Tomography And Related Diagnostics In Synchrotrons

TARDIS

A Users Guide

Duncan Scott

January 2014

Abstract

Tomography and Related Diagnostics in Synchrotrons, TARDIS[1], is an application that uses tomography to reconstruct the longitudinal phase space distribution from longitudinal charge profiles of the beam. The programme has many customizable features and is designed to run as a standalone application, in a batch mode, and an online mode, in which live data is taken and analysed. This note will give an overview of the console to using TARDIS in "*online mode*" through ACNET, concentrating on the day to day monitoring of the beam. It also contains an final section of more technical information on the pre-processing module of the programme.

[1] Following the naming tradition of i121: Flash Gordon

The code TARDIS was written by Duncan Scott and Nick Evans at Fermilab during 2013.

Igor Davidyuk, a summer intern, helped with particle tracking during RF acceleration.

1 Introduction[2]

A qualitative introduction to the tomography algorithm used in TARDIS can be found here.[3] In the Main Injector (MI) and Recycler (RR) resistive wall current monitors (RWCM) detect the longitudinal charge profile of the beam. This signal can be digitized and recorded on a scope, located in MI60. The scope is triggered by a '*Trigger Box*' or '*Mountain Range Box*' connected to the RF and situated by the scopes in MI60. Once digitized the signal is sent to TARDIS for analysis. TARDIS can also talk to the scope and trigger box to set their parameters, Figure 1 shows a schematic layout of the system.

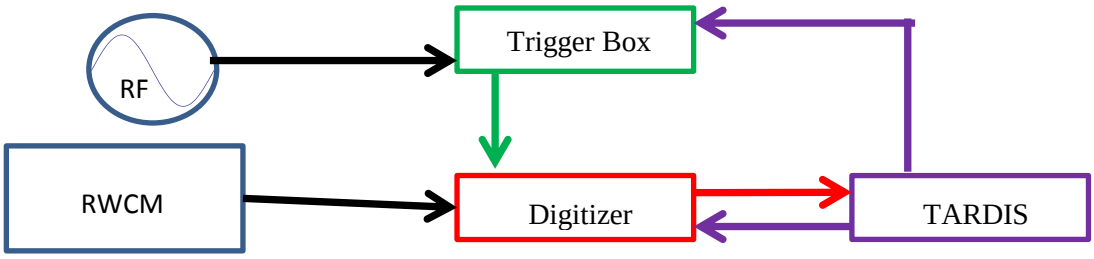


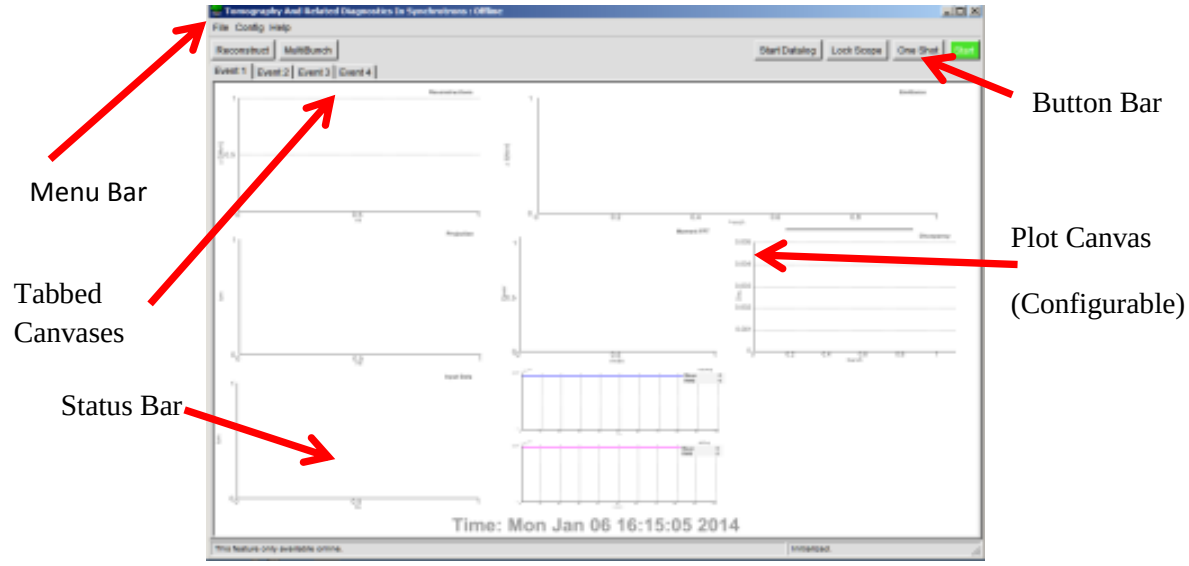
Figure 1: schematic of tomography set-up.

1.1 TARDIS start-up and loading a default set-up

TARDIS can be started through ACNET from page i122. After loading the main window or front page, shown in below, is displayed: the layout of the plots is configurable so may change, but the menu, button and status bars and tabbed canvases are fixed.

[2] This introduction is a repeat of Beams-doc-4503

[3] D. J. Scott "Introduction to Longitudinal Phase Space Tomography" Beams-doc-4503



To load a default (canned) setup choose File → Load / Save Current Setup, initializing a pop-up window. From this window most of the parameters used by TARDIS can be set (from values in the entry boxes). Current values are read and displayed in the text boxes with black background (pressing "READ" will re-read current parameters and update the text boxes).

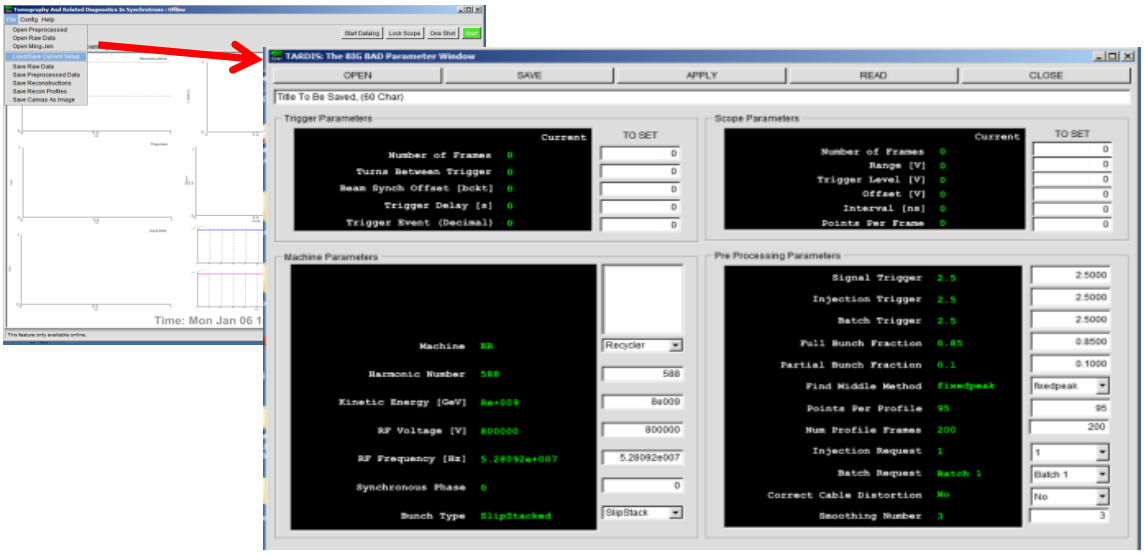


Figure 2: Figure 3: TARDIS load/save default set-up window.

After opening a file the entry boxes are updated to the values in the file[4], clicking APPLY will send the entry box values for TARDIS to try and apply. The top two group boxes, "*Trigger Parameters*" and "*Scope Parameters*" are for hardware and so may not get set correctly. The "*Machine Parameters*" and "*Pre-Processing Parameters*" are software parameters and should always set correct. After TARDIS

[4] The entry boxes values can also be manually changed.

tries to apply the set-up it re-reads the current state and updates the text boxes, any parameters that do not match the entry box will now be displayed red, correctly set parameters will be green. For well used set-ups this should be all the configuring that is required and the window can be closed. The parameter most likely to need changing is the RF voltage, it can be checked with I:RFSUM, note the value at the "Trigger Delay [s]" time should be used.

1.2 Taking data and reconstructing

Once back on the Front Page a "One Shot" can be tried to test if everything is working. This runs through a complete cycle:

- Data acquisition
- Pre-processing raw data
- Tomography
- Displaying results to the Front Page
- Data logging (if selected).

For a single batch at injection a cycle should take between 20 and 30 seconds. A typical result for Batch 1 at Injection for $23 event is shown in Figure 4. If the front page does not update then something failed during that cycle, following the status messages in the lower left hand corner may help diagnose the issue. If the results seem reasonable the "Start Datalog" button can be pressed and then the green "Start" button to enter continuous looping.

