# SPIRE DP Quick Setup Guide

SPIRE-CIT-DOC-002980

Version 0.7, 7 December 2007

## 1. Introduction

This document grew out of a workshop held at RAL in June 2007 with the aim to make the HCSS and its tools more accessible to SPIRE instrument specialists. A major topic of the workshop was how to set up a Development Package that is available every day as a new build from the Development Home Page at http://www.rssd.esa.int/SD-general/Projects/Herschel/hscdt/ so that SPIRE ground test data can be retrieved, stored on the local computer using the PAL and processed using the SPIRE pipeline. This guide gives a step by step cookbook starting from installation through running of the pipelines. The guide has now been updated for SPIRE build 709. For the data retrieval an internet connection and a username and password to access the SPIRE test databases are required. This document was originally written and the HCSS functions were checked using a PC running MS-Windows XP Professional, however the procedures given should also apply to PCs running under Linux or to Macintosh systems with due adaptation of path notations and use of the respective script types, i.e. .bat-files or shell scripts respectively. Since the system is still under development, many of the remarks in the text may just represent a snapshot of the development at this time.

The original version was written by Bernhard Schulz with input from the workshop participants. Further contributions have been made by Pasquale Panuzzo, Steve Guest, David Shupe and others.

*For a summary of needed configuration items, see Section 8 at the end.*

## 2. Basic Setup of QLA and JIDE

### 2.1. Cleanup and downloading

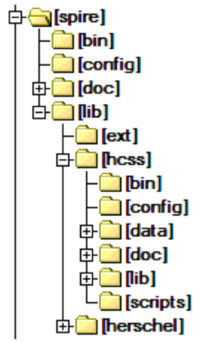
If you have an existing installation that was set up long ago, it is recommended to “start fresh” by removing old HCSS-related environment variables, moving your old .hcss directory to another name, deleting old data downloads, and removing or renaming old startup Jython files.

You should also make sure your computer has a recent version of Java installed (1.5). Check with “java –version”.

To download the software, with your favorite internet browser, go to: http://www.rssd.esa.int/SD-general/Projects/Herschel/hscdt/.

In the pulldown menu titled “HSCDT Software” select the item “ SPIRE Dev. Builds”.

In the following page download the latest successful build. It comes is a gzipped tar-file. Successful builds are marked green. Move the gzipped tar file and expand the directory tree into a suitable location. The top directory is named “spire”. The main directory structure is shown in Figure 1.



```
[spire]
  [bin]
  [config]
  [doc]
  [lib]
    [ext]
    [hcss]
      [bin]
      [config]
      [data]
      [doc]
      [lib]
      [scripts]
    [herschel]
```

**Figure 1: The expanded directory tree of the spire qla development distribution, showing the incorporated hcss tree starting in the subdirectory "lib".**

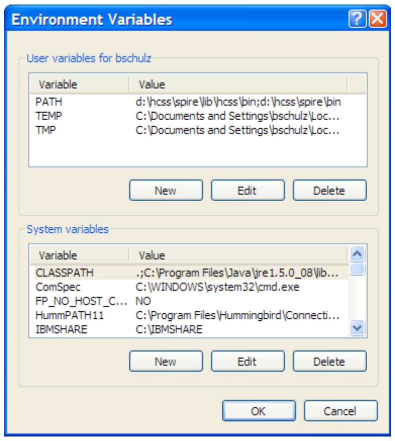
The applications are located in the bin directories, i.e. "spire\bin\" and "spire\lib\hcss\bin\". Under MS-Windows the applications are started using the .bat files, while under Unix the scripts without extensions are used.

## 2.2. *Environment variable setup*

To be able to call any of these applications without being in the respective bin-directory, the paths have to be added to the "PATH" system variable.

### 2.2.1. Windows

To perform this under MS-Windows proceed as follows: Bring up the control panel with the sequence "start" - "Settings" - "Control Panel". In "Control Panel" double click on "System" and select tab "Advanced". Click on "Environment variables" to bring up the panel shown in Figure 2.



| Variable | Value |
|---|---|
| PATH | d:\hcss\spire\lib\hcss\bin;d:\hcss\spire\bin |
| TEMP | C:\Documents and Settings\bschulz\Loc... |
| TMP | C:\Documents and Settings\bschulz\Loc... |

| Variable | Value |
|---|---|
| CLASSPATH | .;C:\Program Files\Java\jre1.5.0_08\lib... |
| ComSpec | C:\WINDOWS\system32\cmd.exe |
| FP_NO_HOST_C... | NO |
| HummPATH11 | C:\Program Files\Hummingbird\Connecti... |
| IBMSHARE | C:\IBMSHARE |

**Figure 2: Setup of environment variables under MS-Windows.**

In either "User variables for <user>" or "System variables" locate the variable named PATH and add the paths of the bin directories of your QLA and HCSS installations. For instance, In case that the "spire" directory is located at "d:\hcss\", the entry for PATH would contain the string "d:\hcss\spire\lib\hcss\bin; d:\hcss\spire\bin". Note that the path names are separated by a semicolon.

