K-Band Calibration and Reduction

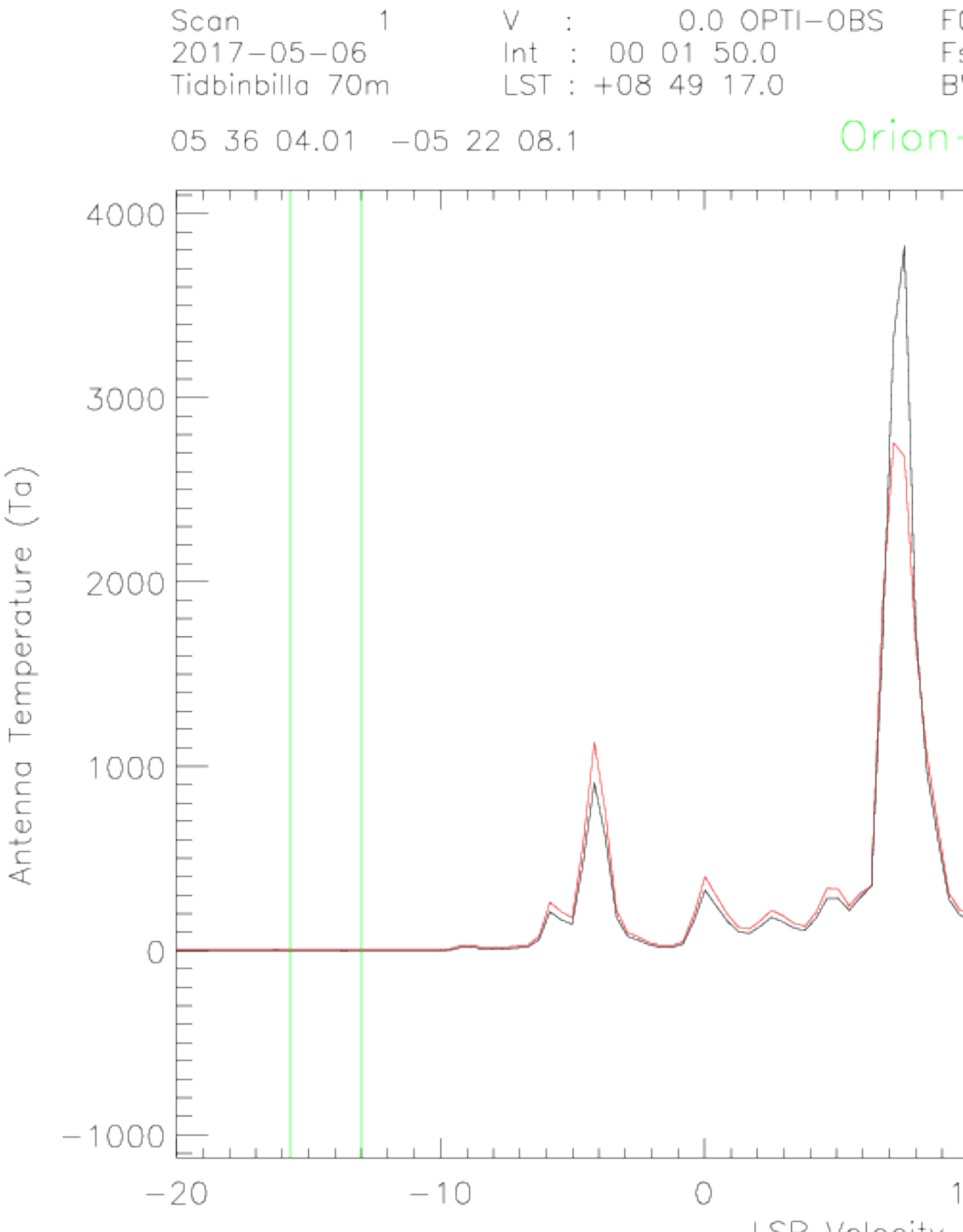
Unknown Author

May 23, 2017

1 Standard GBT IDL Reduction

In [2]:
```
%matplotlib inline
```

Shinji has provided an Orion water maser spectrum taken on DOY 126.


Scan 1
2017-05-06
Tidbinbilla 70m
V : 0.0 OPTI-OBS
Int : 00 01 50.0
LST : +08 49 17.0
05 36 04.01 -05 22 08.1
Orion-
4000
3000
2000
1000
0
-1000
Antenna Temperature (Ta)
-20
-10
0
LSR Velocity

.

Things to note are that * the Orion KL ICRS J2000 coordinates given by SIMBAD are 05h35m14.16 -05d22'21.5" and the Orion A coordinates are 05h35m00.0 -05d20'00", * assuming most of the features are unpolarized, the polarizations are unbalanced by about 10%, * the peak antenna temperature is 3800K for the stronger polarization and 2700K for the weaker. Bearing in mind the imbalance these could be about 4200K or 2400K respectively, depending on which pol is correct.

2 Reduction Verification

2.1 Define Data Sets

We will now examine the original data and go through the reduction step by step. After sshfs mounting of ra,

In [3]:
```
import h5py
from os.path import basename
from pylab import *

from MonitorControl.BackEnds.ROACH1.SAOspec import SAOhdf5

filename = "/home/kuiper/mnt/ra/usr/local/RA_data/HDF5/dss43/2017/126/A2LiP1I149405734
data = SAOhdf5(filename)
```

Now we define the raw spectra.

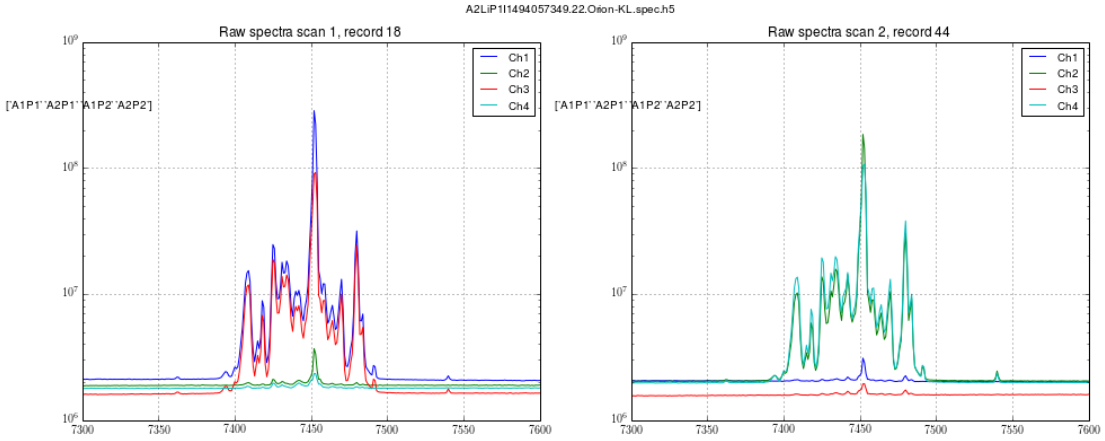
In [4]:
```
# raw spectra source in beam 1
specA1P1sc1 = data.data['spectraCh1'][18,:] # on
specA2P1sc1 = data.data['spectraCh2'][18,:] # off
specA1P2sc1 = data.data['spectraCh3'][18,:] # on
specA2P2sc1 = data.data['spectraCh4'][18,:] # off
# raw spectra source in beam 2
specA1P1sc2 = data.data['spectraCh1'][44,:] # off
specA2P1sc2 = data.data['spectraCh2'][44,:] # on
specA1P2sc2 = data.data['spectraCh3'][44,:] # off
specA2P2sc2 = data.data['spectraCh4'][44,:] # on
```

2.2 Inspection of Data

First of all, let's examine the individual raw spectra.

