU flash memory it is not recommended to load the frequency configuration file at every boot, so the above option normally is not used.

The callisto software also has debug capability as described in the callisto manual. This is handy for troubleshooting. For example, to temporarily turn on debug mode and save the logs in the /tmp/ directory, enter

```
callisto -dDLC 2>/tmp/upload.log
```

or

```
callisto -dD 2>/tmp/startup.log
```

The first command above attempts to load the default frequency configuration file into the instrument EEPROM and then check the channels; it saves the log as plain text in *upload.log*. This command will exit on its own. The second command attempts to start the callisto software and saves the log in *startup.log*. The second command will exit on its own if a startup failure occurs but if there is no failure, it will not exit, in which case the RPi must be rebooted to regain control using, for example, a duplicate PuTTY session.

3.5. Additional commands for reference, unrelated to the callisto software:

Use scrot to produce a screenshot image of the active window with date-time stamp and move it to the `/home/pi/images/` directory after 10 s delay:

```
scrot '%Y-%m-%d-%T_$wx$h.png' –u –d 10 –e 'mv $f ~/images/'
```

Assuming minicom has been setup as previously specified, it may be used to control Callisto and capture files to /Callisto-Lx/ directory. For example

```
minicom Callisto-Lx –C ~/Callisto-Lx/Callistotest.txt
```

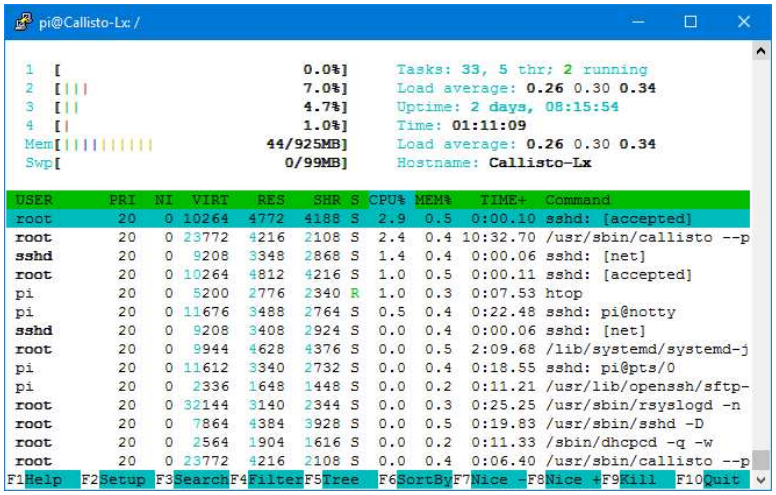
To check RPi CPU, memory usage and other performance parameters, install the process viewer `htop`

```
sudo apt-get install htop
```

After installation run (figure 11)

```
htop
```

Figure 11 ~ Resource viewer htop is similar to the Windows Task Manager.



3.5. Completion:

After all changes are made, shutdown the RPi.

```
sudo halt
```

When shutdown is complete (Activity LED on RPi stops flashing, about 10 to 15 s), remove power from the RPi and remove the memory card. Backup the memory card image as described in {RPiBckRst} and then reinstall in the RPi.

4. Callisto Configuration Files

Calllisto-Lx uses configuration files that are similar to the Windows version: callisto.cfg, freqxxxxx.cfg and scheduler.cfg. For the most part, the rules described in the Callisto Software Setup Guide ({CallistoSSG} apply to the Linux versions, but the Linux versions have some limitations, which are briefly described below and in more detail in the callisto manual embedded in the software.

An example callisto.cfg file is shown below. A simple way to edit the configuration files after they have been placed on the RPi is to navigate to the file in WinSCP and then double-click it. The file will open in the WinSCP editor. When finished editing, click the Save icon in the upper-left corner of the editor window.

The callisto.cfg file parameter entry for `[rxcomport]` is not in the familiar Windows COMx format but is of the form `/dev/ttyxx` (in the implementation described here the actual port is `/dev/ttyAMA0`).