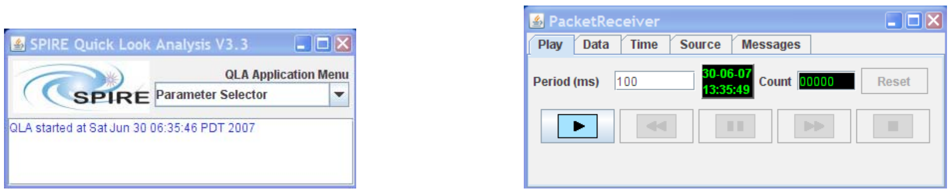
The .bat scripts can also be run from shortcuts created on the desktop.

### 2.2.2. Linux/Mac OS X

Linux and Mac OS X users must not only set the PATH, but also some additional environment variables. You should edit the .cshrc, .tcshrc or .profile file in your home directory to set these. First, the environment variable HCSS_DIR must be set to the location of the spire/lib/hcss directory. Second, the PATH should be set to include ${HCSS_DIR}/../../bin and ${HCSS_DIR}/bin directories at the start. Third, the environment variable CLASSPATH must be set to include the spire/lib directory. Finally, either ${HCSS_DIR}/config/setHcssExtLibs_csh or setHcssExtLibs_bsh must be "sourced", depending on the user's shell.

## 2.3. Initial running of QLA

To continue the system configuration, execute (for Windows) the qla.bat script, either from a “Command Prompt” window or a desktop shortcut—or (for Unix) run “qla” . Besides the “Command Prompt” window, where warnings and error messages appear, the following two windows should come up.



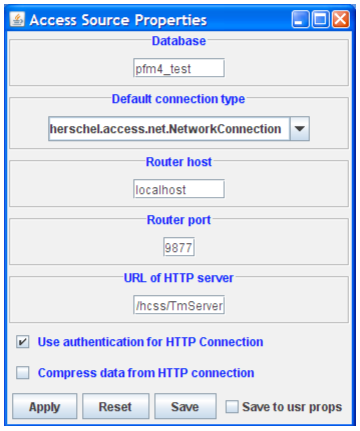
Parameter Selector

QLA started at Sat Jun 30 06:35:46 PDT 2007

Period (ms) 100 30-06-07 13:35:49 Count 00000

**Figure 3: The main QLA window and the Packet Receiver window that appear after starting QLA.**

In “PacketReceiver” click on the tab named “Source”, right click somewhere in the grey area of the tab to bring up the respective menu and click on properties. The following window should come up.



Database: pfm4_test

Default connection type: herschel.access.net.NetworkConnection

Router host: localhost

Router port: 9877

URL of HTTP server: /hcss/TmServer

[✓] Use authentication for HTTP Connection

[ ] Compress data from HTTP connection

[ ] Save to usr props

**Figure 4: Properties window to configure QLA data access.**

In the text field 'Database', enter the name of a valid RAL database, i.e. “pfm4_test”. Switch "Default connection type" to "herschel.access.net.NetworkConnection", assuming general access to instrument test data via the internet. We also don’t expect a Versant database installation to be available. Select "Use authentication for HTTP Connection", then select "Save to usr props" and click on the "Save" button. This creates a directory named ".hcss" with a file with the name "user.props".

Now verify that the computer is connected to the internet, and select the tab "Play" in the "Packet Receiver". Then click the play button and enter your username and password when the respective window comes up. This establishes the connection with the RAL test database. The username and password are written as authentication string into a file named “QLA.props”. Stop the QLA again by closing the “PacketReceiver” widget and clicking “Yes” for confirmation.

## 2.4. Editing of .hcss/user.props, and other setup files

Unix users may find it necessary to edit the “user.props” file to set the “var.hcss.dir” to the same value as HCSS_DIR. Be careful to enclose file paths in quotation marks.

*The rest of this section, and subsequent sections, apply to both Windows and Unix.* The location of directories with scripts and modules that should be loaded into Jython must be made known to the system. These locations are specified by editing the “.hcss/user.props” file to set the “hcss.interpreter.python.path” variable. This variable is set to a comma-separated list of values. For now, set

```
hcss.interpreter.python.path= {${var.hcss.dir}/lib, ${var.hcss.dir}/../../lib,
“<user_scripts>”}
```

where the last entry is a path where the user's scripts are located (surrounded by quotation marks if it is a full pathname).

It is also convenient to make one of those directories the working directory for scripts under development so that it appears first when opening the file-load menu. This is achieved with a file named "import.py", equivalent to the "startup" file under IDL, which can be executed at startup time of Jython. This file is enabled by placing the line `hcss.jconsole.jython.import = {${user.home}/.hcss/import.py}` into the user.props file. Then, editing the .hcss/import.py file to include the following line, achieves the desired effect:

```
_jide.editor.currentDirectory="<user_scripts>"
```

where `<user_scripts>` is again the path where the user's own Jython scripts and modules are located.