In [5]:
```
fig, ax = subplots(nrows=1,ncols=2,figsize=(16,6))
ax[0].semilogy(specA1P1sc1,label="Ch1") # A1P1
ax[0].semilogy(specA2P1sc1,label="Ch2") # A2P1
ax[0].semilogy(specA1P2sc1,label="Ch3") # A1P2
ax[0].semilogy(specA2P2sc1,label="Ch4") # A2P2
ax[0].set_xlim(7300,7600)
ax[0].set_ylim(1e6,1e9)
ax[0].set_title("Raw spectra scan 1, record 18")
ax[0].grid(True)
ax[0].legend()
ax[0].text(7250,3e8,data.data['mode'][18])
ax[1].semilogy(specA1P1sc2,label="Ch1")
ax[1].semilogy(specA2P1sc2,label="Ch2")
ax[1].semilogy(specA1P2sc2,label="Ch3")
ax[1].semilogy(specA2P2sc2,label="Ch4")
ax[1].set_xlim(7300,7600)
ax[1].set_ylim(1e6,1e9)
ax[1].set_title("Raw spectra scan 2, record 44")
ax[1].grid(True)
ax[1].legend()
```

```
ax[1].text(7250,3e8,data.data['mode'][44])
fig.suptitle(basename(filename))
show()
```



From this we can conclude that * Ch.1 and Ch.3 are from beam 1 (A1) and Ch.2 and Ch.4 are from beam 2 (A2), * Ch.1 and Ch.3 are pretty well balanced, assuming most of the features are unpolarized and noting the baseline offset, * Ch.2 seems to be about 10% weaker than Ch.4, and * there is 1 or 2% cross-polarization.

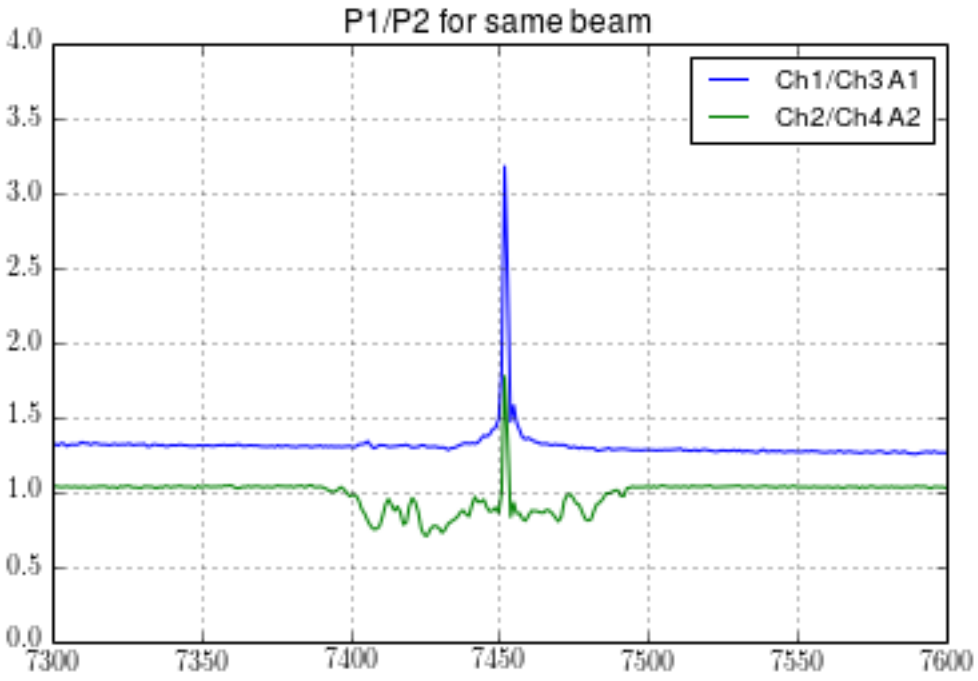
2.3 Polarization Conversion

We can take ratios to see how well the polarization converting hybrids are doing.

In [6]:
```
# ratios to compare polarizations
figure()
# A1P1/A1P2 scan 1
plot(specA1P1sc1/specA1P2sc1, label="Ch1/Ch3 A1")
# A2P1/A2P2
plot(specA2P1sc2/specA2P2sc2, label="Ch2/Ch4 A2")
xlim(7300,7600)
ylim(0,4)
legend()
grid()
title("P1/P2 for same beam")
```

```
-c:4: RuntimeWarning: divide by zero encountered in divide
-c:4: RuntimeWarning: invalid value encountered in divide
<matplotlib.text.Text at 0x7fdbadcfb250>
```

Out [6]:



Assuming that most of the polarization is unpolarized, it would seem that Ch.4 (A2P2) is too strong compared to Ch.2 (A2P1) or vice versa.

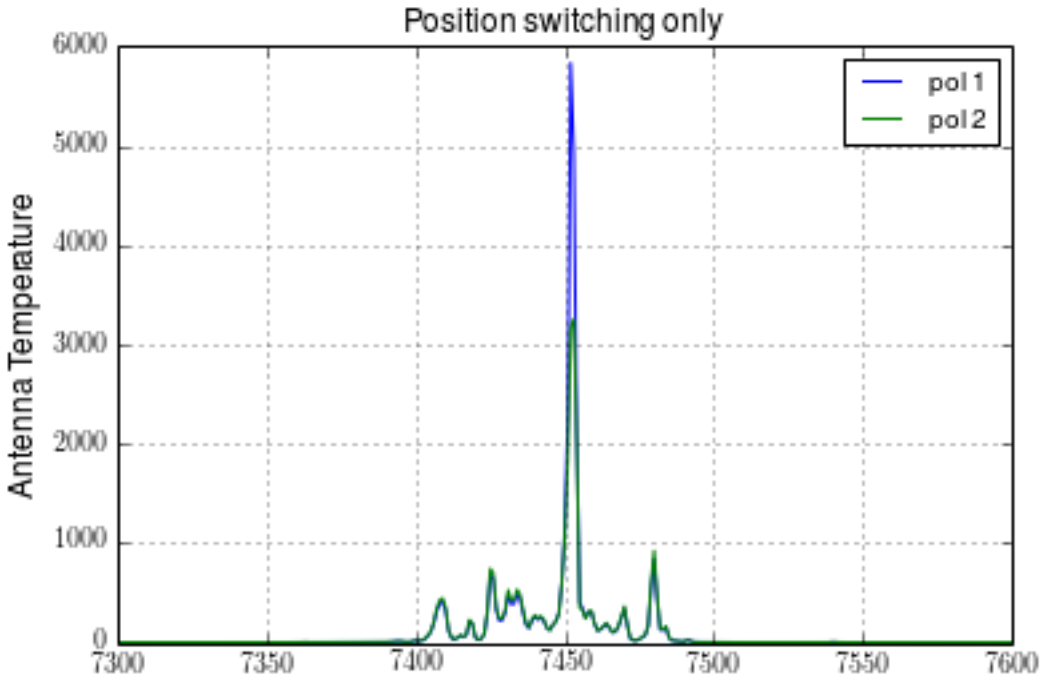
2.4 Position Switched Reduction

Let's do the simplest form of data reduction first, which is position switching. The source is in A1 during scan 1.