Other important differences between the callisto.cfg file in Windows and Linux involve the `[autostart]` and `[net_port]` parameters. If `[autostart] = 1`, recording to a FITS file is automatically started when the callisto service is started. If `[autostart] =0`, recording is not automatically started (equivalent to manual mode in Windows). If not defined or set to a negative value, recording follows the entries in scheduler.cfg, or set to 1 if there is no schedule. The default is `[autostart]` = -1 (follow scheduler.cfg).

The `[net_port]` parameter is optional and does not have to exist (unlike all other parameters). If it does exist, the callisto service starts a command server on the RPi (that is, starting the service also starts the command server if the parameter exists). This server then may be used to manually control the Callisto instrument such as start and stop data recording or record a spectral overview (OVS). The use of the command server is described in more detail in the next section.

Important: The scheduler.cfg file used in the Linux version does not support variable focus codes or variable frequency files (as does the Windows version). The focus codes used in the schedulere.cfg file must be the same as in the callisto.cfg file`[focuscode]`parameter. For example, if `[focuscode]`=58 in callisto.cfg then 58 must be used in each entry in scheduler.cfg. The callisto service checks the scheduler.cfg file every minute so the scheduler can be changed "on-the-fly".

```
[rxcomport]=/dev/ttyAMA0   /* usually of the form /dev/tty...                    */
[observatory]=12           /* CALLISTO=12, fixed                                 */
[instrument]=ALASKA-Pi     /* instrument code -> filename_                       */
[titlecomment]=NA008       /* Title of API                                       */
[origin]=Anchorage_AK_USA  /* Place of instrument                                */
[longitude]=W,149.9565     /* default geographical longitude in decimal degree   */
[latitude]=N,61.1993       /* default geographical latitude in decimal degree    */
[height]=20.0              /* default altitude [m] above sealevel                */
[clocksource]=1            /* 0=software, 1=internal, 2=external, default 1      */
[filetime]=900             /* time period for one single FIT-file (in seconds)   */
[frqfile]=frq00215.cfg     /* default frequency program                          */
[focuscode]=59             /* default focuscode                                  */
[mmode]=3                  /* default 3=raw data (only one supported)            */
[fitsenable]=1                        /* 0=no FITSfile, 1=FITS write On          */
[datapath]=/home/pi/Callisto-Lx/Data/  /* default datafile path (*.fit)           */
[logpath]=/home/pi/Callisto-Lx/Log/    /* default logfile path (LOG*.txt)         */
[lcpath]=/home/pi/Callisto-Lx/LC/      /* default light curve path (LC*.txt)      */
[ovspath]=/home/pi/Callisto-Lx/Ovs/    /* default spectral overview path (OVS*.prn)*/
[chargepump]=1             /* charge pump: 0=false=off, 1=true=on, default 1     */
[agclevel]=150             /* PWM level for tuner AGC 50...255, default 120      */
[detector_sens]=25.4       /* detector sensitivity mV/dB, default 25.4           */
[db_scale]=5               /* dB per division in XY-plot, default 6              */
[autostart]=-1             /* autostart: 0=false, 1=true, -1=use scheduler.cfg   */
[net_port]=6789            /* Network port number for command server             */
```

The Linux version of the frequency configuration file supports 400 frequency channels (same as the Windows version).

For additional information on the Linux configuration files, use the Callisto manual that accompanies the Linux version. At the command prompt enter

```
man callisto
```

5. Daily Operation

After setup has been completed, the Callisto system is ready for operation. For a new setup, it is first necessary to manually load the default frequency configuration file specified in the Callisto configuration file and then start the system. At the command line prompt enter

```
sudo callisto –LC
sudo service callisto start
```

The first entry may require a few moments depending on the size of the frequency configuration file and the second entry will require a few seconds as the service is started.