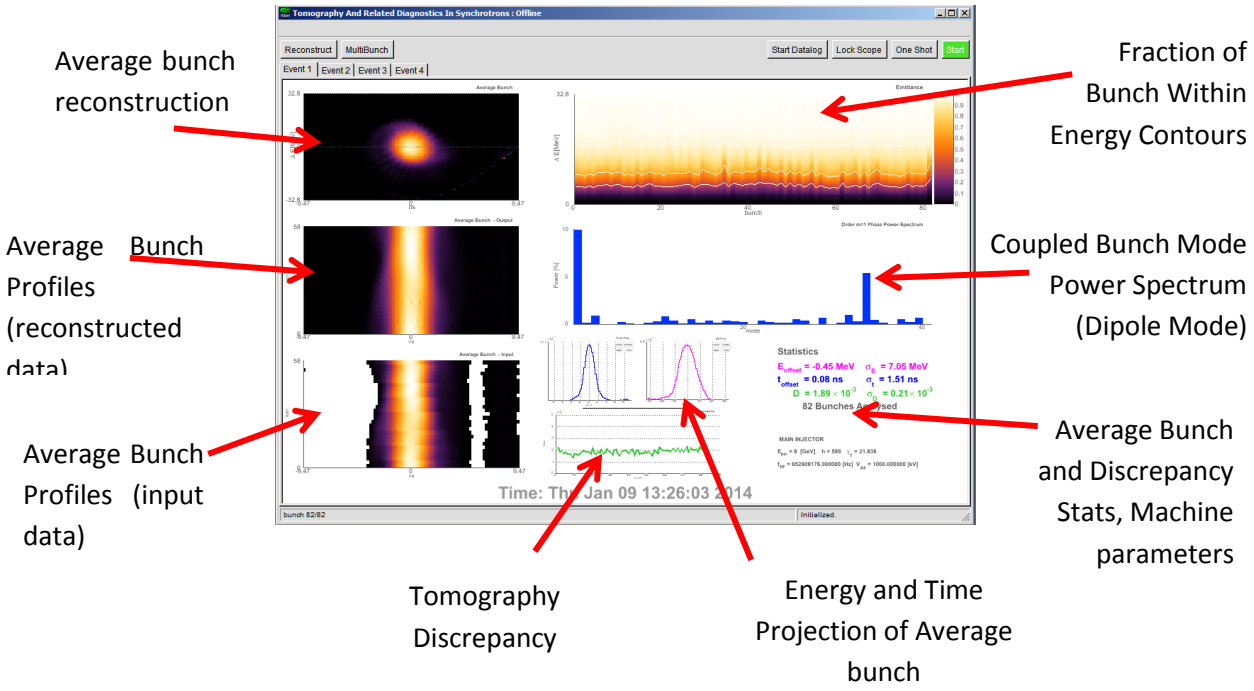


Figure 4: Example TARDIS results.

1.3 Continuous looping

Once looping TARDIS keeps repeating its cycle. There are 4 event tabs on the front page and these each hold different data sets. It is possible to save and re-analyse any of the current datasets. After the four tabs have data the programme cycles back to the first tab and that dataset is overwritten. If you want to change parameters, unlock the scope, look at some interesting data in a tab, stop continuous acquisition etc. click "Stop" and wait. Due to the different phases of the algorithm TARDIS does not stop instantly. When out of loop the red "Stop" button while turn green and display "Start."

1.4 Common Issues

The interaction between the CERN Root libraries, ACNET, and console operating system windows manager can sometimes be fragile, too many rapid button clicks, especially during window initialization, updates and resizing can cause crashes.

Many of the parameters can be set in multiple pop-up windows so be careful to avoid cross-talk.

1.4.1 Scope-Lock

Only one ACNET application can have control of the scope. If TARDIS has the scope the front page button changes color and text. Locks are based on an honor system and can be viewed in D60:LockPeeker.

1.4.2 Failed During Data Acquisition

Was there any beam? TARDIS does not know a priori if there was beam present, it makes this decision during the pre-processing phase. Whilst looping if there is no beam TARDIS will keep blindly asking the scope for new data and try and process it, this is the standard behavior and expected. If you know there was beam and TARDIS still failed during data acquisition then some setting up of the DAQ will be required. For well used set-ups that normally work there are two likely things to check:

- **Has the beam intensity significantly changed?** If so then the scope range, offsets and pre-processing triggers may need to be optimised to avoid signal clipping, improve signal to noise and enable TARDIS to find the requested batch from the raw data.
- **Beam Synchronization Offset Jitter.** The Trigger Box triggers the scope based on the beam synch offset signal, jitters have been observed somewhere in this system (probably in the trigger box hardware). It may be necessary to change the Trigger Box Beam Synchronization Offset so the scope records data for the requested batch over the correct time period.

1.4.2.1 Tomography Discrepancy is Too High

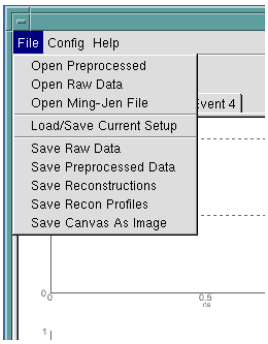
The discrepancy measures the difference between the raw data and the reconstruction. The mean value '*D*' is shown in the stats box, the graph shows D for each bunch. Values below 0.002 with little spread are desired, however the transition from good to bad is a judgment call. It is very possible to have

reasonable looking reconstructions with incorrect input parameters that give low(ish) D. Setting known parameters to correct values, such as RF voltage etc. is important and should help mitigate this.

2 Detailed Overview of GUI

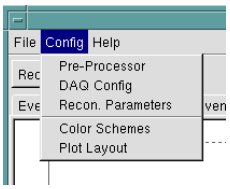
2.1 Menu bar: File

Opening files loads their data into the current active tab, each tab can have its own data set. However, at any one time only one set of machine and tomography parameters is used. If you have data for two different set-ups in two different tabsyou will have to change the machine accordingly.



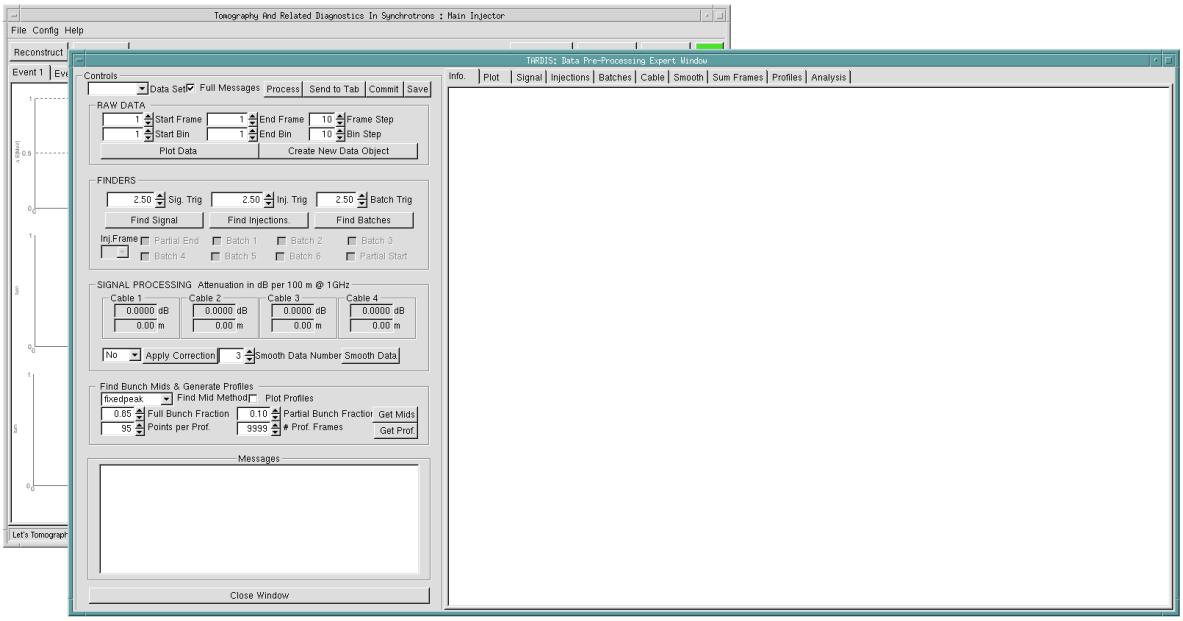
Open Preprocessed	Opens a unique type of binary file, *.prp used by TARDIS. The file has bunch profile data with a header ready to be reconstructed. These files do not have information on machine parameters.
Open Raw Data	These are also binary files unique to TARDIS, *.bin. They contain raw data not p-pre-processed for reconstruction. They also have a header containing information about the DAQ.
Open Ming-Jen file	Files saved in format from Ming Jen Yang's i86 (etc.) programme can be opened and used within TARDIS. Generally they have no file extension.
Load / Save Current Setup	Opens the "canned parameters GUI" pop-up window.
Save Raw Data	Saves the Raw Data associated with the current (active) tab
Save Preprocessed Data	Saves the pre-processed data associated with the current tab
Save Reconstructions	Saves each reconstruction associated with the current tab in a binary .tom format. These files can then be used in other applications etc.
Save Recon Profiles	Not Working (?)
Save Canvas as image	Save pdf image of the current Front Page.