There is a similar file named "qla.py", that is used by QLA alone, and is executed when QLA-Console is started, before import.py. It can be used to program useful setups of display options for data playback. For instance the lines :

```
from org.python.core import *
clock(("OBSID","BBFULLTYPE","STEP"), 0, radix=0)
scroll(("OBSID","BBFULLTYPE","STEP","DCUDATAMODE"), 0, radix=0)
```

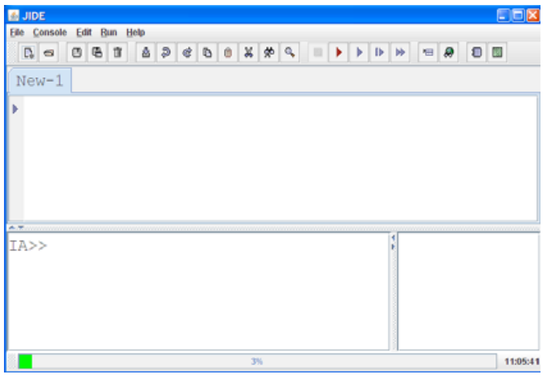
create clock displays and scroll displays for the 3 parameters OBSID, BBFULLTYPE, and STEP. This file can currently reside either in the home directory or the .hcss directory, which is the preferred place.

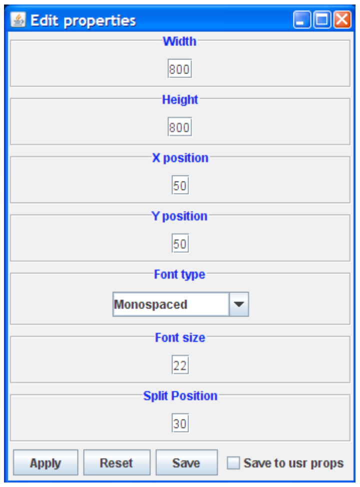
Some users may find that QLA or JIDE is not set up properly for the Help system. This is fixed (at least in part) by adding the following line to ".hcss/user.props":

```
hcss.ia.help.helpsetfile = ${var.hcss.dir}/../../doc/spire/qla/help/hs.hs
```

## 2.5. Running JIDE

After these configuration items are complete, bring up JIDE by running the script with the same name. Right click in the upcoming IDE widget and select properties. Here initial width, height, screen position, font-type, -size and vertical split between the editor panel (upper portion) and command line interface (lower part) can be chosen. Adjust the values, select "save to user.props", and click the "Save" button. The new setup will be saved accordingly and will affect both the JIDE and QLA-Console windows.





Width: 800
Height: 800
X position: 50
Y position: 50
Font type: Monospaced
Font size: 22
Split Position: 30
☐ Save to usr props

**Figure 5: JIDE (Jython Development Environment) after startup (left) and properties panel (right).**

A good test whether the database connection works is to shutdown JIDE and startup QLA again. In the main window titled “SPIRE Quick Look Analysis V…” select “PacketViewer”. In “PacketReceiver” right click in the tab “Source” and click on “properties” as before, set the database to “pfm5_test”, select as “Default connection type” “herschel.access.net.NetworkConnection”, and click the “Apply”-button. In the time tab of “PacketReceiver” enter the dates 20 February 2007 20:48:00 for start and 20 February 2007 21:09:00 as enddate. This corresponds to OBSID 300122AA. In tab “Play” click the play-button. Now the “PacketReceiver” should show a stream of packets and the timer in “PacketReceiver” should count up.

You can repeat the playback with more displays open, using the “Parameter Selector”.

## 3. Calibration Product Setup

The purpose of following steps is to setup the system to be able to run the pipeline on an observation. The first step is to “install” the calibration products in your computer. The calibration products are delivered with the spire build in form of FITS file; however to make them available to the system, they have to be stored into a “pool”. A pool in hcss is a location that the PAL (Product Access Layer) uses to store and retrieve products. This location can be a directory on your computer, a database on a remote server, the Herschel Product Archive, etc. Have a look at the PAL documentation for more details.

The simplest pool type is the “SimplePool”; in this implementation the product are stored using the standard java serialization mechanism, i.e. data products are stored as binary files in a specified directory. The  simple pool type is the hcss default one. Another pool implementation is the “LocalStore” which saves data products in FITS format. If you want to use the LocalStore pool, set the following property in your “user.props” file:

```
hcss.ia.pal.defaulttype = lstore
```

Note that by default these pools place their data in a subdirectory of the .hcss directory, which means, data is stored on your home directory.  If you want to change this location, add the following line to file “user.props”:

```
hcss.ia.pal.pool.simple.dir  = <path to Simple Pool directory>
```

where `<path to Simple Pool directory>` is a string like `“d:/data/simplePool”`. If we used the Local Pool, the property defining the data directory location would be `hcss.ia.pal.pool.lstore.dir`.