The –LC switch used above loads the frequency configuration file (frqxxxxx.cfg) specified in the callisto configuration file (callisto.cfg) into EEPROM in the Callisto instrument and then checks (verifies) that it is

properly loaded. For any given frequency file, the –LC switch only needs to be used once. If the frequency configuration is changed later, stop the system

```
sudo service callisto stop
```

Update the frequency configuration file and, if necessary, change the filename in callisto.cfg. Then, enter

```
sudo callisto –LC
sudo service callisto start
```

When the callisto service is started, and `autostart` =-1 in the callisto.cfg file, callisto reads the scheduler.cfg file. Observations will automatically start at the next available time indicated in the scheduler file. For example, assume the scheduler file contains the listing below. If the callisto service is started at 17:16, actual observations will not start until the next listed time of 19:00.

```
02:00:00,59,3  // restart
04:00:00,59,0  // STOP
15:00:00,59,8  // Spectral overview
17:00:00,59,3  // START
19:00:00,59,3  // restart
21:00:00,59,3  // restart
23:00:00,59,3  // restart
```

If the scheduler file contains improper format or syntax errors, it cannot be read by the callisto software. No indication will be provided except that data collection is stopped. If a problem is suspected, query the software status at the command line interface prompt, as in

```
sudo service callisto status
```

If the scheduler.cfg file is the problem, the response will be similar to

```
Apr 18 02:48:57 Callisto-Lx2 callisto[1528]: Malformed schedule file entry i...0
```

If it is desired to start collecting data immediately and then revert to the scheduler, the command server may be used. In this case, open a New Session in PuTTY as explained in the next sub-section and issue a `start` command. The callisto software will start collecting data at that time and then revert to the scheduler at 19:00.

If the RPi is rebooted for any reason, it is necessary to manually start the callisto service. The callisto software system is not setup to start on its own when the RPi reboots. At the PuTTY command line prompt enter

```
sudo service callisto start
```

The callisto service may be restarted if it already is running. This may be required if there is a fault that affects the service. At the PuTTY command line prompt enter

```
sudo service callisto restart
```

The status of the callisto service may be checked any time by entering

```
sudo service callisto status
```

If the service is running, the status response will be similar to below. Note that in this case there is a list of actions at the end that indicate the system has been recording according to a scheduler.cfg.

```
pi@Callisto-Lx:~ $ sudo service callisto status
● Callisto.service - LSB: e-Callisto
   Loaded: loaded (/etc/init.d/Callisto)
   Active: active (running) since Thu 2017-03-02 19:00:44 UTC; 8h ago
  Process: 3959 ExecStart=/etc/init.d/Callisto start (code=exited, status=0/SUCCESS)
   CGroup: /system.slice/Callisto.service
           └─3965 /usr/sbin/callisto --pidfile /var/run/callisto.pid

Mar 02 19:00:44 Callisto-Lx Callisto[3959]: Starting e-Callisto : callisto.
Mar 02 19:00:44 Callisto-Lx callisto[3965]: e-Callisto for Unix 1.1.0 started
Mar 02 19:00:44 Callisto-Lx systemd[1]: Started LSB: e-Callisto.
Mar 02 20:00:00 Callisto-Lx callisto[3965]: Recording (re)started by schedule
Mar 02 21:00:00 Callisto-Lx callisto[3965]: Recording (re)started by schedule
Mar 02 22:00:00 Callisto-Lx callisto[3965]: Recording stopped by schedule
pi@Callisto-Lx:~ $
```

If the service has been stopped, for example as the result of reboot, it show as `inactive (dead)` similar to below.

```
pi@Callisto-Lx:~ $ sudo service callisto status
● Callisto.service - LSB: e-Callisto
   Loaded: loaded (/etc/init.d/Callisto)
   Active: inactive (dead)
pi@Callisto-Lx:~ $
```

As explained in the previous section, when the callisto service is started, and if [`autostart`] =`0`, no further action is taken. If [`autostart`] = `-1`, the callisto service will use the scheduler.cfg file specified in the callisto.cfg file to determine observation start and stop times and save the data in the location and interval specified. If [`autostart`] =`1`, the callisto service will start the instrument and it will record data until manually stopped by stopping the callisto service or by using the command server described below.