2.2 Menu bar: Config



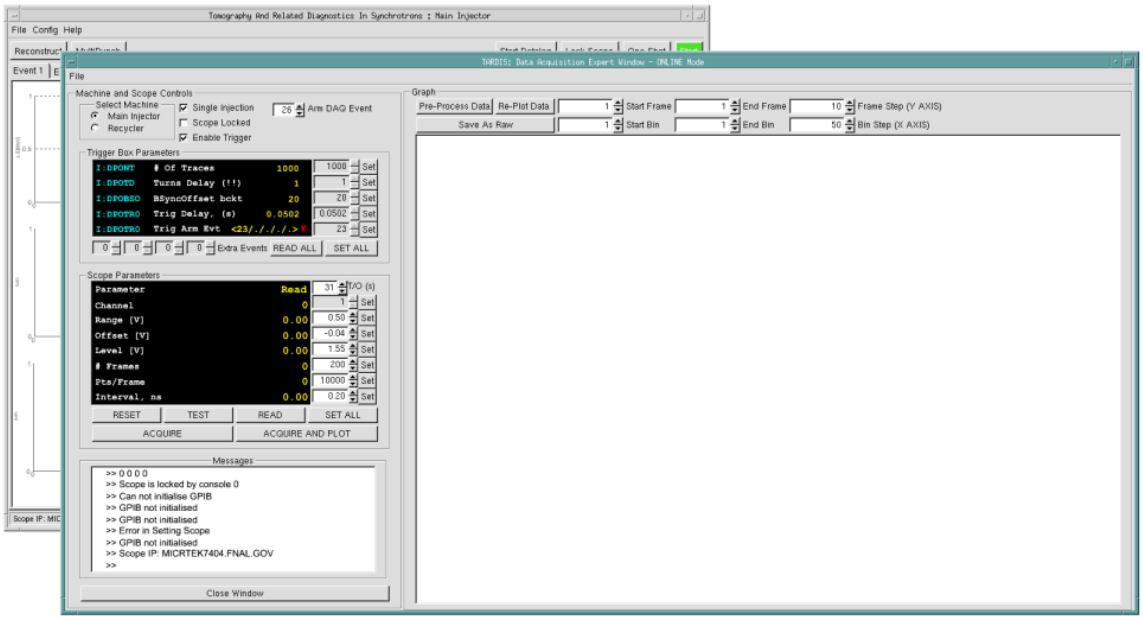
2.2.1 Pre-Processor

Opens the preprocessor GUI for setting up the pre-processing parameters



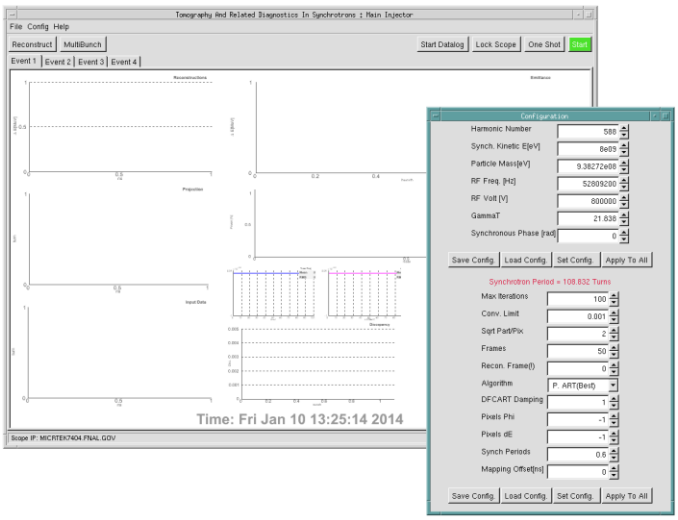
2.2.2 DAQ Config

Opens the DAG GUI where the scope and trigger box can be set-up



2.2.3 Recon Parameters

Opens a pop up window where machine and tomography parameters can be set. This is the only place in the GUI tomography parameters can be set.



2.2.3.1 Machine Config

Harmonic Number	Machine Harmonic Number
Synch. Kinetic E [eV]	Kinetic Energy of the synchronous particle
Particle Mass [eV]	Particle Rest mass
RF Freq [Hz]	The frequency of the RF system (NB during acceleration this changes).
RF Volt [V]	The voltage of the RF system (NB during acceleration this changes).
Gamma T	The Machine's transition gamma.
Synchronous Phase [rad]	Phase of synchronous particle (i.e. non zero during acceleration).

2.2.3.2 Tomography Config

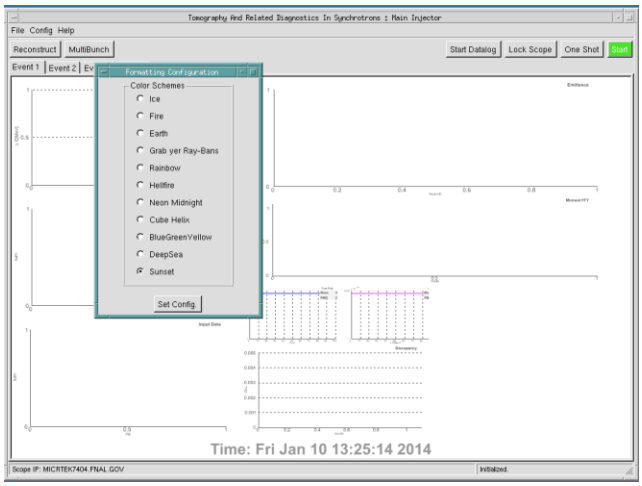
Max Iterations	The maximum number of iterations the tomography algorithm will perform (1 iteration uses all input profiles once).
Conv. Limit	Will exit if this limit (discrepancy) is reached before max iterations

Sqrt Part/Pix	The square root of the number of particles per pixel to use when constructing the maps
Frames	Number of Frames to use for tomography, only used if Synch Periods < 0
Recon Frame (!)	The frame to reconstruct, currently the first frame (labeled 0) is the only frame you can reconstruct using the GUI
Algorithm	Which tomography algorithm to use, **use P.ART (Best)**
DFCART Damping	The damping factor or the DFCART algorithm
Pixels Phi	The number of bins in the reconstruction grid for the time axis, setting -1 means the same as the points per profile from the pre-processed data will be used
Pixels dE	The number of bins in the reconstruction grid for the energy axis, setting -1 means the same as the points per profile from the pre-processed data will be used
Synch Periods	The number of synchrotron periods to use in the reconstruction (related to frames but this takes precedence). This should be at least 0.5, 0.6 seems to work best, but may need to be higher if much beam is towards the edge of the bucket and rotating slower.
Mapping Offsets (ns)	Offsets the maps relative to the profiles. This is used to make corrections when the bucket centre is not in the centre of the profile. When this is ***too great*** the oscillations do not fit the model and the tomography can give untrustworthy results.

These buttons save and load the config, separate for machine and tomography. Default values are loaded on startup from the files:

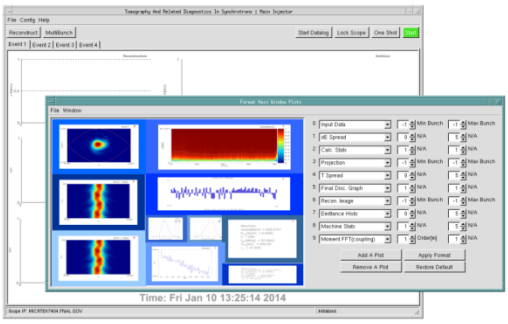
- defaultSettings.machine
- defaultSettings.tomography

2.2.4 Color Schemes



2.2.5 Plot Layout

TARDIS calculates many different things that can be displayed to the Front Page. The number of plots, which ones and their layout is configured from here. On initialization the current layout is displayed. Person layouts can be saved and loaded. The default view, loaded on startup can be changed in the in the file defaultViewFormat.view.



2.2.5.1 Description of Plot Type

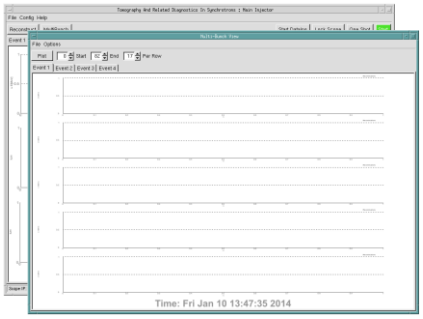
Recon Image	The reconstruction for a particular bunch. -1 gives the “average reconstruction” of all available bunches
Projection	The projection of the reconstruction image for the same frames used to create the reconstruction. -1 gives the average projection from all available bunches
Input data	The profiles generated by the pre-processor. -1 gives the average from all available bunches
Emittance histo.	The fraction of the reconstruction that is contained within increasing energy contours.
Machine stats	Display current machine stats
Recon stats	Display Tomography config for the current reconstruction
Calc stats	not used

Moment phase	The phase of each bunches n^{th} order moment, where 1 = dipole, 2 = quadruple etc. calculated from the reconstructions. The order is chosen next to the plot selector
Moment FFT coupling	The FFT of the moment phases and displayed the power spectrum in each available mode (coupling between bunches).
Final disc graph	Discrepancy of each bunch
De/e #sigma graph	Relative energy spread graph (Don't use)
T spread	The projection of the average bunch on the time axis, with mean and rms statistics
De spread	The projection of the average bunch on the energy axis, with mean and rms statistics.

2.3 Button Bar

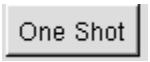
Reconstructs pre-processed data in the current tab

Opens Multi-Bunch View where all available reconstructions, input and output profiles can be viewed.



Enable / disable Data-logging

Lock or unlock the scope



Perform one complete cycle, from DAQ to reconstruction. Current TARDIS parameters are used

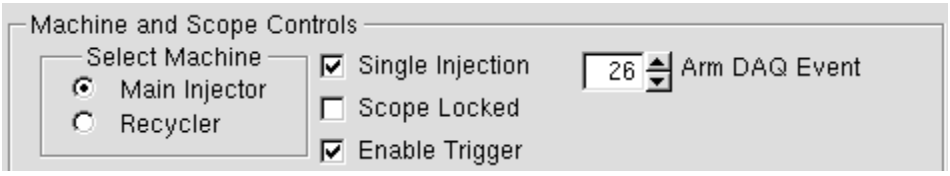
Start stop continuous looping

3 TARDIS Set-up

3.1 Setting up Data Acquisition

The Trigger Box (TB) and scope can be set through the canned parameters GUI or through the DAQ Config. Using the DAQ Config GUI gives more messages, read-backs and graphs of acquired data and is the preferred method for setting up DAQ. The GUI has entry boxes and read-backs for the TB and scope, a message box which displays information and a graph canvas for plotting the latest scope data transferred. There are also buttons to launch the pre-processing GUI and save the current raw data. On start up this window tries to lock the scope for the current machine setting, unless TARDIS already has the lock. If TARDIS has the scope lock then the Trigger Box Parameter entry Boxes are enabled, this is to help prevent someone changing values while someone else has locked the scope.[5] On locking a scope TARDIS always sends a clear and factory reset command.

3.1.1 Machine and Scope Controls



Select Machine Selects a trigger box / scope combination. [6] Changing machine will unlock the current scope and try and lock the new scope. **Be careful clicking too fast and many times when changing machine is one of the most common ways to crash the programme.**

Scope Lock Similar to the Front page lock / Free Scope

Enable Trigger The current MI TB can be enabled or disabled with this button. The "*new*" RR TB does not have this feature

Single Injection When checked TARDIS tries to unify the scope and TB parameters. This is planned to be unchecked in an as–yet-untested mode where multiple batch injection data can be recorded for the same event.

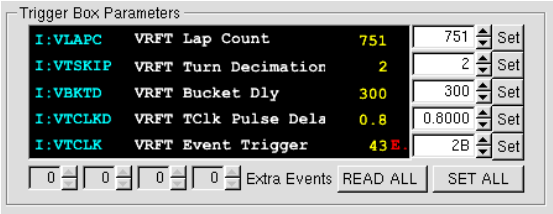
Arm DAQ event A decimal entry (for a hex TClock event). When in an acquisition cycle TARDIS resets the scope the next time this event occurs.