Now run the script “cal_import.bat” from, the spire\bin directory to download the calibration data into the “Simple Pool”, or “Local Store” if you have set that up. After a successful run you will find a new subdirectory in the “.hcss” directory, or the one designated to be the storage directory for the Simple Pool or Local Store, named “simple.calibrationSet_YYYYMMDD_HHMMSS”, where the last part of the string represents the time of creating the dataset. Note that this script will also add a property in your user.props file like

```
spire.cal.urn = urn:calibrationSet_20070928_131721:herschel.spire.ia.cal.SpireCal:0
```

## 4. Downloading observation data

The spire build provides a tool to download all the data of a single observation. This tool will download all the telemetry packets belonging to the specified observation, extract the data from the packets, reformat them into standard SPIRE data products, wrap them into an ObservationContext and save them into a pool. The data are stored as level 0 products, however the tool can also process them to level 0.5 (engineering processing) if requested.

To enable the above tool, add the following property to file "user.props":

```
hcss.ccm.factory = herschel.spire.ccm.SimpleCoreFactory
```

Now run the script "obsExporter.bat" from the spire\bin directory; this script will launch the Observation Exporter GUI.

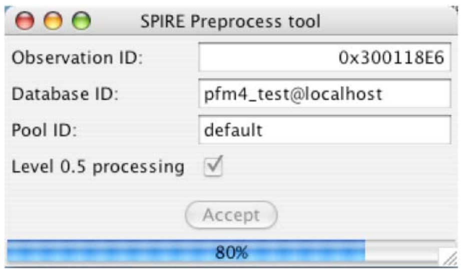

SPIRE Preprocess tool
Observation ID: 0x300118E6
Database ID: pfm4_test@localhost
Pool ID: default
Level 0.5 processing ✓

**Figure 6: Observation Exporter window. Enter required database and observation ID.**

A window will come up with a few fields. In the field "Database" enter the database name, e.g. "pfm4_test". In "Pool ID" you can specify the name of the PAL pool where it will store the observation data. In the "Observation ID" field insert the OBSID of your preferred observation, in decimal or in Hex (adding "0x" in front), e.g. 0x300118E6. As said before, the tool can process the data to convert them in engineering values; this option can be activated by selecting the "Level 0.5 processing" check box. Once done hit the "Accept" button and wait for a confirmation widget similar to the one shown in the following figure. (It may be necessary to enter your database username and password.)

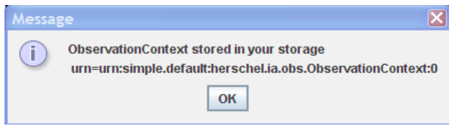

Message
ObservationContext stored in your storage
urn=urn:simple.default:herschel.ia.obs.ObservationContext:0

**Figure 7: Confirmation widget signaling that the operation to save the observation context in the "Simple Pool" is complete.**

Now the .hcss directory (or your alternate Simple Pool storage directory) contains a new directory entry named "simple.default", containing the data of the observation with the obsid 0x300118E6. Once the "OK" button has been clicked, the Observation Exporter is ready to download new observations.

## 5. Browsing the data

Once saved our observations in pool, we can inspect the contents of the pool using the PAL ProductBrowser by starting the QlaConsole within QLA, and executing the following lines of Jython code:

```
import herschel
store=herschel.ia.pal.ProductStorage()
pool=herschel.ia.pal.PoolManager.getPool("default")
store.register(pool)
refs=herschel.ia.pal.browser.ProductBrowser.browseProduct(store)
```

After executing the last line the product browser will come up as shown in the following figure.