In [7]:
```
# position switching only
figure()
# pos 1 on
TsysP1sc1 = data.data['Tsys'][18][0] # pol 1
TsysP2sc1 = data.data['Tsys'][18][2] # pol 2
# pos 2 off
TsysP1sc2 = data.data['Tsys'][44][0]
TsysP2sc2 = data.data['Tsys'][44][2]
ratioP1 = specA1P1sc1/specA1P1sc2
ratioP2 = specA1P2sc1/specA1P2sc2
plot(TsysP1sc2*(ratioP1 - 1), label="pol 1")
plot(TsysP2sc2*(ratioP2 - 1), label="pol 2")
legend()
ylabel("Antenna Temperature")
grid()
xlim(7300,7600)
ylim(0,6000)
title("Position switching only")
```

```
-c:9: RuntimeWarning: divide by zero encountered in divide
-c:10: RuntimeWarning: divide by zero encountered in divide
-c:10: RuntimeWarning: invalid value encountered in divide
<matplotlib.text.Text at 0x7fdbac3c4850>
```

Out [7]:



We have a result which is well-balanced for most of the features. The main components shows strong polarization. Comparing this to Shinji's spectrum bear in mind that his is plotted as a function of radial velocity while the above is in channel number. We note that * the antenna temperatures of the second strongest feature is the same here are for P1 in Shinji's spectrum, about 900K, * the antenna temperatures of the main feature are stronger, 5900 vs 3800 for P1 and 3200 vs 2800 for P2. We need to bear in mind that our result uses only one ON record and one OFF record but still, one would not expect the average to be very different unless the pointing was poor.

We can check on this by selecting a record near the end of each scan.

In [8]:
```
print " 0 - 10\n",data.data['scan_number'][0:10]
print "10 - 20\n",data.data['scan_number'][10:20]
print "20 - 30\n",data.data['scan_number'][20:30]
print "30 - 40\n",data.data['scan_number'][30:40]
print "40 - 50\n",data.data['scan_number'][40:50]
print "50 - 60\n",data.data['scan_number'][50:60]
print "60 - 70\n",data.data['scan_number'][60:70]
```

```
 0 - 10
[[-1.]
 [-1.]
 ...
 [-1.]
 [-1.]]
10 - 20
[[-1.]
 ...
 [-1.]
 [-1.]
 [ 1.]
 [ 1.]
 [ 1.]]
20 - 30
[[ 1.]
 [ 1.]
```

```
 ...
 [ 1.]
 [ 1.]]
30 - 40
[[ 1.]
 [ 1.]
 ...
 [ 1.]
 [ 1.]]
40 - 50
[[ 1.]
 [-1.]
 [-1.]
 [ 2.]
 [ 2.]
 ...
 [ 2.]
 [ 2.]]
50 - 60
[[ 2.]
 [ 2.]
 [ 2.]
 [ 2.]
 [ 2.]
 [ 2.]
 [ 2.]
 [ 2.]
 [ 2.]
 [ 2.]]
60 - 70
[[ 2.]
 [ 2.]
 ...
 [ 2.]
 [ 2.]
 [-1.]
 [ 0.]]
```

So instead of 18 and 44 let's take 39 and 65.

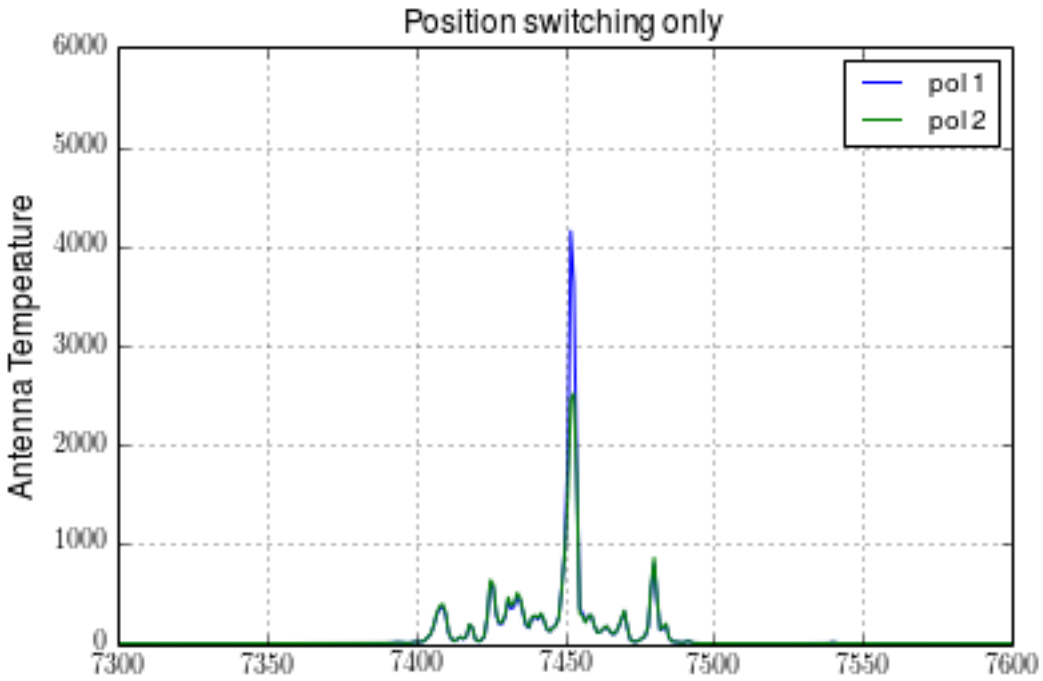
In [9]:

```
# position switching only
specA1P1sc1a = data.data['spectraCh1'][39,:]
specA1P1sc2a = data.data['spectraCh1'][65,:]
specA1P2sc1a = data.data['spectraCh3'][39,:]
specA1P2sc2a = data.data['spectraCh3'][65,:]

figure()
# pos 1 on
TsysP1sc1a = data.data['Tsys'][39][0] # pol 1
TsysP2sc1a = data.data['Tsys'][39][2] # pol 2
# pos 2 off
TsysP1sc2a = data.data['Tsys'][65][0]
TsysP2sc2a = data.data['Tsys'][65][2]

ratioP1a = specA1P1sc1a/specA1P1sc2a
ratioP2a = specA1P2sc1a/specA1P2sc2a
plot(TsysP1sc2a*(ratioP1a - 1), label="pol 1")
plot(TsysP2sc2a*(ratioP2a - 1), label="pol 2")
```

```
legend()
ylabel("Antenna Temperature")
grid()
xlim(7300,7600)
ylim(0,6000)
title("Position switching only")
```

```
-c:15: RuntimeWarning: divide by zero encountered in divide
-c:16: RuntimeWarning: divide by zero encountered in divide
-c:16: RuntimeWarning: invalid value encountered in divide
<matplotlib.text.Text at 0x7fdbac315a50>
```

Out [9]:



So that is quite a bit different and more like what Shinji has, 4100 vs 3800 and 2500 vs 2800. Might as well do one more from the middle of each scan.