[5] NB there is always a "backdoor" into changing these parameters through ACNET

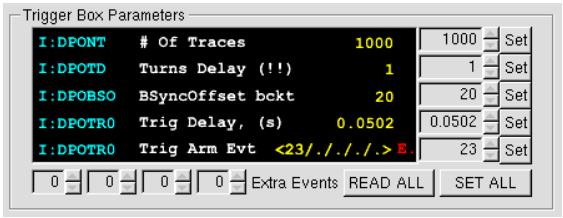
[6] The TB parameter names and scope IP are hardcoded. If they change TARDIS will need to be recompiled.

3.1.2 Trigger Box Parameters

Recycler



Main Injector

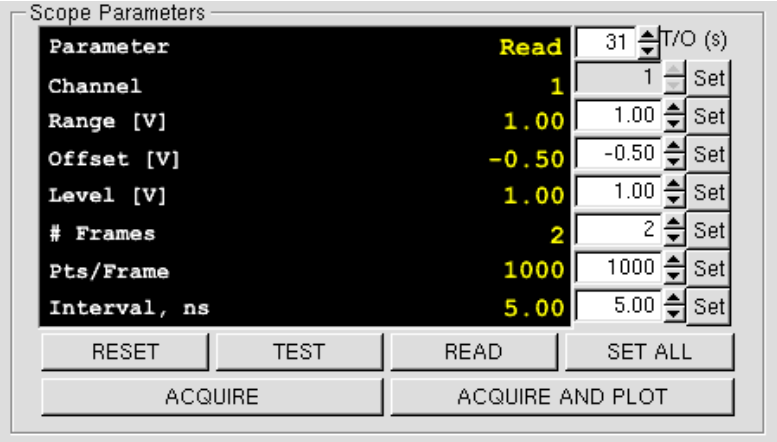


The text display shows the ACNET device name, description and current value, for the selected machine's TB. These are read when the DAQ GUI opens, and after a Set or READ ALL is pressed.

1. **Lap Count** 2. **# of Traces**	The number of "turns" or frames of data to take.
1. **Turn Decimation** 2. **Turns Delay (!!)**	This is the number of turns between trigger pulses. E.g. a value of 1 means there is a trigger every other turn.
1. **Bucket Dly** 2. **BSyncOffset**	The delay from the Beam Synch Offset marker. Sets the position in the ring each frame starts at
1. **TClk Pulse Delay** 2. **Trig. Delay**	The delay from the TClock event the first trigger is sent in seconds
1. **Event trigger** 2. **Trig Arm Event**	Events for which trigger signals will be sent. The MI TB can send multiple events, (25+) the RR just one. To avoid confusions set only one have one trigger event (0 is interpreted as a non-event).

Generally setting and reading TB should "*just work*". There is one known issue, the MI TB sometimes reset the lap Count to zero. To try and mitigate this when acquiring TARDIS checks the current value if a zero is read it resets the lap count to the last read value.

3.1.3 Scope Parameters



T/O (s) Time out in seconds. If a DAQ cycle takes longer than this then it will be aborted. This is not a robustly applied limit, for example, data transfer across the network is not timed. It will prevent long waits if an event that is not in the timeline is used as a trigger.

Channel Which Channel to read data of the scope. Currently hardcoded to channel 1 for both scopes

Range [V] The voltage range the scope will record signal

Offset [V] The minimum voltage the scope will record

Level [V] The voltage level of the trigger signal required to trigger the scope. Generally this should be left at 1V.

 # Frames The number of Frames i.e. triggers the scope will acquire. If this is greater than the number of trigger frames then the scope will acquire data off multiple events.

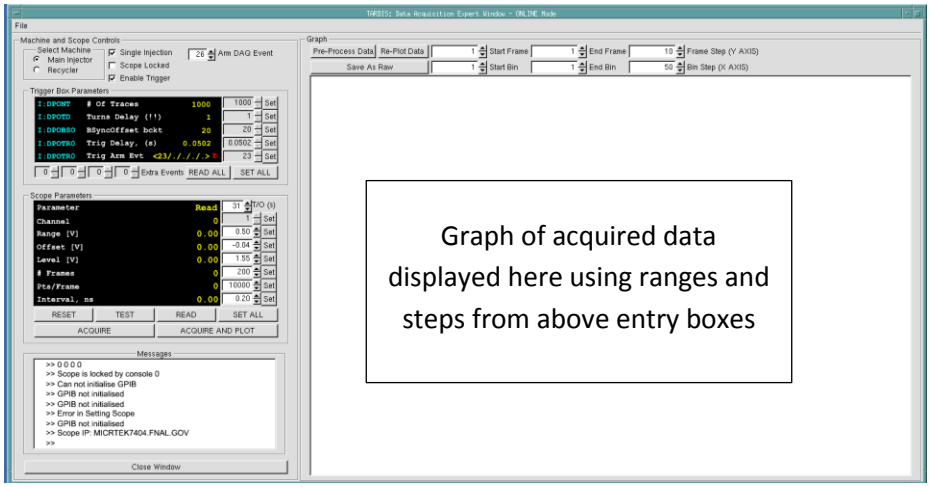
Pts / Frame The number of readings the scope will take per trigger

Interval ns The time interval between points

Due to the way the scope buffers data and the amount and type of memory it has etc., there are only certain combinations of # Frames, Pts/Frame and Interval it will accept. If you save a working set-up through the “canned parameters GUI” it is not clear that applying it with a fresh scope connection will necessarily work. For creating new default set-ups that will be used a lot it would be worth checking they work as expected.

3.1.3.1 Clipping

The pre-processing and reconstructions will not be reliable and accurate, if the data is clipped. Even a small amount of clipping at the high or low voltage level can cause problems and should be avoided. The signal from the RWCM changes due to beam intensity and effects attributed to heat and humidity in the tunnel. The scope digitizes the signal voltage to a single byte, or 0 to 255 "*scope counts*." A good range would be 30 scope counts being the minimum signal up to 200 scope counts for the maximum.



The bin and frame step size are used to reduce the number of plot points and speed up plotting.

3.2 Pre Processing GUI

Probably, from a user's point of view, the pre-processor is the most esoteric part of TARDIS, once the purpose, jargon and general idea behind many of the functions are understood it should become more accessible.

3.2.1 Pre Processing Philosophy[7]

The general purpose of the preprocessing is to take the raw data from the scope and divide it into individual bunch profiles suitable for reconstruction. Furthermore the bunch profiles should be centered on the bunch's bucket middles, the stable fixed point for the longitudinal motion of the bunch. A priori there is no convenient way of knowing where these middles are relative to the raw data acquired from the scope, therefore a number of algorithms, suitable for different cases are available to try and find these values. Once the bucket middles have been found it is relatively easy to generate profiles. Also included are some algorithms that can do some data processing of the signal to correct for cable

[7] For even more info on pre-processing see *Pre-Processing "under the hood"*

distortion effects and noise. These are generally only needed for short bunches at high energy. The pre-processing sequence is:

- Find signal start and end frame
- Find injection frames
- Find batch positions
- Select injection and batch
- Correct distortion (if desired)
- Smooth data (if desired
- Find bucket centers (bunch mids)
- Generate profiles

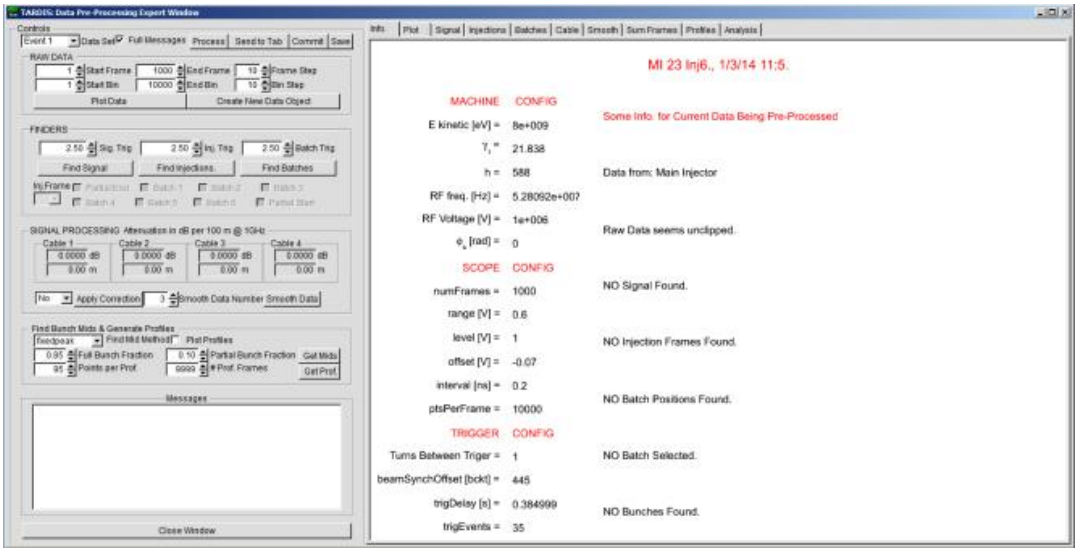
3.2.2 Pre-Processor Start-up

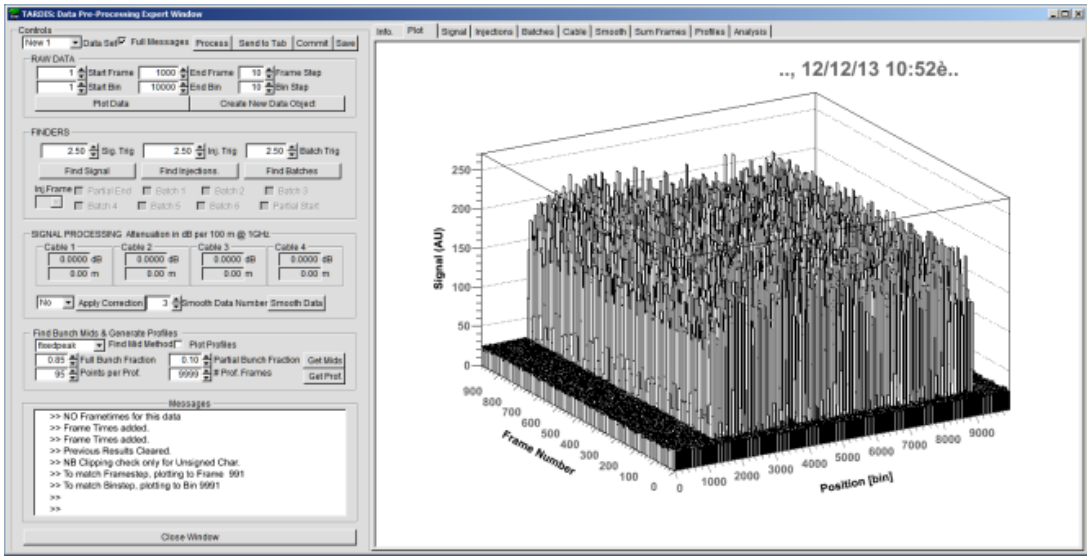
When the window is initialized the entry boxes are updated with the current parameters. There are controls and a message pad on the left and different tabs where graphs of results are displayed on the right. Data sets that the pre-processor knows about can be selected from the top left drop down box, changing data set clears all previous results***, in general performing any pre-processing task that is not the next in sequence will clear any results subsequent to the latest task performed***. There are a number of default data set names:[8]