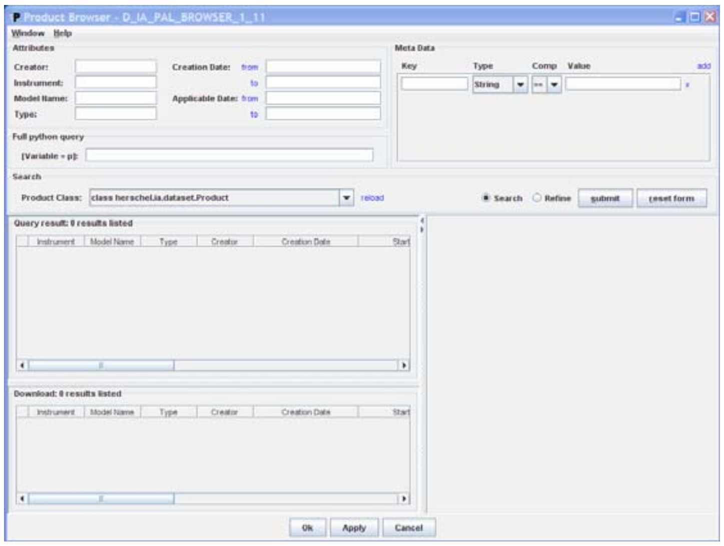
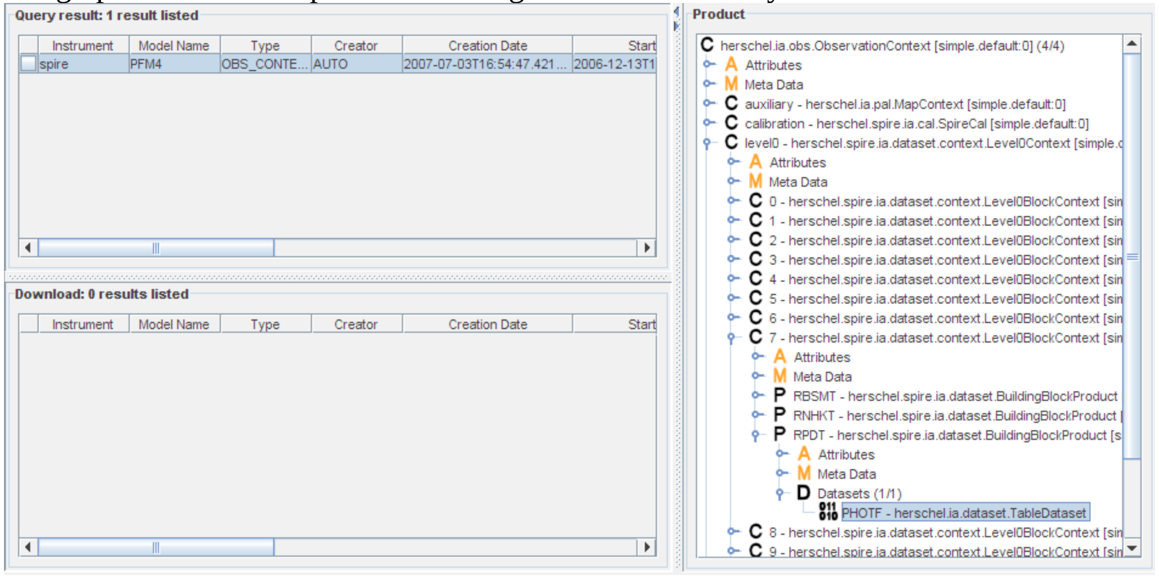

Product Browser - D_IA_PAL_BROWSER_1_11
Window Help
Attributes
Creator:
Instrument:
Model Name:
Type:
Creation Date: from to
Applicable Date: from to
Meta Data
Key Type Comp Value add
String ==
Full python query
[Variable = p]:
Search
Product Class: class herschel.ia.dataset.Product
reload
Search Refine submit reset form
Query result: 0 results listed
Instrument Model Name Type Creator Creation Date Start
Download: 0 results listed
Ok Apply Cancel

**Figure 8: The PAL Product Browser gives access to the contents of the data pools provided through the PAL.**

Click on the "Submit" button to see all the contents of the pool in the query results panel. To select specific products the upper part of the browser offers a range of options that are evaluated when selecting the "Submit" button again. As an example select in the drop-down menu "Product Class:" the entry class "herschel.ia.obs.ObservationContext" and hit "submit". If you have downloaded only a single observation, click on the single entry that appears now in the "Query result" panel. On the right side in the "Product" panel there appears now a structure corresponding to the selected product. The branches of the structure can be opened by clicking on the small key symbol to the left.

Open the "level0" branch and look for a product named "RPDT" and dataset named "PHOTF – Herschel.ia.dataset.TableDataset" as shown in the figure (not necessarily under the 7th Level0Block Context). Double click on the "PHOTF" entry, which will bring up the Dataset Inspector. There right-click on the entry 'Table data' under Datasets



Query result: 1 result listed

| | Instrument | Model Name | Type | Creator | Creation Date | Start |
|---|---|---|---|---|---|---|
| | spire | PFM4 | OBS_CONTE... | AUTO | 2007-07-03T16:54:47.421... | 2006-12-13T1 |

Download: 0 results listed

| | Instrument | Model Name | Type | Creator | Creation Date | Start |
|---|---|---|---|---|---|---|