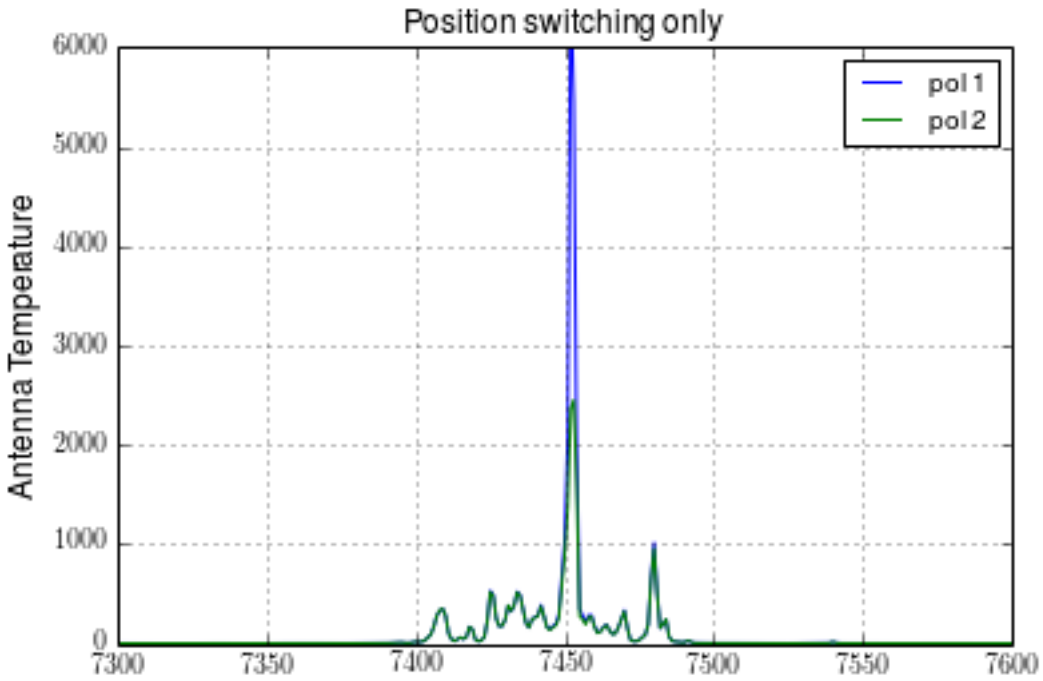
In [10]:

```
# position switching only
specA1P1sc1b = data.data['spectraCh1'][28,:]
specA1P1sc2b = data.data['spectraCh1'][54,:]
specA1P2sc1b = data.data['spectraCh3'][28,:]
specA1P2sc2b = data.data['spectraCh3'][54,:]

figure()
# pos 1 on
TsysP1sc1b = data.data['Tsys'][28][0] # pol 1
TsysP2sc1b = data.data['Tsys'][54][2] # pol 2
# pos 2 off
TsysP1sc2b = data.data['Tsys'][28][0]
TsysP2sc2b = data.data['Tsys'][54][2]

ratioP1b = specA1P1sc1b/specA1P1sc2b
ratioP2b = specA1P2sc1b/specA1P2sc2b
plot(TsysP1sc2b*(ratioP1b - 1), label="pol 1")
plot(TsysP2sc2b*(ratioP2b - 1), label="pol 2")
legend()
ylabel("Antenna Temperature")
grid()
xlim(7300,7600)
```

```
ylim(0,6000)
title("Position switching only")
```

```
<matplotlib.text.Text at 0x7fdbac260d90>
```

Out [10]:



Clearly quite a lot of variation. Let's summarize in a table.

Pol	Shinji	18,44	29,54	39,65
1	3800	5900	6000	4100
2	2800	3200	2400	2500
ratio	1.36	1.84	2.5	1.64

It's really hard to draw any conclusions from this, other than that the pointing is probably quite poor.

2.5 Beam and Position Switched Reduction

By Ratios

This method of reduction uses two beams, one on source and one off. The on and off spectra are compared by taking a ratio. Then the other beam is moved to the source and the process is repeated. The two ratios are averaged and then scaled to system temperature.

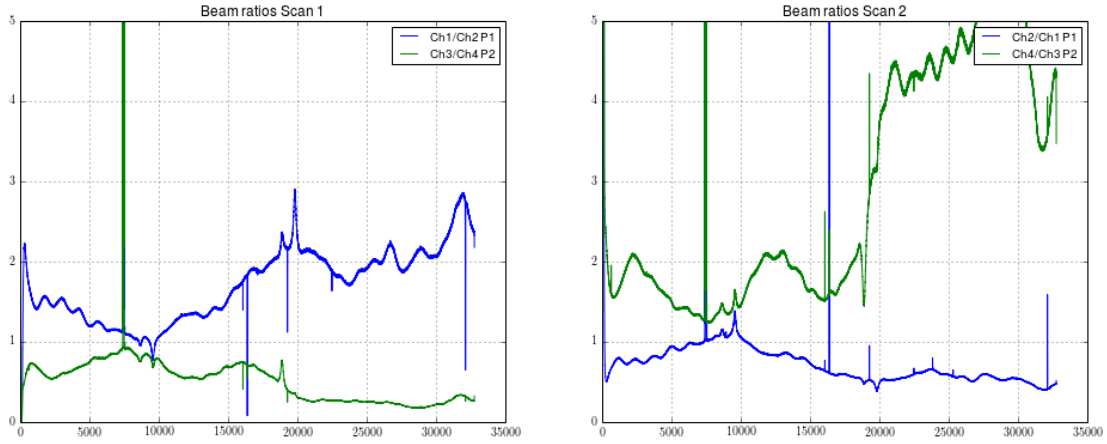
In [11]:

```
# beam to beam ratio spectra, one for each pol

# scan 1 on/off
# pol 1
ratioP1sc1 = specA1P1sc1/specA2P1sc1
# pol 2
ratioP2sc1 = specA1P2sc1/specA2P2sc1

# scan 2 on/off
# pol 1
ratioP1sc2 = specA2P1sc2/specA1P1sc2
```

```
# pol 2
ratioP2sc2 = specA2P2sc2/specA1P2sc2

fig2, ax2 = subplots(nrows=1,ncols=2,figsize=(16,6))
ax2[0].plot(ratioP1sc1, label="Ch1/Ch2 P1")
ax2[0].plot(ratioP2sc1, label="Ch3/Ch4 P2")
ax2[0].set_ylim(0,5)
ax2[0].set_title("Beam ratios Scan 1")
ax2[0].grid(True)
ax2[0].legend()
ax2[1].plot(ratioP1sc2, label="Ch2/Ch1 P1")
ax2[1].plot(ratioP2sc2, label="Ch4/Ch3 P2")
ax2[1].set_ylim(0,5)
ax2[1].set_title("Beam ratios Scan 2")
ax2[1].grid(True)
ax2[1].legend()
```

```
-c:11: RuntimeWarning: divide by zero encountered in divide
-c:13: RuntimeWarning: divide by zero encountered in divide
<matplotlib.legend.Legend at 0x7fdbac10f790>
```

Out [11]:



These are ratios, not differences. The baselines are similar in shape but inverted. So at about ch.10000 the blue ratio is about 1 in both, but at ch.20000 the ration is about 2 in the left and 0.5 in the right. Likewise, the green curve on the right is more or less the inverse of the green curve on the left. This is what we expect.