Command server: The callisto software has a built-in command server that always listens on a predetermined TCP port and can be used to control the Callisto instrument. Thus, the instrument operation can be controlled by a schedule file (scheduler.cfg) and also directly via the command server. Typical uses of the command server are to manually start and stop data recording and collect a spectral overview (see callisto manual for additional commands)

`start`	(start recording data irrespective of scheduler.cfg)
`stop`	(stop recording data irrespective of scheduler.cfg)
`overview`	(record spectral overview)

The instrument commands passed through the command server are not the same as the software service commands previously described. To use the command server, it is necessary that the callisto service is running (via `sudo service callisto start` at the PuTTY command line interface). It also is necessary to add the [net_port] parameter at the end of the callisto.cfg file. The [net_port] parameter does not exist in the Windows version of the Callisto software. The example callisto.cfg file shown in the previous section has the [net_port] parameter set to 6789 (this value is only an example; the range is 0 to 65535 for IPv4 but some port numbers are preassigned for other services such as SSH, Telnet and FTP).

After the [net_port] parameter has been set in callisto.cfg and the file saved, PuTTY can be used to control the instrument through the command server. Open PuTTY for a new session and in the Configuration window (figure

12) enter the Callisto-Lx IP address (10.0.0.25 in the example), desired port (6789 in the example) and click the radio button for Connection type Raw. Click Open.

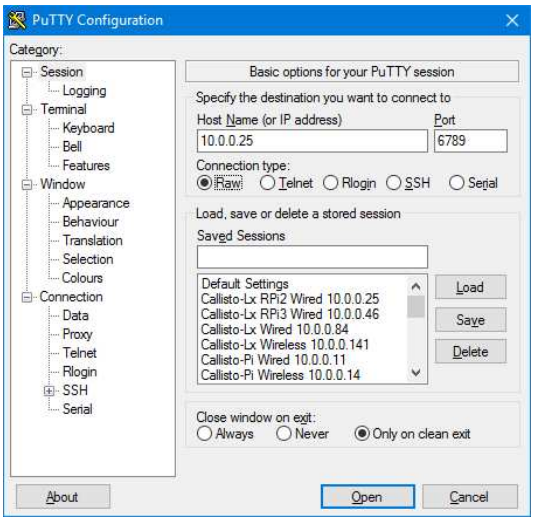


Figure 12 ~ Setting up PuTTY to access the callisto command server on example port 6789.

Successful connection to the command server is indicated by the prompt **`e-Callisto for Unix 1.1.0`**. At this point commands can be entered (figure 13). In the example screenshots below, data recording has been started and then stopped. Each command is acknowledged. All supported commands are described in the callisto manual. To close the connection to the command server, enter **`quit`** followed by Enter.

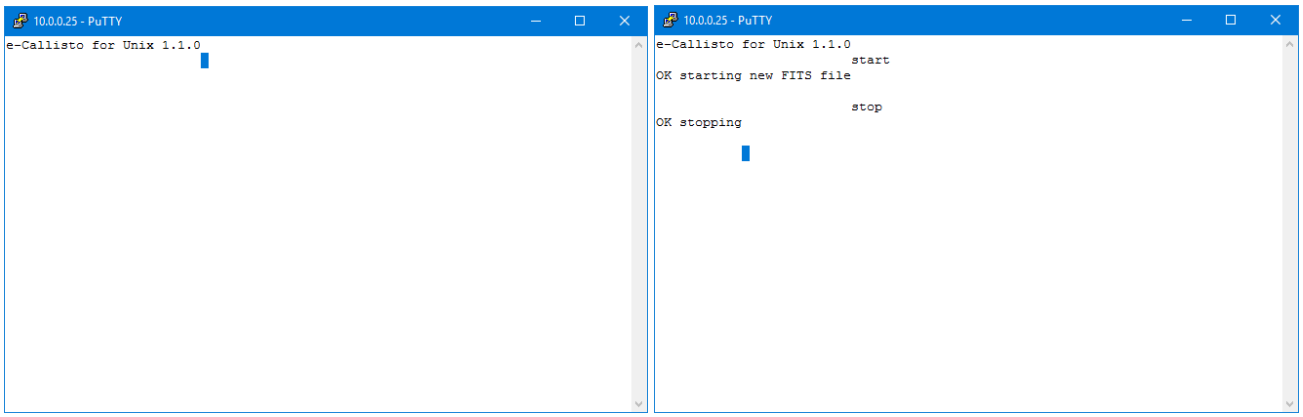


Figure 13 ~ Screenshots of PuTTY accessing the callisto command server. Left: Prompt on connection to the command server. Right: Examples of starting and stopping the Callisto.