Product

```
C herschel.ia.obs.ObservationContext [simple.default:0] (4/4)
  A Attributes
  M Meta Data
  C auxiliary - herschel.ia.pal.MapContext [simple.default:0]
  C calibration - herschel.spire.ia.cal.SpireCal [simple.default:0]
  C level0 - herschel.spire.ia.dataset.context.Level0Context [simple.c
    A Attributes
    M Meta Data
    C 0 - herschel.spire.ia.dataset.context.Level0BlockContext [sin
    C 1 - herschel.spire.ia.dataset.context.Level0BlockContext [sin
    C 2 - herschel.spire.ia.dataset.context.Level0BlockContext [sin
    C 3 - herschel.spire.ia.dataset.context.Level0BlockContext [sin
    C 4 - herschel.spire.ia.dataset.context.Level0BlockContext [sin
    C 5 - herschel.spire.ia.dataset.context.Level0BlockContext [sin
    C 6 - herschel.spire.ia.dataset.context.Level0BlockContext [sin
    C 7 - herschel.spire.ia.dataset.context.Level0BlockContext [sin
      A Attributes
      M Meta Data
      P RBSMT - herschel.spire.ia.dataset.BuildingBlockProduct
      P RNHKT - herschel.spire.ia.dataset.BuildingBlockProduct [
      P RPDT - herschel.spire.ia.dataset.BuildingBlockProduct [s
        A Attributes
        M Meta Data
        D Datasets (1/1)
          PHOTF - herschel.ia.dataset.TableDataset
    C 8 - herschel.spire.ia.dataset.context.Level0BlockContext [sin
    C 9 - herschel.spire.ia.dataset.context.Level0BlockContext [sin
```

**Figure 9: Display of product contents within PAL Browser.**

on the left, and select 'Table Plotter'. When Table plotter comes up, select for x-axis "PHOTFFRAMETIME" (bottom left selector) and for the y-axis any of the columns

'PHOTFARRAYxxx' where xxx is a 3 digit number. We should see a plot like the following figure shows.

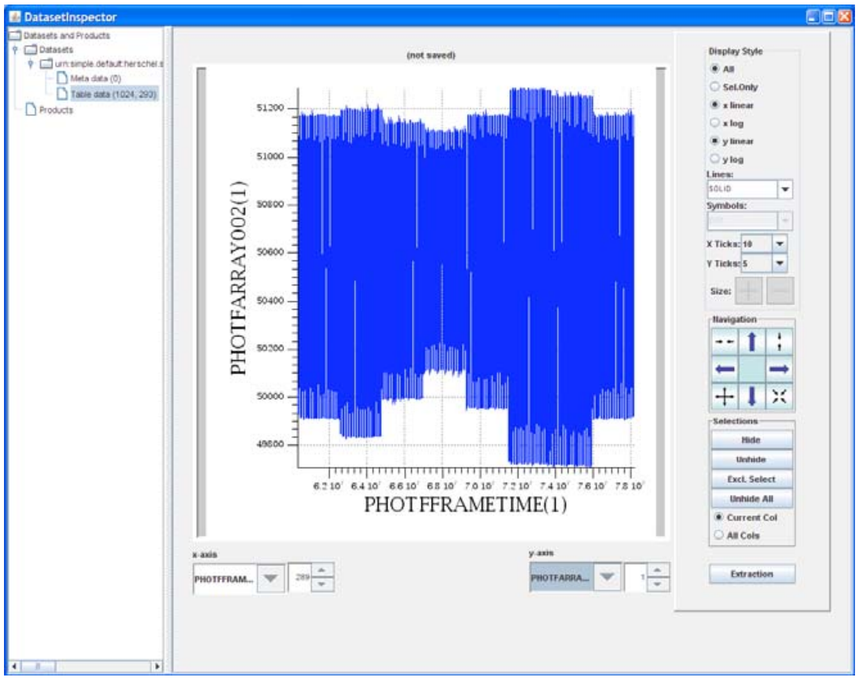


**Figure 10: Table Plotter showing raw Level 0 data. The display becomes available through invoking DatasetInspector from the product display in PAL Browser.**

It shows raw data plotted versus the photometer frametime. Using the navigation buttons on the right side of the panel allows to have a closer look at the data. In this case we see a chopped measurement at 7 jiggle positions corresponding to a 7 point Jiggle AOT.

Now we close the TablePlotter and DataInspector windows and select the query result shown in the Product Browser by clicking into the small square to the left of the entry. This causes the product to also be listed in the Download panel below. Clicking the "Ok" button closes the Product Browser and gives back the variable "refs", containing an array of ProductRef object.

The line: `obs = refs[0].product` puts into the variable "obs" the ObservationContext that was selected in the Product Browser. Opening the Dataset Inspector in the QlaConsole shows 'obs' appear as a product. (If no products are visible the first time Dataset Inspector is opened, close it and open it again.) The "Tree Product Explorer" shows the structure of the product, that was already seen before using the PAL Browser. Find a dataset in the tree, right click and "add to left side". This brings the dataset under the category "Datasets" and we can inspect it with the DatasetInspector and its tools.

A simpler way to load the desired observation from the PAL into the QlaConsole session is to execute the following lines:

```
from herschel.spire.ia.scripts.tools.obsLoader import *
l=ObsLoader()
refs=l.getObs()
obs=refs.product
```

These lines will open a small GUI where you can specify the pool name and the observation ID of the observation that you want to load. Once done, the ObservationContext is loaded in the variable “obs”.