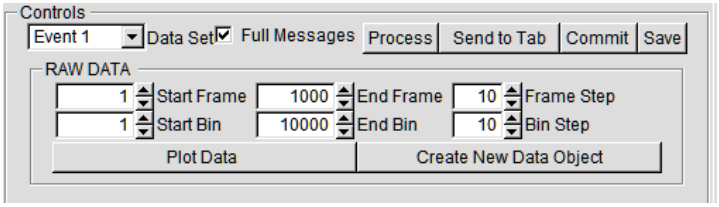
***Event 1* etc.**	Data in corresponding Front Page tab
Scope	The latest data acquired by the scope, only available if acquired through the DAQ Config window
***'New 1'* etc.**	New pre-processing data objects can be created from existing ones (for example to cut down a large set of data).

The Info Tab displays some information about the currently selected data set.

[8] When making a standalone application other names for data sets can be given.







Full Messages There was a time when the message pad was slow to update, so toggling displaying messages was useful. This has been fixed but the button remains. The displayed messages and their usefulness to those not involved in writing TARDIS have not been tested.

Process Tries to do a full pre-processing sequence using values in the entry boxes. It defaults to looking for batch 1 or a partial end of batch In

injection one unless a different value has been selected in the current sequence. (C.f. results after the current step generally get cleared.)

Send to Tab Sends a generated set of profiles to the active tab on the front page. (NB as a check it first re-runs the entire pre-processing sequence parameters)

Commit Commits the current GUI values to TARDIS, it does not save them for use on programme startup. (This can be achieved through canned set-ups.)

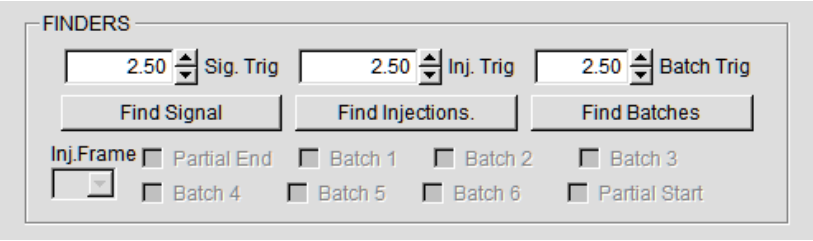
Save Saves the current raw data object to file.

Plot Data Plots the raw data over the specified range. It changes the end frame and bin to match the start and step. Entering a -1 is interpreted as use the maximum possible value. The bin and frame step size are used to reduce the number of plot points and speed up plotting.

Create New Data Object Creates a brand new raw data object within TARDIS, it uses the frame and bin ranges from the GUI, but the step-size is always 1. Entering a -1 is interpreted as use the maximum possible value. The entry boxes change their maximum limit depending on how many frames and bins are available. This useful to study parts of large data sets.

3.2.3 The First Three Finders

Each finder has a trigger value, which is just a number.[9] Generally in the range 1 to 5 is best.[10] Generally higher values are less sensitive and lower values will generate more false positives. Before any batches are found the "*Inj.Frame*" combo box and batch selection check boxes are disabled. Another thing to note is that the finders can find answers that are not correct leading to further mistakes in subsequent finders. It is advised to get a sensible result, not just any result, for each finder in turn.



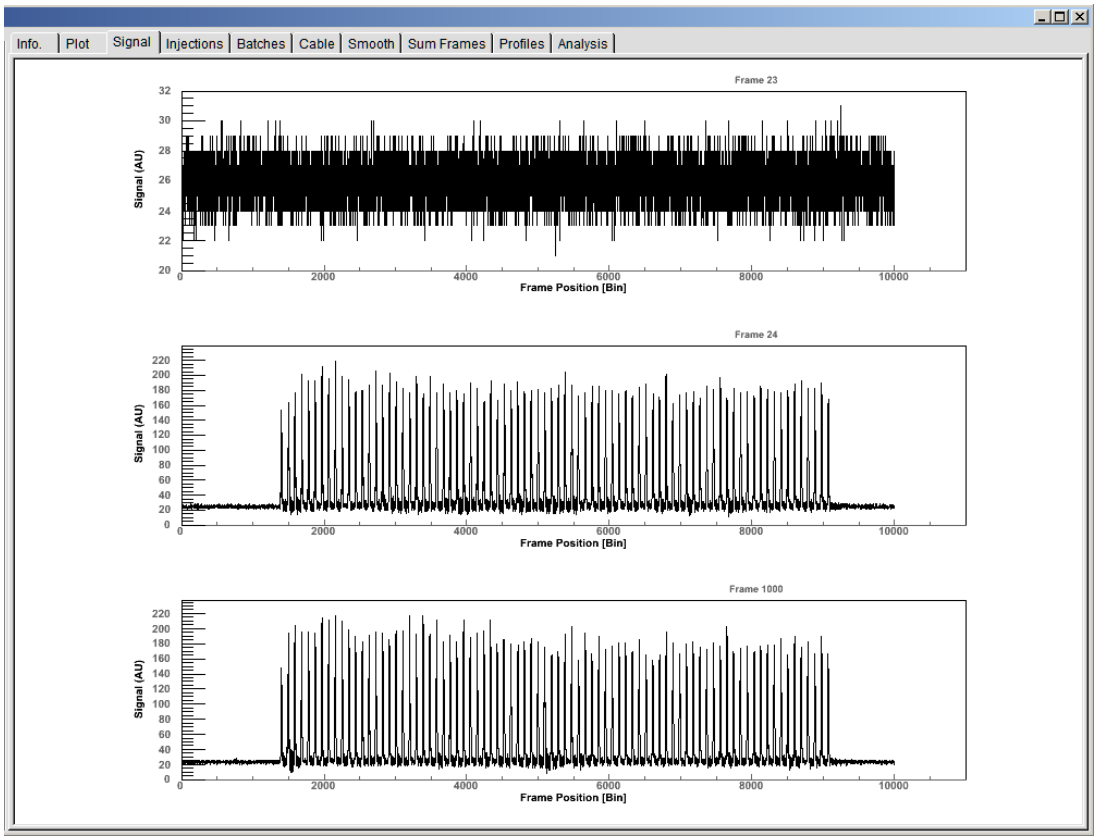
[9] This was deliberate, so different methods could be added in the source code and scaled appropriately.

[10] This was to allow different finding methods to be added to their class and to be scaled accordingly

3.2.3.1 Find Signal

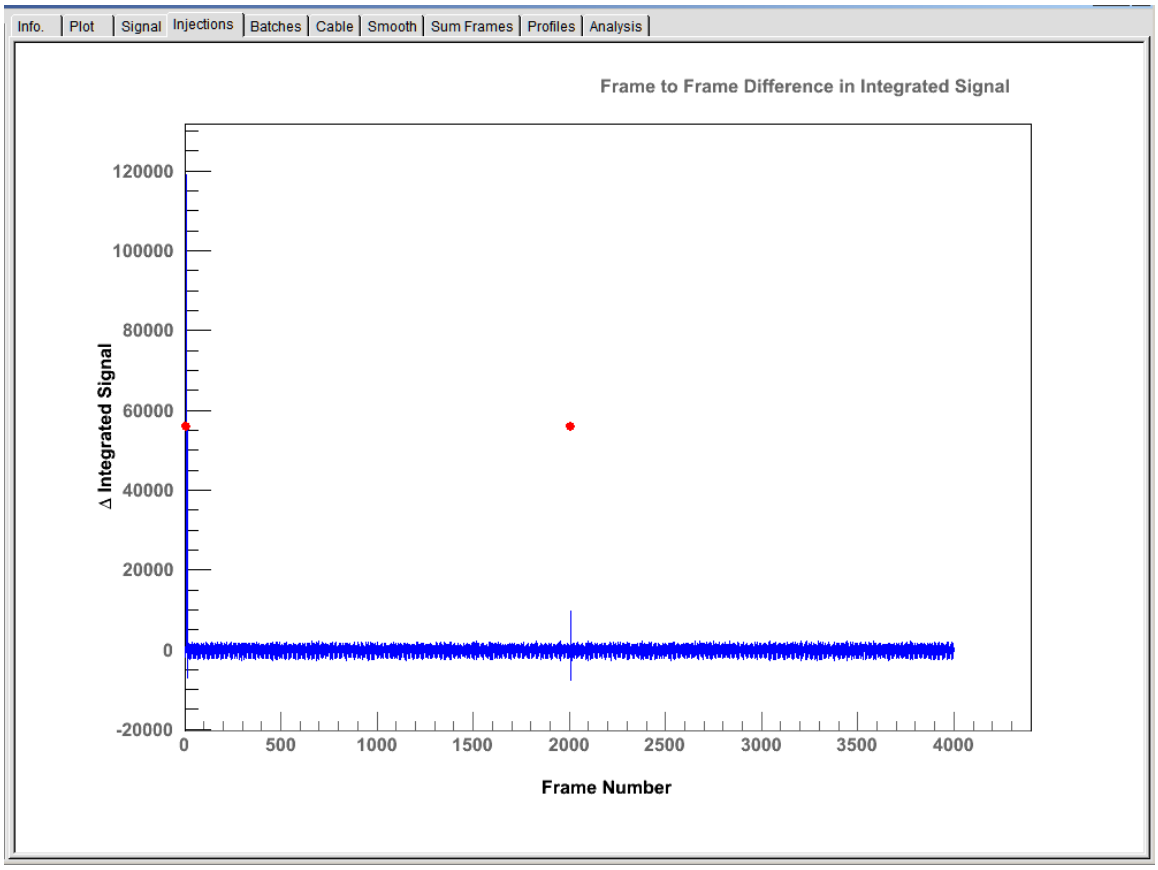
This looks for the first and last frames with signal. It should be simple to check if this finder has worked based on your knowledge of the data (which can be checked by plotting if you are unsure). After finding results the first and last frame with data are plotted to the signal tab, plus the frame before the first and after the last (if they exist). Simply checking the plots with what is expected is often enough to see if this finder has worked. The two main sensible results are:

- The first and last frame have signal, (two plots)
- The signal starts on frame “X” and the last frame has signal, for example:



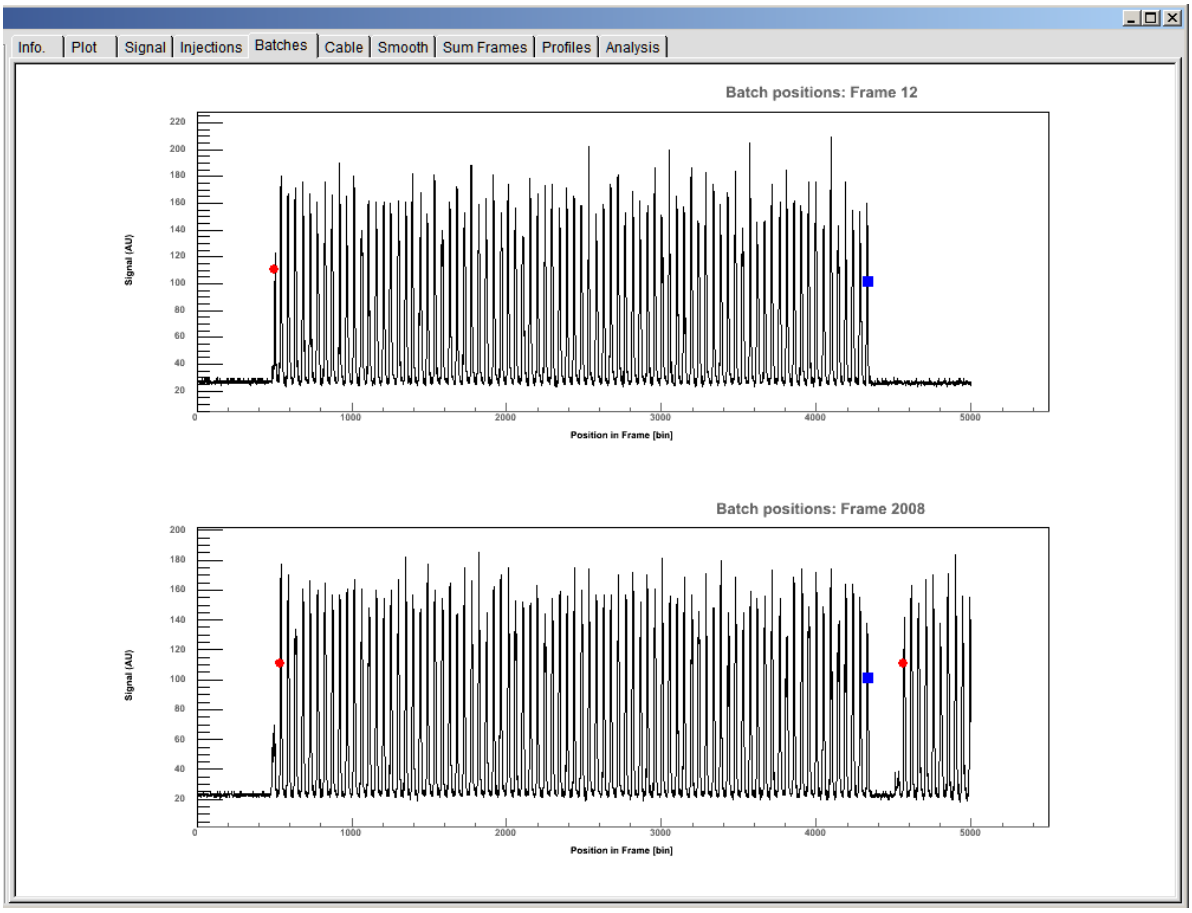
3.2.3.2 Find Injections

This finder tries to guess when beam has been injected. This is useful for a number of reasons, for example, looking at different batches at different injection points. The algorithm integrates the signal for each frames and then looks at the frame to frame difference (gradient), spikes clearly show when more beam is injected that can be found with a trigger. This algorithm has two main problems, (for speed and other reasons) the signal background is not subtracted and changes from frame to frame. The size of the injection spike is proportional to the relative amount of beam injected, so injection 6 is harder to find than injection 1. The plot can be used to check the number of injections is as expected.



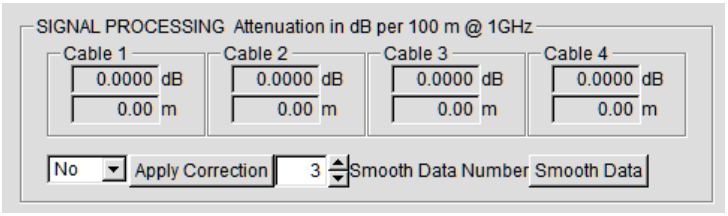
3.2.3.3 Find Batch Positions

This finder searches each injection frame for the star and end positions of batches by trying to find bunches. It then assumes that bunches that have a spacing of greater than a bucket width are a batch apart.[11] The finder performs some internal consistency checks, for example there are relationships between the number of start and positions, and an end can't come before a start etc. The results are plotted and the combo and check boxes become activated depending on the result. In the plots red points signify a start and blue an end.

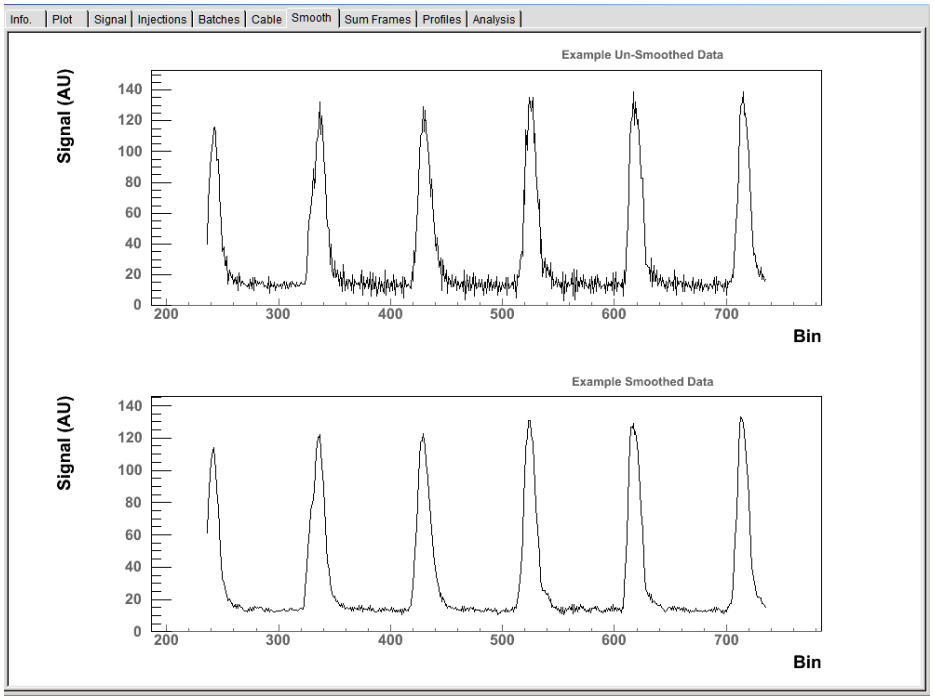


[11] This could cause problems for notched beams. A GUI way of implementing batch position finding was never implemented.

3.2.4 Signal Processing

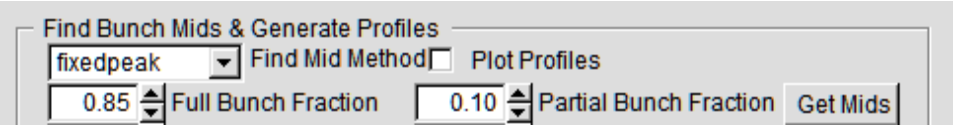


The ability for some simple signal processing has been included. Correction due to finite resistance distortion in coaxial cables can be included,[12] plus a simple moving average filter used to smooth the data. Here the default values are not correct, (the source code will need updating with correct values if necessary), and in any case user values can be added. Applying will plot graphs of example data before and after the correction.



[12] D.J. Scott Distortion in Resistive Wall Current Monitor Signal Transmission Lines Beams-doc-4436

3.2.5 Finding Bunch Mids



This finder tries to find the centre of the bucket relative to the signal for each bunch. First all the data for the selected injection and batch positions is summed (frame to frame). This "*sumFrames*" is used to find the bunch middles, the assumption being that summing enough frames will average out the motion. The finder works with a different number of methods, split into two types of two types:

Non - Local Keep the spacing between middles constant, based on the RF frequency

Local Does not restrict the bunch spacing by the RF frequency

Moment Calculate the moment (balance points) of the distribution and use that to estimate the centre

Peak Use the position of the peak intensity of each bunch.

When using non-local methods statistics are weighted by whether a bunch is full, partial or empty, depending on the fractional values given in the entry boxes. The peak methods were implemented after the moments methods because during testing it was noticed that there are often artefacts associated with the bunch that can shift the moment centre too much. Moment methods are still useful, especially when there is not a clear single peak to find, i.e. , during slip-stacking;

fixedpeak Finds the peak signal of each bunch, then finds the optimum values closest to the peaks whilst keeping their spacing constant

localpeak Finds the peak signal of each bunch

allbunches Non-local method that uses the moment of all non-empty bunches to find the optimum middles.