If the network connection is refused (figure 14), check that the `bindv6only` file is set to 0. This file is located in **`/proc/sys/net/ipv6/bindv6only`** and its value may be determined by entering at a PuTTY command line prompt

`sysctl net.ipv6.bindv6only`

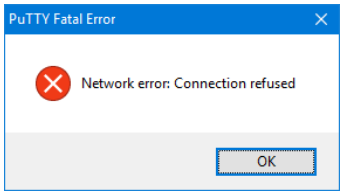


Figure 14 ~ Error window indicates that the network connection to the callisto command server has been refused.

Logging: The Linux version of callisto software logs program operation using the daemon facility of syslog (the Windows version has a dedicated folder for system logs). In the RPi, the path to the daemon log is `/var/log/daemon.log`. The simplest way to access this log is to double-click it in WinSCP. The log is useful for troubleshooting.

Safely remove power: Never remove or turn off power to the RPi without first shutting down the software system. If the RPi needs to be turned off, enter at the PuTTY command line prompt

`sudo halt` (or `sudo shutdown -h now`)

Wait for the Activity LED on the RPi to stop flashing (about 10 to 15 s) before removing power. Remember that when power is reapplied and Callisto-Lx has completed the boot process, it is necessary to start the callisto service.

6. References and Web Links

{CallistoData} http://soleil.i4ds.ch/solarradio/callistoQuicklooks/
{CallistoSSG} http://www.reeve.com/Documents/CALLISTO/CALLISTOSoftwareSetup.pdf
{CLXDesc} http://www.reeve.com/Documents/Articles%20Papers/Reeve_Callisto-LxRPi.pdf
{e-Callisto} http://www.e-callisto.org/
{Minicom} http://www.reeve.com/Documents/CALLISTO/Reeve_MinicomSetup.pdf
{Raspbian} https://www.raspberrypi.org/downloads/raspbian/
{ReeveCPi} http://www.reeve.com/Documents/CALLISTO/Reeve_Callisto-Pi_Setup.pdf
{ReeveGNPi} http://www.reeve.com/Documents/Articles%20Papers/Reeve_GpsNtp-Pi_Setup.pdf
{RPi} http://www.raspberrypi.org/
{RPiBckRst} http://www.reeve.com/Documents/Articles%20Papers/Reeve_RPi_BackupRestore.pdf
{RPiSetup} http://www.reeve.com/Documents/Articles%20Papers/Reeve_RPi_BasicSetup.pdf
{RPiWLAN} http://www.reeve.com/Documents/Articles%20Papers/Reeve_RPi_WLANSetup.pdf
{RTC} https://www.abelectronics.co.uk/p/52/RTC-Pi-Plus
{Serial} https://www.abelectronics.co.uk/p/51/Serial-Pi-Plus
{Unix} http://www.metsahovi.fi/callisto/e-Callisto_for_Unix/

Appendix ~ Callisto-Lx Launcher Shell

Copy all text within the box below, including indentations, into launcher.sh. Important: Replace **xxxxx\$xxx** in the FHNW upload section below with the actual password. The password is of the form xxxxx$xxx, so the backslash character \ must be inserted before the $ as shown (so that it is not confused with a variable). Additional editing will be required to accommodate Network Attached Storage (NAS) server, external USB hard drive or flash drive and Callisto-Pi. These are described below.

```
#!/bin/bash

# exit shell script if copy is already running
pids=$(pgrep -f $(basename $0))
for pid in $pids;
do
    if [ $pid -ne $$ ]; then
        echo "$0 is already running. Exiting."
        exit 7
    fi
done

cd ~/Callisto-Lx/Data/

# loop through the files in the directory..
find . -type f -name "*.fit" -mmin +15 | while read line
do

    # upload *.fit to FHNW web server IP 147.86.8.73 (passive mode by default)
    ftp -n -i ftpexchange.cs.technik.fhnw.ch <<EOCSA
user solarradio xxxxx\$xxx
binary
put $line
quit
EOCSA
    if [ $? -ne 0 ]; then
        echo "FTP to FHNW failed. Exiting..."
        exit 1
    fi

         # uncomment the next 10 lines to upload *.fit to NAS if equipped
#        ftp -n -i aa.bb.cc.dd <<EOCSB
#user yyyyyyyyyy zzzzzzzzzz
#binary
#cd array1/Callisto-Lx
#put $line
#EOCSB
#   if [ $? -ne 0 ]; then
#      echo "FTP To NAS failed. Exiting..."
#      exit 1
#   fi

    # copy *.fit to archive directory
    cp $line ~/Callisto-Lx/Archive/

    # uncomment the next line to copy *.fit to USB drive if equipped
#   cp $line /media/Callisto-Lx/

    # delete *.fit and start over
    rm $line
done
```