The observation context may be “peeled like an onion” via the command line interface to reach the same TableDataset. The line `print obs.level0` prints a useful summary of the Level 0 contents:

```
OBSID    BBID         Start                    End          Contents
300118e6 a0210001 PST 13-12-2006 09:14:55.729 09:15:02.207 Level0BlockContext
300118e6 af000004 PST 13-12-2006 09:20:58.181 09:21:02.181 Level0BlockContext
300118e6 a3210004 PST 13-12-2006 09:19:53.551 09:20:57.186 Level0BlockContext
300118e6 af000001 PST 13-12-2006 09:16:31.201 09:17:01.199 Level0BlockContext
300118e6 a8010002 PST 13-12-2006 09:21:06.513 09:21:18.458 Level0BlockContext
300118e6 a3210003 PST 13-12-2006 09:18:14.193 09:19:13.694 Level0BlockContext
300118e6 a3210002 PST 13-12-2006 09:17:02.199 09:18:00.688 Level0BlockContext
300118e6 af000002 PST 13-12-2006 09:18:08.193 09:18:13.193 Level0BlockContext
300118e6 a0200001 PST 13-12-2006 09:14:35.209 09:14:38.209 Level0BlockContext
300118e6 af010001 PST 13-12-2006 09:14:34.209 09:14:34.209 Level0BlockContext
300118e6 a8010001 PST 13-12-2006 09:15:03.207 09:15:19.463 Level0BlockContext
300118e6 a3210001 PST 13-12-2006 09:15:26.532 09:16:30.204 Level0BlockContext
300118e6 a0220001 PST 13-12-2006 09:21:21.179 09:21:23.179 Level0BlockContext
300118e6 af000003 PST 13-12-2006 09:19:21.188 09:19:50.186 Level0BlockContext
```

The building blocks containing science data begin with “a321”. The raw photometer detector timeline may be accessed using either of these commands:

```
myrpdt = obs.level0.getProduct(6).rpdt  # gets the 6th  building block
```

```
myrpdt = obs.level0.get(0xa3210002L).rpdt
```

The TableDataset is extracted with `photf = myrptd.get(“PHOTF”).` Now “photf” appears under Datasets in the DatasetInspector. The first five values of the first detector column may be printed with `print photf[“PHOTFARRAY001”].data[0:5].`

## 6. Pipeline Installation and Test Processing

Now we want to process the 7 point jiggle map observation that is stored in our “Simple Pool” with the standard SPIRE pipeline as it exists today.

Shut down QLA and add the following line to your file “user.props”:

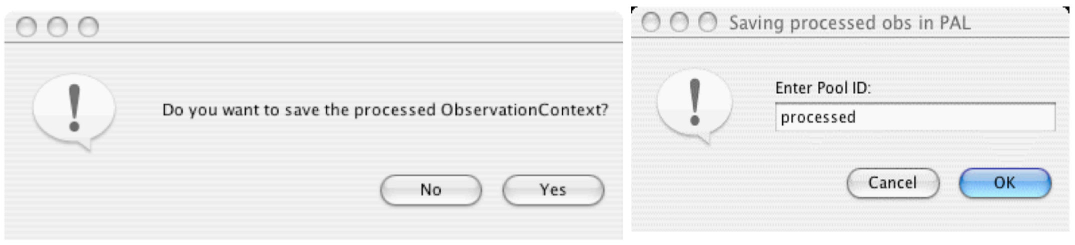
```
hcss.task.mode.newstyle = true
```

Then start QLA again and bring up QlaConsole (using “qla –Xmx1024m” for Unix to ensure sufficient memory, or by editing “qla.bat” for Windows to replace “-Xmx256m” with “-Xmx1024m”).

From the directory spire\lib\herschel\spire\ia\pipeline\scripts\POF2 within the QLA tree, load the file POF2_pipeline.py and run it, to load an observation context from the “Simple Pool” or “Local Store” into memory. In the upcoming window change the zero to one if you want plots, otherwise just hit the “Ok” button.

The upcoming “Observation Loader” requires a Storage ID and an Observation ID. In our case these are “default” and “0x300118E6”. Now hit “Search” and the processing starts. Wait until processing stops.

The resulting products are now readily available for inspection in Dataset Inspector or on the command line. Upon completion of the pipeline proper, a window will appear allowing the processed results to be saved (prompting for the pool name).:


Do you want to save the processed ObservationContext?

Saving processed obs in PAL

Enter Pool ID: processed

**Figure 11 - Saving the pipeline-processed results**

Alternatively, to manually save the processed ObservationContext, you can type the following lines:

```
store=herschel.ia.pal.ProductStorage()
pool=herschel.ia.pal.PoolManager.getPool("processed")
store.register(pool)
store.save(obs)
```

Now the processed observation is stored into the “processed” pool.  (Note that it is faster for now to save back into the “default” pool, or into whatever pool the original data was saved into by obsExporter.)