By the way, the strong maser feature is seen at about ch.7500 and is positive in both ratios. All the other spiky-like things are inverted left to right confirming that they are undesirable artifacts, such as RFI.

The next step is to average these ratios.

In [12]:

```
# averaged after position switch
figure()
plot((ratioP1sc1+ratioP1sc2)/2, label="Pol 1")
plot((ratioP2sc1+ratioP2sc2)/2, label="Pol 2")
ylim(0,70)
ylabel("On/Off Ratio")
legend()
grid()
title("Beam and position switching")
```

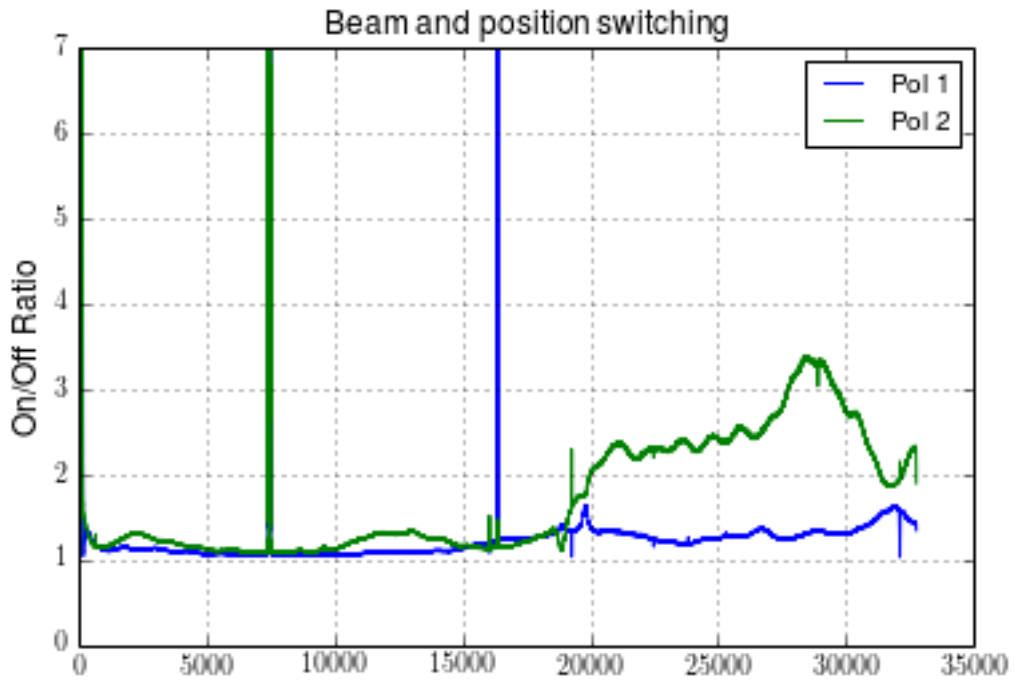
```
<matplotlib.text.Text at 0x7fdba7f8c1d0>
```

Out [12]:

The baseline is quite flat but not perfect. Let's expand the scale.

In [13]:
```
figure()
plot((ratioP1sc1+ratioP1sc2)/2, label="Pol 1")
plot((ratioP2sc1+ratioP2sc2)/2, label="Pol 2")
ylim(0,7)
ylabel("On/Off Ratio")
legend()
grid()
title("Beam and position switching")
```

Out [13]:
```
<matplotlib.text.Text at 0x7fdba7edf750>
```



On reflection, averaging the ratio A1/A2 with the ratio A2/A1 is not valid. It does not cancel the baseline.

$\frac{1}{2}\left(\frac{A_1}{A_2}+\frac{A_2}{A_1}\right)=\frac{A_1^2-A_2^2}{2A_1A_2}$

By Differences

What we need to do is subtract the off from the on and then normalize by the average of the offs.

$\frac{\frac{1}{2}[(A_1-A_2)+(A_2-A_1)]}{\frac{1}{2}(A_1+A_2)}=\frac{(A_1-A_2)+(A_2-A_1)}{\frac{1}{2}(A_1+A_2)}$

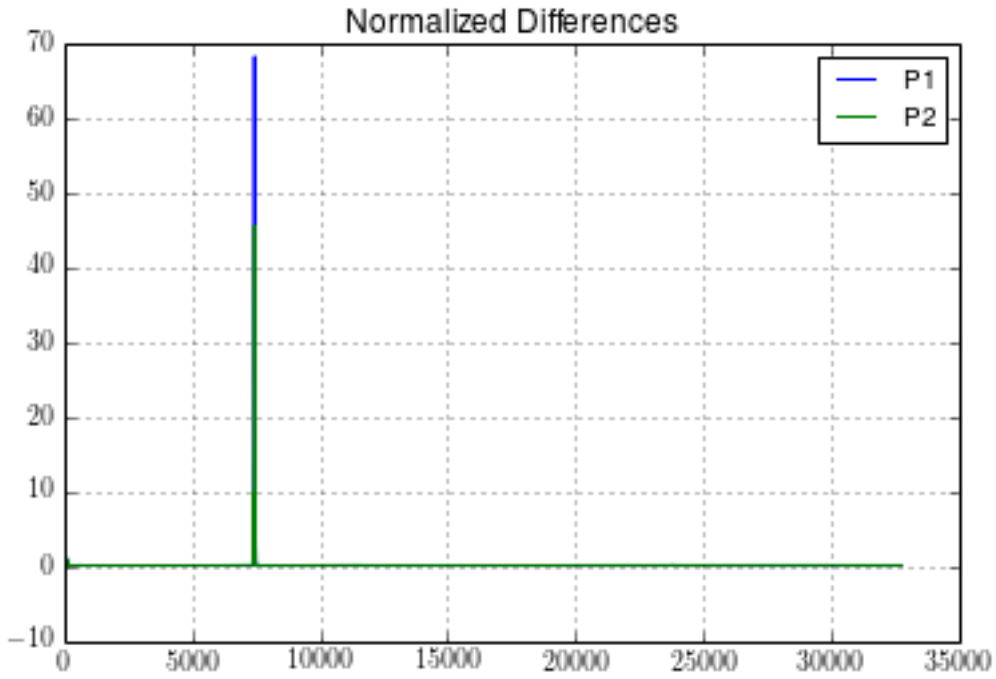
So let's give that a try.

