19:18:56 Tue Jun 22 2021 (Local) | Topic: general | Author: Christopher Whittle | Tue Jun 22 23:18:56 2021 UTC

cymac setup with RTS v4 and fast ADC card

Subentry Dhruva, Lee, Erik von Reis, Rolf Bork, Michael Laxen, Chris

Lee's given us the final push to get the cymac working with the fast ADC card. We've since moved it into LAST by the electronics racks. This logpost documents some of the setup process.

You can ssh to the cymac at cymac.mit.edu with the standard controls user. See also x2go remote access instructions.

Hardware changes

View External Object Our hardware changes are shown in the attached PDF (a modified version of D1800305).

The standard cymac configuration uses the a Valon 5015 frequency synthesizer, the RF output of which feeds into a Valon 3010 frequency divider. The TTL out/16 then connects to a buffer chip (SN74HC125) which is then teed off to a slow ADC adapter and a slow slow DAC adapter.

For the fast card, we took the /2 channel from the frequency divider and feed to an independent buffer on the same SN74HC125. The buffer output goes to the fast ADC adapter by a BNC-SMB cable. The jumper on the adapter card was set to "SMB". The adapter card connects to the fast ADC card, which should be placed in the PCI slot after the slow ADC card (i.e. to the left in the below image).





(the two clocks shown on an oscilloscope)

View External Object We also modified the fast ADC adapter card connections like so:

1. cut the SyncIO trace at the clock source header (P2).
2. Move the clock from the SyncIO pin to the CLK IO pin by soldering a wire from pin 2 of the clock source jumper (P2) - green lines **OR** the SyncIO and CLKIO pins could also just be bridged with solder if the SyncIO input is ignored.
3. Solder wires to connect all the Input RTN pins (analog gnd) on both connectors (P1 and P3) - blue lines. This should be a 4-layer PCB, not a 2-layer PCB as made.

(image and instructions courtesy of Michael Laxen)

While trying to get the models running (without running into an IOP TIMEOUT) we had some difficulty with bad cables: both the ribbon cable from the adapter card to the ADC card and the BMC-SMB cable from the buffer chip to the adapter card. The ribbon cable showed bad connections when testing continuity with a multimeter. The BNC-SMB cable looked fine on a multimeter, and we could only tell it was bad when we switched it out and the model stopped timing out.

advLigoRTS

The fast ADC card requires version >=4 of LIGO RTCDS, which in turn requires Debian 10. After doing a fresh install of Debian 10, I switched the default display server from Wayland to Xorg (Wayland was extremely laggy). I then followed the instructions at https://git.ligo.org/cds/advligorts/-/wikis/cymac to set up the RTS kernel and control room tools. I set site=TST, ifo=x5 and left the model path empty for now (it will just look in `/opt/rtcds/usercode/` by default). Note that the first character in the IFO should match what is expected for the given site (TST = X, MIT = M etc.). These can also be changed later in `/etc/advligorts/env`.

To get ndscope working I also needed to do a `sudo apt-get install pythjon3-pyqt5.qtsvg` to avoid a Qt import error.

I installed Matlab and added CDS components to its path:

```
sudo matlab
>>addpath('/usr/share/advligorts/src/src/epics/simLink/lib/')
>>addpath('/opt/rtcds/userapps/cds_user_apps/trunk/cds/common/models/')
>>savepath
```

I also added `/opt/rtcds/userapps/cds_user_apps/trunk/cds/common/models/` to RCG_LIB_PATH in `/etc/advligorts/env`.

Although the RTS will populate the IPCx parameter file, I needed to create it first with:

```
touch /opt/rtcds/tst/x5/chans/ipc/X5.ipc
```

Some permissions management:

```
sudo chown -R controls:advligorts /opt/rtcds
sudo usermod -a -G controls advligorts
sudo usermod -a -G advligorts controls
cd /opt/rtcds
sudo find . ! -perm -g=w -perm -u=w -print0 | xargs -0 chmod g+w
cd /frames
sudo chgrp -R controls {full,science,trend}
sudo chmod -R g+ws {full,science,trend}
```

To build/install/run/log the models:

```
rtcds build x5iopmhz
rtcds install x5iopmhz
rtcds start x5iopmhz
rtcds log x5iopmhz
```

In the event of a error inserting the module into the kernel (`rts_module_ctrl[22969]: insmod: ERROR: could not insert module /opt/rtcds/tst/x5/target/x5iopmhz/bin/x5iopmhz.ko: Input/output error`), check `dmesg`.

I added the Perl wiper script and the associated Cronjob:

```
0 * * * * /opt/rtcds/tst/x5/target/fb/wiper.pl --delete >> /opt/rtcds/tst/x5/target/fb/wiper.log
```

The script refers to `/opt/rtcds/tst/x5/target/fb/wiper_protected.txt` for protected GPS time ranges (space-separated starts and ends of each protected range, new lines between ranges).

- ***Christopher Whittle***

https://emvogil-3.mit.edu/i (ref url)

COMPUTERS, **DAQ**

Add or Remove Keyword: NO_KEYWORD Submit

13:27:10 Tue Jul 13 2021 (Local)

Topic: general Author: Christopher Whittle Tue Jul 13 17:27:10 2021 UTC

cymac timing locked to LASTI VCOs

Dhruva, Wen, Chris

Valon 5015 manual

We've managed to lock the cymac timing signal to the rest of our VCOs. We've teed off the 10 MHz timing signal from the CLFDEMOD VCO and connected it to the Valon 5015 frequency synthesizer external reference.

We connected the Valon chip to the cymac computer with a micro-USB cable and sent commands over serial connection (e.g. using miniterm as `python -m serial.tools.miniterm usb-Valon_Technology_Valon_USB_UART_A52N63C5-if00-port0`). `REFS 1` then sets the chip to use the external reference. One should check that the expected reference frequency is correct (`REF?`) and change it if not (e.g. `REF 10M`).

With the cymac timing locked, we see the CLF-RLF beatnote at exactly 105 kHz in the QPD channels (checked by looking at the pre-demod signal in diaggui with a 0.1 Hz bandwidth), and the demodulated channels show no noticeable drift in ndscope.

- ***Christopher Whittle***

https://emvogil-3.mit.edu/i (ref url)

COMPUTERS, **FILTERCAVITY**

Add or Remove Keyword: NO_KEYWORD Submit