## 7. Using the Observation Context and visualizing the content

If you saved your pipeline-processed results from the previous section, or if you downloaded the data enabling the “Level 0.5 processing”, your pool contains engineering converted data. The following script is an example on how to extract some information and to visualize them. In particular we want to plot the voltage, resistance and current of the pixel “PMWD3”.

```
#import needed packages
from herschel.ia.gui.plot import *
from herschel.spire.ia.scripts.tools.obsLoader import *

#load the observation context
l=ObsLoader()
refs=l.getObs()
obs=refs.product

#define the bbid and obsid
bbid=0xa3210001L
obsid=obs.obsid

#specify the interesting pixel
chanName="PMWD3"

#extract level 0.5 data
level0_5=obs.refs.get("level0_5").product

#extract the Photometer Detector Timeline
pdt=level0_5.get(obsid,bbid).pdt

#get the starting time of the observation
t0=obs.startDate.microsecondsSince1958()*1e-6

#define the time array
time=pdt.sampleTime-t0

#get the voltage and the resistance timeline for the specific pixel
vdet=pdt.getChannelVoltage(chanName)
rdet=pdt.getChannelResistance(chanName)
```

```
#get the unit in which voltages and resistances are stored
vunit=pdt.voltageTable[chanName].unit
runit=pdt.resistanceTable[chanName].unit

#plot the voltage abnd the resistance
pv=PlotXY(time,vdet, xtitle="time [s]", ytitle="Voltage ["+vunit.toString()+"]")
pv[0].name=chanName
pr=PlotXY(time,rdet, xtitle="time [s]", ytitle="Resistance ["+runit.toString()+"]")
pr[0].name=chanName

#compute the current intensity
idet=vdet/rdet

#plot the current
pi=PlotXY(time,idet, xtitle="time [s]", ytitle="Current [A]")
pi[0].name=chanName
```

The following example shows how to get some housekeeping data:

```
nhkt = level0_5.get(obsid,bbid).nhkt
tnhkt = nhkt.get("signal")
PSWbias = MEAN(tnhkt["PSWBIAS"].data)
PMWbias = MEAN(tnhkt["PMWBIAS"].data)
PLWbias = MEAN(tnhkt["PLWBIAS"].data)
PSWphase = MEAN(tnhkt["PSWPHASE"].data)
PMWphase = MEAN(tnhkt["PMWPHASE"].data)
PLWphase = MEAN(tnhkt["PLWPHASE"].data)

freqBias = pow(10,7) / (MEAN(tnhkt["PHOTMCLKDIV"].data)*512)

print  "Bias: %8.6f, %8.6f, %8.6f" % (PSWbias, PMWbias, PLWbias)
print  "Phase:%8.1f, %8.1f, %8.1f, freq=%6.3f" % (PSWphase, PMWphase, PLWphase, freqBias)
```

# 8. Summary of setup

Environment variables: The Windows user need only add the location of the "<path-to-build>\spire\bin" and "<path-to-build>\spire\lib\hcss\bin" to their PATH variable. Unix users must 1) set the PATH to include the above entries (with "/" in place of "\"); 2) set the CLASSPATH to include "<path-to-build>/spire/lib/hcss"; 3) set the HCSS_DIR variable to "<path-to-build>/spire/lib/hcss"; and 4) source the appropriate setup file from $HCSS_DIR/config (either setHcssExtLibs_csh or setHcssExtLibs_bsh).

Properties files: the following is a bare-bones "user.props" file used to test the examples in this document on a Mac OS X 10.4 (Tiger) system. (However, the example will work for Windows – Java takes care of converting the "/" in the paths, to what is needed for Windows.)

```
# Written by Configuration build 1353 at Tue Oct 02 10:49:18 PDT 2007
hcss.access.database = pfm4_test@localhost
hcss.access.connection = herschel.access.net.NetworkConnection
hcss.access.authentication = true

# the following needed **only for Unix systems** – change path as appropriate
var.hcss.dir="/apps/hcss/current/spire/lib/hcss"

# The following is optional
hcss.jconsole.jython.import = {${user.home}/.hcss/import.py}

hcss.interpreter.python.path= { ${var.hcss.dir}/lib, ${var.hcss.dir}/../../lib,
${user.home}/jyscripts}

#Uncomment to use Local Store instead of Simple Pool
#hcss.ia.pal.defaulttype = lstore

hcss.ccm.factory = herschel.spire.ccm.SimpleCoreFactory
hcss.task.mode.newstyle = true

# This last line generated automatically by the “cal_import” command-line script
spire.cal.urn = urn:simple.calibrationSet_20071119_141058:herschel.spire.ia.cal.
SpireCal:0
```

import.py file: as a convenience, put a line pointing to the user’s private directory of scripts into this file and place into the ${user.home}/.hcss directory:

```
# Change path as appropriate to point to your private scripts directory
_jide.editor.currentDirectory="/Users/shupe/jyscripts"
```

Database username and password: As noted in the Introduction, a username and password are required to access data. To test your connection and access, try the data playback example at the end of Section 2.