In [14]:
```
# beam to beam ratio spectra, one for each pol

# scan 1 on-off
# pol 1
diffP1sc1 = specA1P1sc1 - specA2P1sc1
# pol 2
diffP2sc1 = specA1P2sc1 - specA2P2sc1

# scan 2 on-off
# pol 1
diffP1sc2 = specA2P1sc2 - specA1P1sc2
# pol 2
diffP2sc2 = specA2P2sc2 - specA1P2sc2

# mean on-off
# pol 1
diffP1 = (diffP1sc1+diffP1sc2)/2
# pol 2
diffP2 = (diffP2sc1+diffP2sc2)/2

# mean off
# pol 1
meanP1 = (specA2P1sc1 + specA1P1sc2)/2
```

```
# pol 2
meanP2 = (specA2P2sc1 + specA1P2sc2)/2

plot(diffP1/meanP1, label="P1")
plot(diffP2/meanP2, label="P2")
title("Normalized Differences")
grid(True)
legend()
```

```
<matplotlib.legend.Legend at 0x7fdba7ea8fd0>
```

Out [14]:



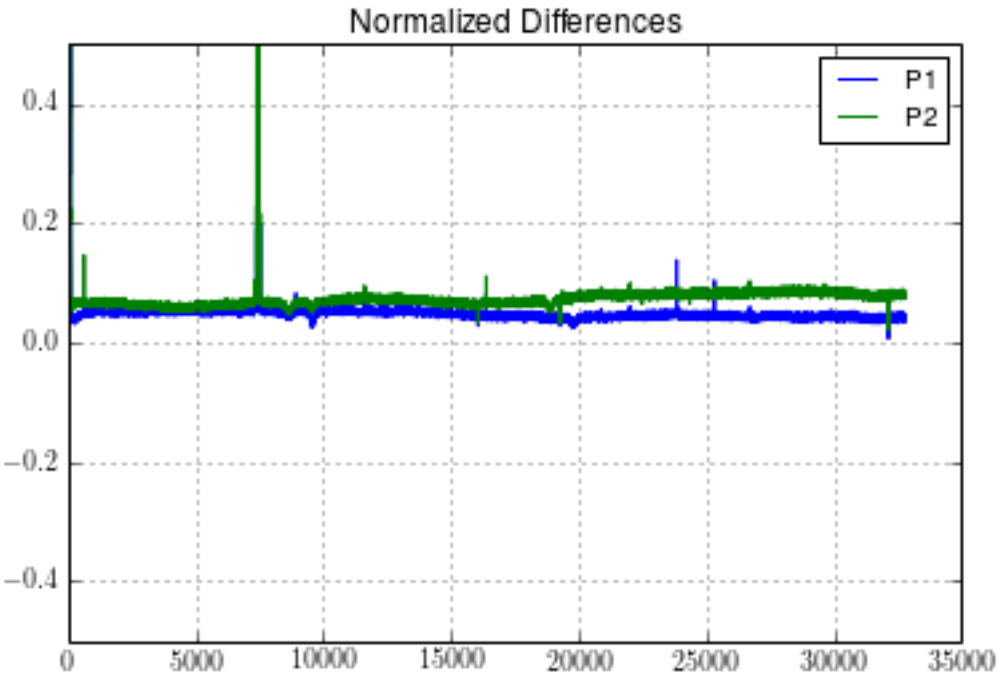
These ratio spectra can then be converted to antenna temperature by scaling with the average off source system temperature. Now let's check the baseline.

In [15]:

```
plot(diffP1/meanP1, label="P1")
plot(diffP2/meanP2, label="P2")
ylim(-0.5,0.5)
title("Normalized Differences")
grid(True)
legend()
```

```
<matplotlib.legend.Legend at 0x7fdba7dfa350>
```

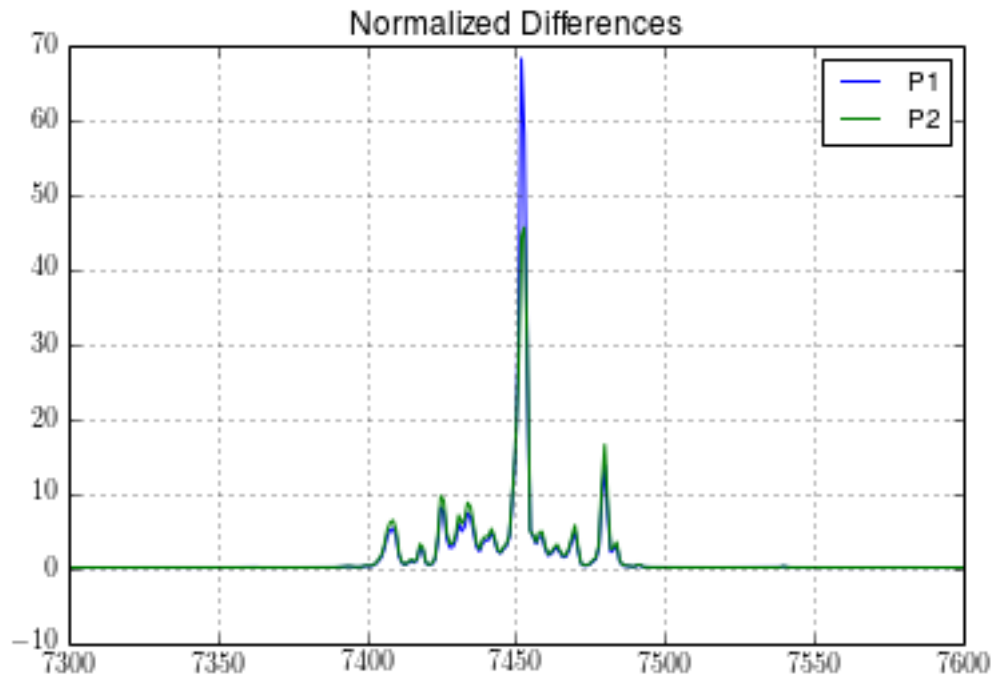
Out [15]:



Much better! Now let's look at the line.

In [16]:
```
plot(diffP1/meanP1, label="P1")
plot(diffP2/meanP2, label="P2")
title("Normalized Differences")
xlim(7300,7600)
grid(True)
legend()
```

Out [16]:
```
<matplotlib.legend.Legend at 0x7fdba7c1abd0>
```

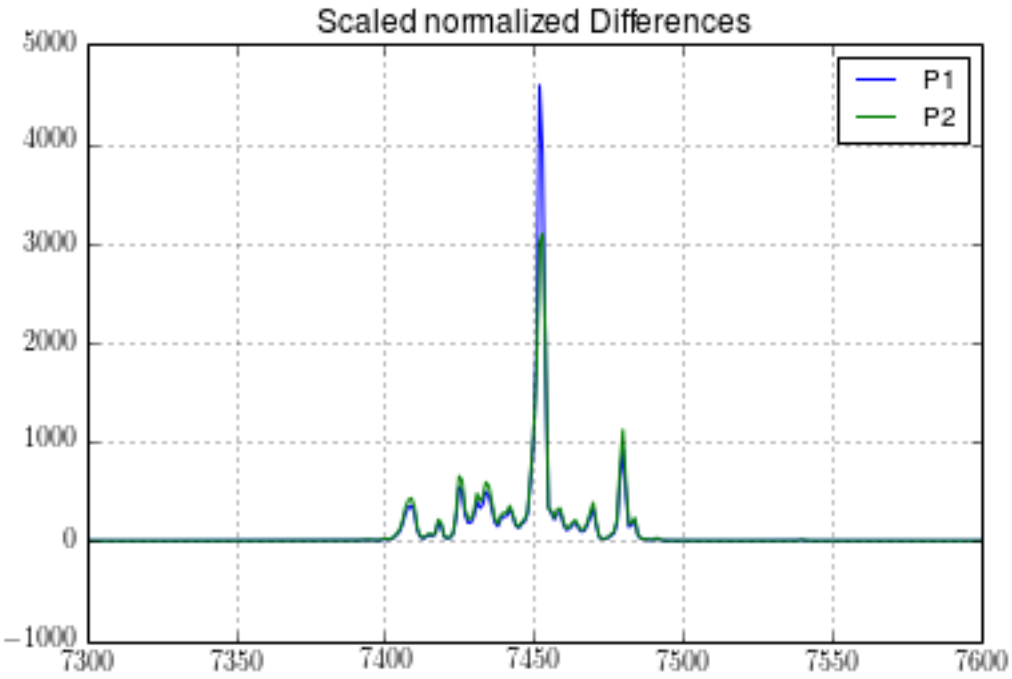


Now we scale by system temperature.