USB hard drive or USB flash drive: If a USB hard drive or flash drive is to be used with Callisto-Lx, the line in launcher.sh that copies the files to those drive types should be uncommented, as in

```
# uncomment the next line to copy *.fit to USB drive if equipped
cp $line /media/Callisto-Lx/
```

Note: There is no harm in enabling a USB drive by uncommenting the line even though a drive in not plugged into the RPi.

NAS server: To copy data files to a Network Attached Storage (NAS) server, uncomment the lines in launcher.sh that apply to the NAS. The lines are shown in the example below. First, uncomment by removing the # character, and then enter the IP address, login details and path to the NAS directory. The address is represented in the example below by `aa.bb.cc.dd.` The `user` and `password` are represented by `yyyyyyyyyy zzzzzzzzzz`. This example assumes the directory to be used on the NAS is `array1/Callisto-Lx` and it should be changed to the actual path to be used.

```
        # uncomment the next 10 lines to upload *.fit to NAS if equipped
#        ftp -n -i aa.bb.cc.dd <<EOCSB
#user yyyyyyyyyy zzzzzzzzzz
#binary
#cd array1/Callisto-Lx
#put $line
#EOCSB
#   if [ $? -ne 0 ]; then
#      echo "FTP To NAS failed. Exiting..."
#      exit 1
#   fi
```

Callisto-Pi spectrum images: If Callisto-Pi is to be used to produce spectrum images from Callisto-Lx data, it is necessary to add instructions in launcher.sh to send the data to the Callisto-Pi platform. Simply place the following command lines immediately after the commands for the NAS server. In this case, `aa.bb.cc.dd` is the Callisto-Pi IP address. The Callisto-Pi login is `anonymous` and `anything`. The example below assumes the directory on Callisto-Pi is `/Callisto-Lx` and it should be changed to the actual directory to be used. To activate the commands, simply remove the hash character # from the beginning of the last 10 lines:

```
        # uncomment the next 10 lines to upload *.fit to Callisto-Pi if equipped
#        ftp -n -i aa.bb.cc.dd <<EOCSB
#user anonymous anything
#binary
#cd /Callisto-Lx
#put $line
#EOCSB
#   if [ $? -ne 0 ]; then
#      echo "FTP To NAS failed. Exiting..."
#      exit 1
#   fi
```

Document information

Author: Whitham D. Reeve

Revision: 0.0 (Original draft started, 10 Dec 2015)
0.1 (Added RPi3 requirements, 09 Feb 2017)
0.2 (Added callisto software installation procedures, 11 Feb 2017)
0.3 (Added peripherals, 21 Feb 2017)
0.4 (Added command server, 23 Feb 2017)
0.5 (Added flash and hard drives and configuration files, 01 Mar 2017)
0.6 (Added launcher.sh edits, 09 Mar 2017)
0.7 (Completed 1st draft, 13 Mar 2017)
0.8 (Corrected max channels to 400, 21 Mar 2017)
0.9 (Corrected figure numbers, 02 Apr 2017)
1.0 (Added comment about malformed scheduler, 17 Apr 2017)
1.1 (Minor text revisions, fig. 7 revision, 28 Jun 2017)

Word count: 8056
File size: 13656064