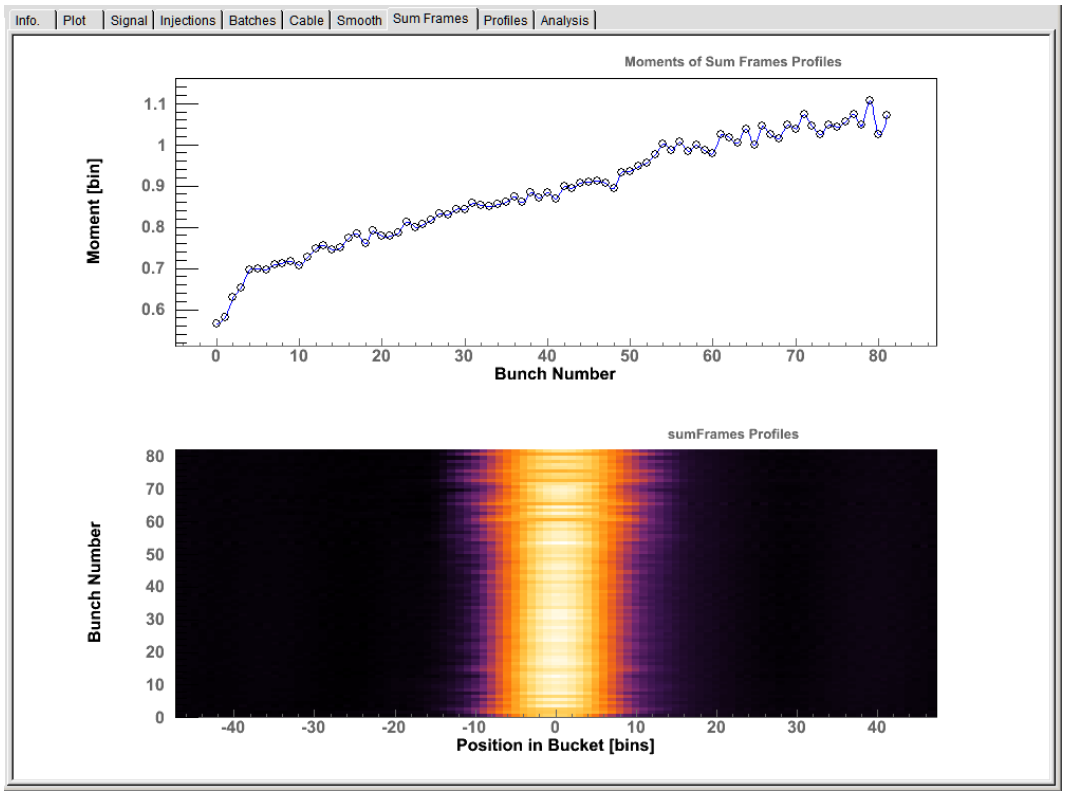
first Non-local method that uses the moment of the first non-empty bunch to be the first bunch mid.

firstfull Non-local method that uses the moment of the first full bunch to set the bunch mids.

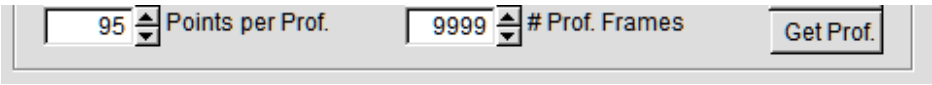
localcentre Local method that finds the uses the moment for each non-empty as that bunches middle.

After calculating the peaks the profiles of each summed frame bunch are plotted, plus the moment of each profile, in the example in figure two the firstfull method was chose, so the first full bunch has a

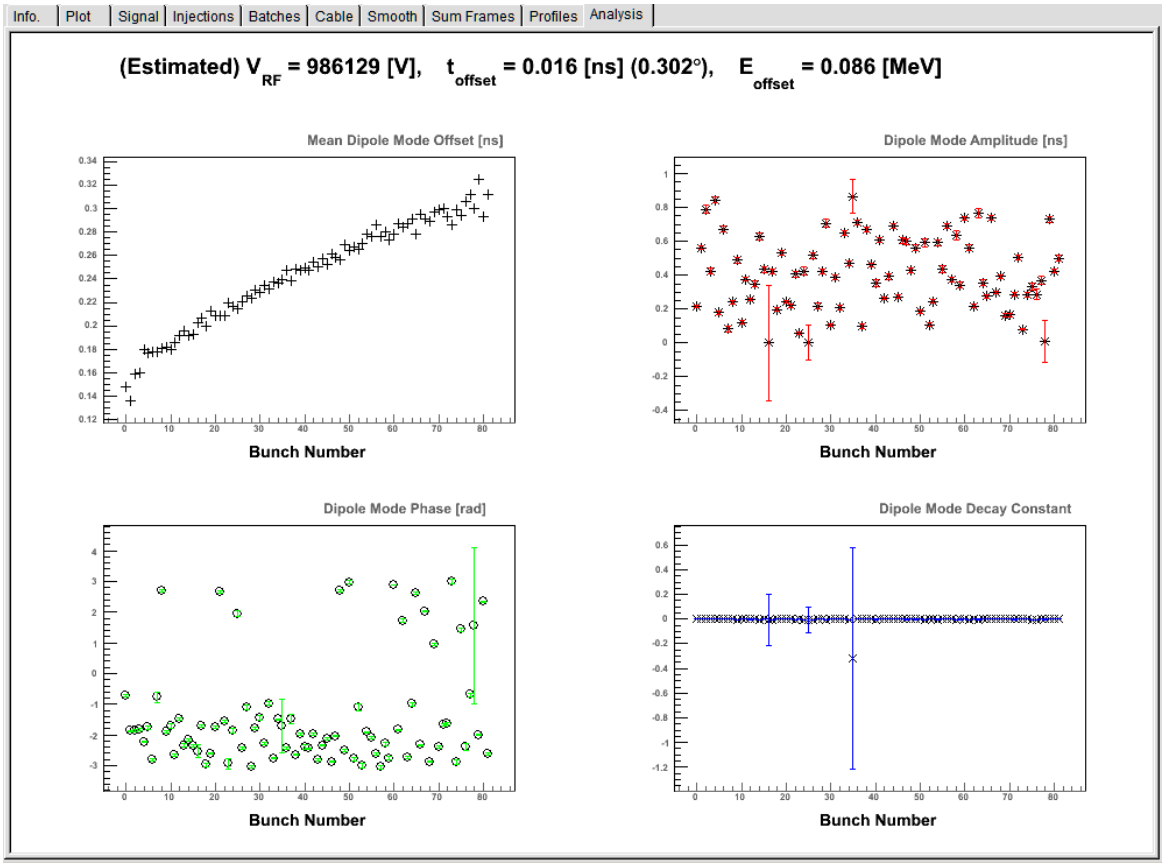
moment of zero. Interesting effects attributed to beam loading can be seen when finding the bunch mids. Also, during the tomography phase of TARDIS a ***Mapping Offset*** can be used to offset the profile centre.



3.2.6 Profiles



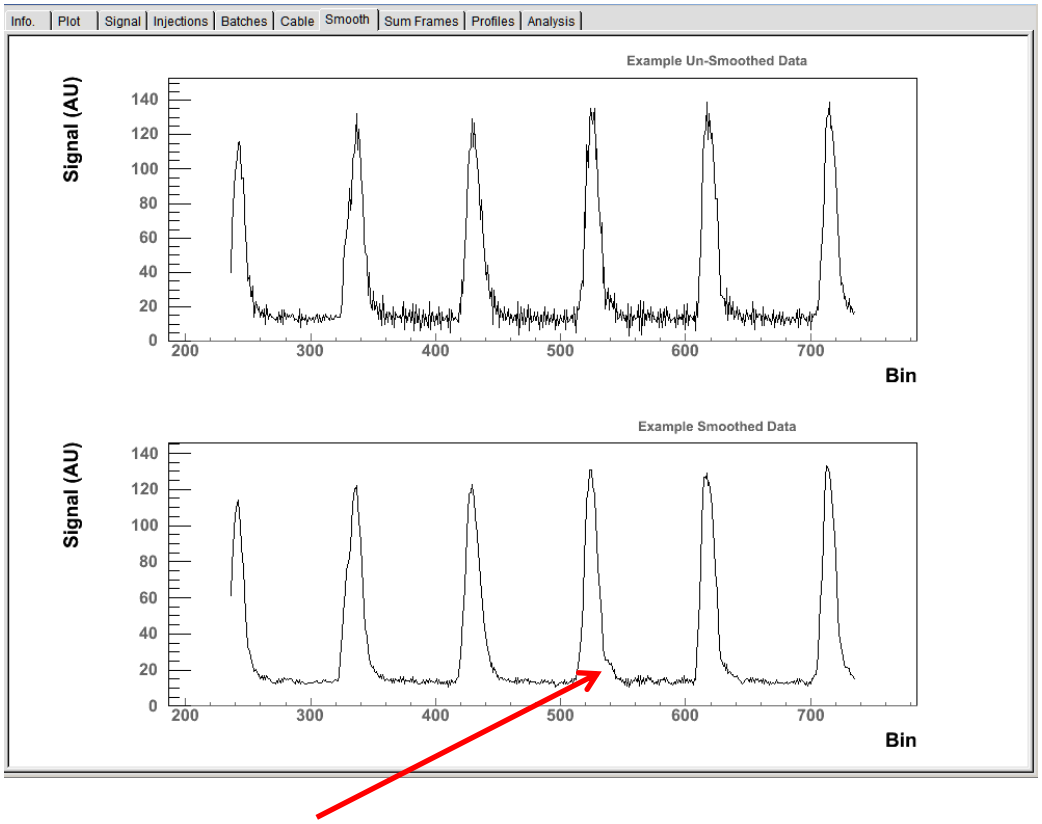
Once the bucket middles have been found profiles can be generated. The raw data is interpolated so any number of points per profile can be used. The number of profile frames can also be chosen, in cases of error the maximum number available is used, therefore for most standard use it is not necessary to change the value in the entry box.[13] Once profiles have been generated, and only in the pre-processing GUI some simple analysis is performed. The dipole moments for each bunch profile are found and then a decaying sinusoid is fitted. From the fit an estimate of the RF voltage, energy and time offset can be made. As they are based on only fitting a dipole mode and averaging over bunches these are only guess, but have been shown to be accurate under good working conditions.