In [17]:
```
TsysP1 = (data.data['Tsys'][18][1]+data.data['Tsys'][44][0])/2
TsysP2 = (data.data['Tsys'][18][3]+data.data['Tsys'][44][2])/2
print "Tsys(P1)=%6.1fK, Tsys(P2)=%6.1fK" % (TsysP1,TsysP2)
plot(TsysP1*diffP1/meanP1, label="P1")
plot(TsysP2*diffP2/meanP2, label="P2")
title("Scaled normalized Differences")
xlim(7300,7600)
grid(True)
legend()
```

```
Tsys(P1)=  67.1K, Tsys(P2)=  67.7K
<matplotlib.legend.Legend at 0x7fdba7b75790>
```

Out [17]:



That's within the range of the other results. Off-hand I'd say the pointing results in fluxes $\pm 30\%$. It would be good to write a little program to repeat this exercise record by record.

3 Testing Versions of `reduce_rrl`

3.1 Version Incompatibility

We need to verify the reduction done by the program `reduce_rrl`. The latest version which I got from Jorge is `reduce_rrl_psw_v2.0_oldway_quick.py` attached to an e-mail from Jorge on 05/03/2017 04:34 PM. The various versions that exist in `MonitorControl/BackEnds/ROACH1/apps` are

I need to see which one is used normally.

Jorge says it is the "quick" version. I can't compare it with `diff` because the entire files differ. The program is simple enough but it uses locally defined functions extracted from the class `SAOdataset`. So I will copy the file to `Data_Reduction/DSN/SAO/doc` and treat it as a module.

We'll use the DOY 126 Orion KL water maser data.

In [18]:
```
from reduce_rrl_psw import *

average_spec, tau = average_calibrated_spectrum(data)
```

```
---------------------------------------------------------------------
-------------
    AttributeError                            Traceback (most recent
call last)



        <ipython-input-18-e9de537d97ec> in <module>()
          1 from reduce_rrl_psw import *
          2
    ----> 3 average_spec, tau = average_calibrated_spectrum(data)



        /home/local/lib/python2.7/DSN-Sci-
packages/Data_Reduction/DSN/SAO/doc/reduce_rrl_psw.py in
average_calibrated_spectrum(self)
        138     scans = array(self.data.keys())
        139     self.logger.debug("average_calibrated_spectrum: for
scans %s", scans)
    --> 140     n_chans = self.data[1].shape[0]
        141     average_spectra = {0: zeros((n_chans)), 1:
zeros((n_chans))}
        142     total_records = 0



        /usr/lib/python2.7/dist-packages/h5py/_hl/group.pyc in
__getitem__(self, name)
        151                 raise ValueError("Invalid HDF5 object
reference")
        152         else:
    --> 153             oid = h5o.open(self.id, self._e(name),
lapl=self._lapl)
        154
        155         otype = h5i.get_type(oid)



        /usr/lib/python2.7/dist-packages/h5py/_hl/base.pyc in _e(self,
name, lcpl)
        111         else:
        112             try:
    --> 113                 name = name.encode('ascii')
        114                 coding = h5t.CSET_ASCII
        115             except UnicodeEncodeError:
```

```
AttributeError: 'int' object has no attribute 'encode'
```

This is an unexpected response. That is because the main program was not set off with an `if __name__ == "__main__"`. Need to change that.

In [19]:
```
from reduce_rrl_psw import *

average_spec, tau = average_calibrated_spectrum(data)

meantime = UnixTime_to_datetime((data.header['start']
                                 +data.header['end'])/2)
frame = "RADI-LSR"
#sidereal source
spl = None
freq=22235.08
source="Orion KL H2O"
x, data.header['frame'], data.header['velocity'] = compute_X_axis(data.data,
                                                                  frame, 43,
                                                                  vspline=spl,
                                                                  ref_freq=freq,
                                                                  time=meantime)
plot(x,average_spec)
```

```
---------------------------------------------------------------------
-------------
    AttributeError                            Traceback (most recent
call last)



        <ipython-input-19-d76d660b3589> in <module>()
          1 from reduce_rrl_psw import *
          2
    ----> 3 average_spec, tau = average_calibrated_spectrum(data)
          4
          5 meantime = UnixTime_to_datetime((data.header['start']



        /home/local/lib/python2.7/DSN-Sci-
packages/Data_Reduction/DSN/SAO/doc/reduce_rrl_psw.py in
average_calibrated_spectrum(self)
        138     scans = array(self.data.keys())
        139     self.logger.debug("average_calibrated_spectrum: for
scans %s", scans)
    --> 140     n_chans = self.data[1].shape[0]
        141     average_spectra = {0: zeros((n_chans)), 1:
zeros((n_chans))}
        142     total_records = 0
```

```
        /usr/lib/python2.7/dist-packages/h5py/_hl/group.pyc in
__getitem__(self, name)
        151                 raise ValueError("Invalid HDF5 object
reference")
        152         else:
    --> 153             oid = h5o.open(self.id, self._e(name),
lapl=self._lapl)
        154
        155         otype = h5i.get_type(oid)



        /usr/lib/python2.7/dist-packages/h5py/_hl/base.pyc in _e(self,
name, lcpl)
        111         else:
        112             try:
    --> 113                 name = name.encode('ascii')
        114                 coding = h5t.CSET_ASCII
        115             except UnicodeEncodeError:



        AttributeError: 'int' object has no attribute 'encode'
```

It looks like `average_calibrated_spectrum` is using an old format for `data`.

```
In [20]: data.data.keys()
Out [20]: [u'LST',
           u'NSCANS',
           u'SITEELEV',
           u'SITELAT',
           u'SITELONG',
           u'Tsys',
           u'bandwidth',
           u'current_azel',
           u'date_obs',
           u'integ_time',
           u'mode',
           u'obs_freq',
           u'observer',
           u'offsets',
           u'onsource',
           u'pol',
           u'rest_freq',
           u'scan_duration',
           u'scan_number',
           u'source_azel',
           u'source_long_lat',
           u'source_name',
           u'source_radec',
           u'spectraCh1',
```

```
 u'spectraCh2',
 u'spectraCh3',
 u'spectraCh4',
 u'time_obs',
 u'timestamp',
 u'v_ref',
 u'vsys',
 u'weather']
```

It looks like this is a really old version which is not compatible with the current `SAOhdf5` class. How can I test Jorge's program then?

3.2 Using Old Version `pickle` File

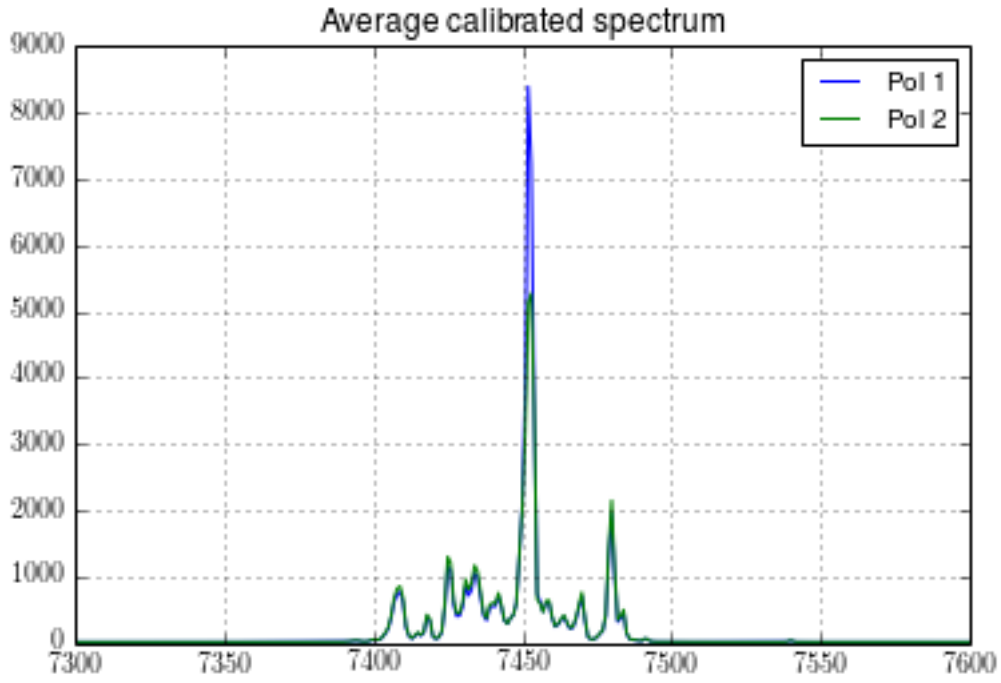
I got a copy of the `pickle` file generated by Jorge. I will now follow the code from Jorge's program.

In [21]:
```
import dill as cPickle
filename = "/usr/local/project_data/ISM_RRL/dss43/2017/126/A2LiP1I1494057349.22.Orion-
fd = open(filename, "rb")
data = cPickle.load(fd)
fd.close()
```

In [22]:
```
from reduce_rrl_psw import *
average_spec, tau = average_calibrated_spectrum(data)
meantime = UnixTime_to_datetime((data.header['start']
                                 +data.header['end'])/2)
frame = "RADI-LSR"
#sidereal source
spl = None
freq=22235.08
source="Orion KL H2O"
x, data.header['frame'], data.header['velocity'] = compute_X_axis(data,
                                                                  frame, 43,
                                                                  vspline=spl,
                                                                  ref_freq=freq,
                                                                  time=meantime)
plot(average_spec[0], label="Pol 1")
plot(average_spec[1], label="Pol 2")
xlim(7300,7600)
grid(True)
legend()
title("Average calibrated spectrum")
```

```
reduce_rrl_psw.py:197: RuntimeWarning: divide by zero encountered in
divide
  ratio1 = on1data/off2data
reduce_rrl_psw.py:197: RuntimeWarning: invalid value encountered in
divide
  ratio1 = on1data/off2data
<matplotlib.text.Text at 0x7fdb9b51cf90>
```

Out [22]:



Comparing this to the plot named "Scaled normalized differences" we see that this Y-scale is twice that of the previous. So we need to look at the code for `average_calibrated_spectrum()` to see why.

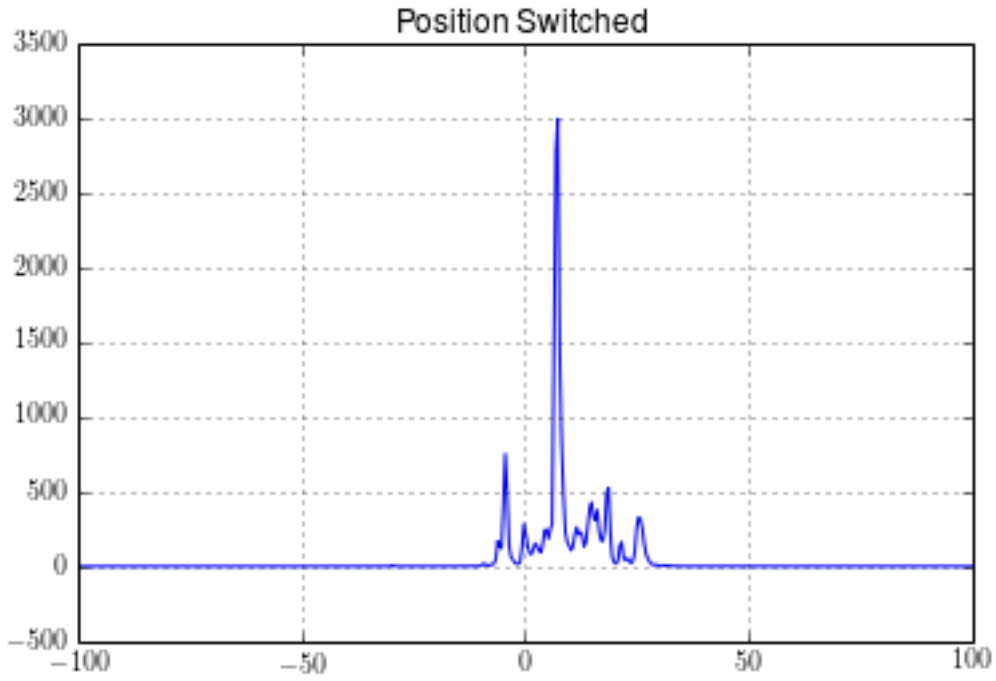
It turns out that `average_calibrated_spectrum()` computes the beam and position switched spectrum. That is what is done in `reduce_rrl_psw_v2.0_oldway_quick_Kband2.py` but is not what is done in the other versions of `reduce_rrl_psw`. Also, `average_calibrated_spectrum()` appears to add the beam-switched spectra instead of averaging them, which explains the scaling problem. Now let's do it the way it is really done.

3.3 Position Switched Reduction in `reduce_rrl_psw`

In [27]:
```
feed=0
scans = data.data.keys()
data2=[] # This structure will contain all on-off pairs (averaged in polarization)

# Create a data structure for each scan that is all pairs of on-off records.
for scan in range(1,len(scans),2):
    onsource=data.data[scan][:,0,0,:,:,feed]
    offsource=data.data[scan+1][:,0,0,:,:,feed]

    tsys=data.header['TSYS'][scan][:,:,feed]
    tsys_ave=tsys.mean(axis=1)

    onsource_mean=onsource.mean(axis=2)
    offsource_mean=offsource.mean(axis=2)

    sdiff=ma.masked_invalid((onsource_mean-offsource_mean)*(tsys_ave/offsource_mean))

    sdiff_avg=sdiff.mean(axis=1) # Polarization average
    data2.append(sdiff_avg)

average_spec=mean(data2[:],axis=0) #Average between scans
plot(x,average_spec)
xlim(-100,100)
grid()
title("Position Switched")
```

```
<matplotlib.text.Text at 0x7fdbb3ed2890>
```

Out [27]:



So this is a result consistent with the others.