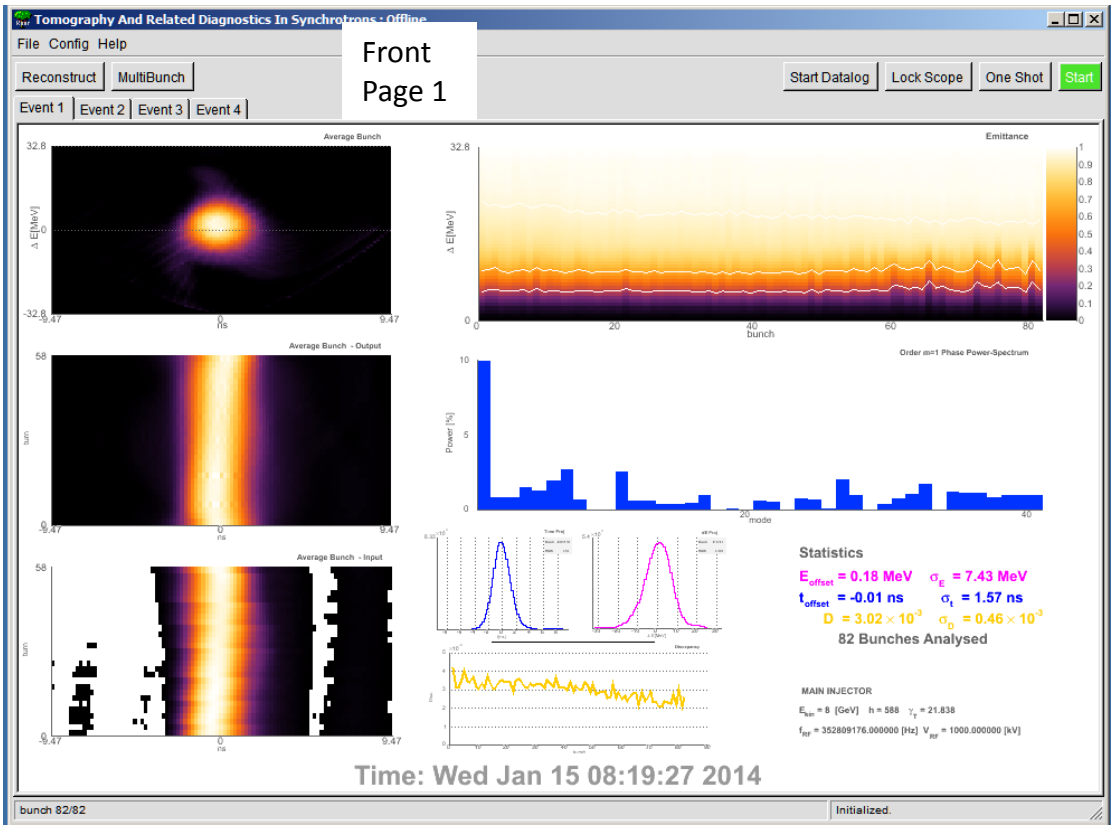
[13] If a large data set is being used with cable distortion correction then pre-processing can be significantly slower, that is the main purpose for this feature being available.

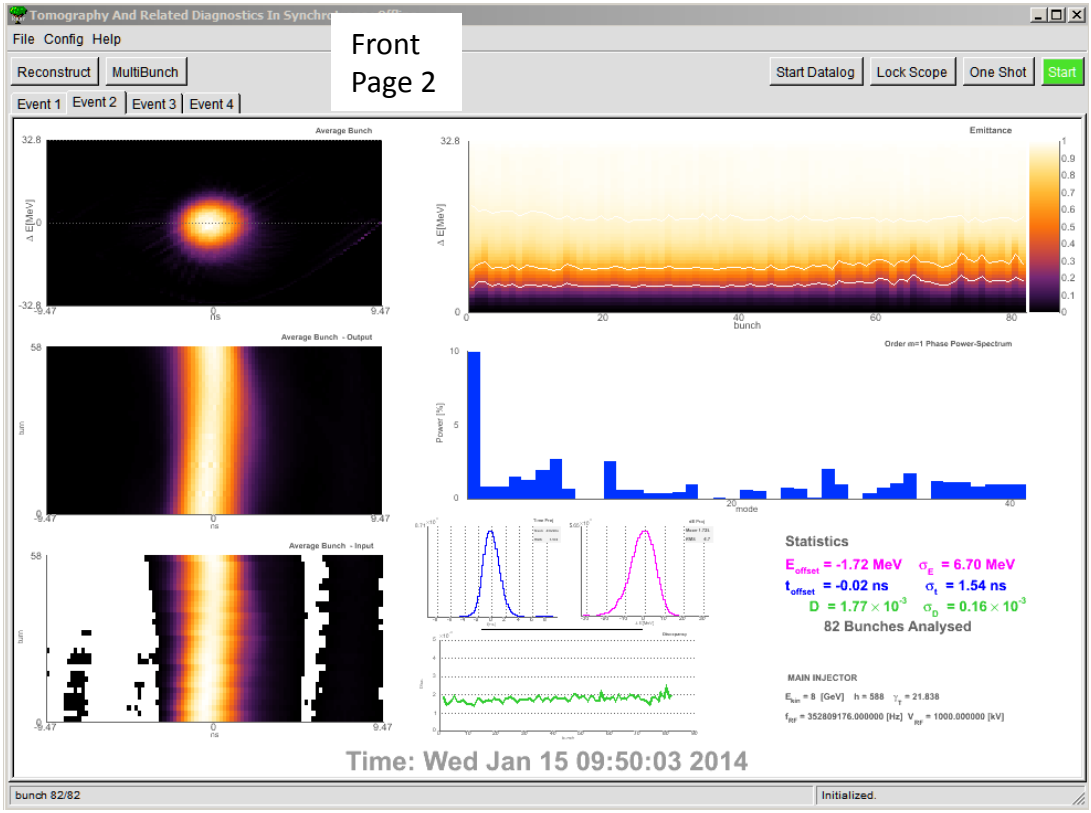
3.3 Example Effects of "bad" Pre-Processing

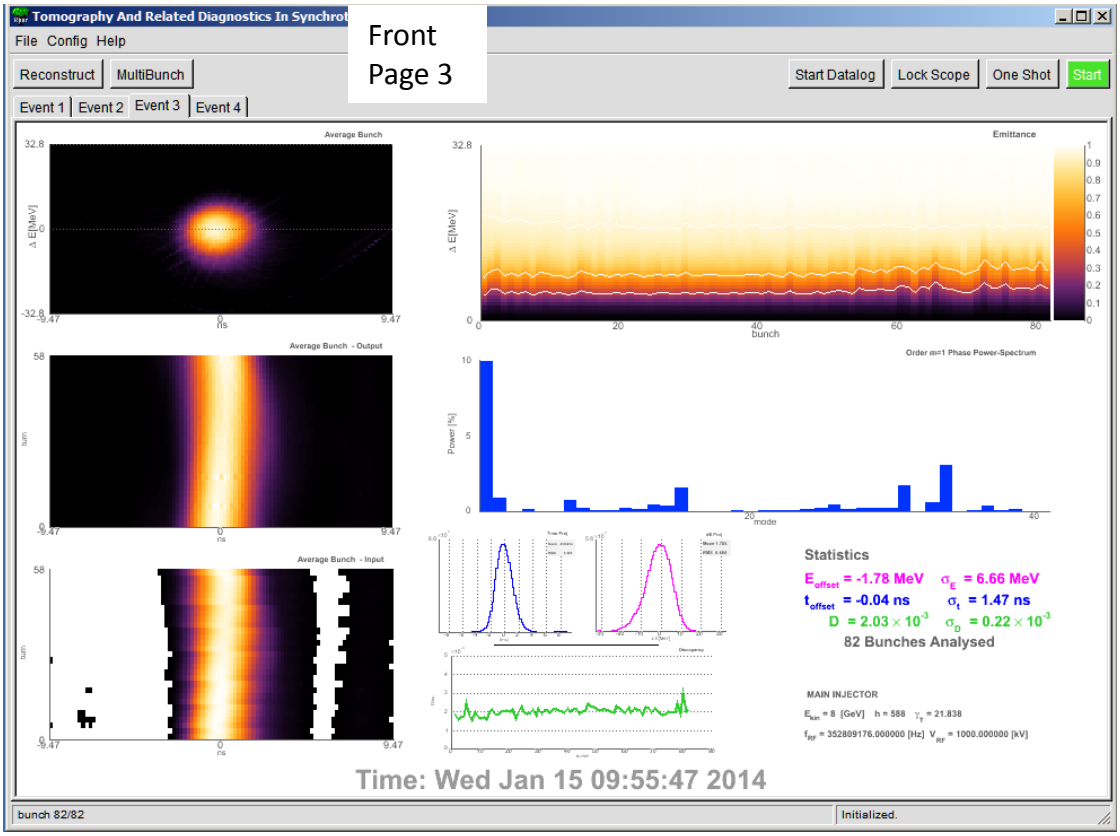
During DAQ the signal can acquire artefacts,[14] the following shows how these artefacts can affect the tomography and results, it also gives some ideas of things to check before gaining confidence in a reconstruction. The figure below shows an example of a shoulder artefact that is often seen. If the bucket centers are found using a moment method this shoulder shifts the center to the right. There are two problems for the tomography algorithm, it has to reconstruct an artefact that does not fit the model, and the bunch is now oscillating around a different centre. Front Page 1 shows the effect of this, the reconstruction acquires a 3-fold symmetric tail, the recon profiles do not bend like the input data and they have a kink at the start and end. Also the discrepancy is higher than the normal acceptable levels. Front Page 2 and 3 show better reconstructions using a mapping offset of -0.3 ns and a peak method in pre-processing, these two techniques give more consistent results with much lower discrepancy, and so should be trusted more.



[14] For example impedance mismatches in the cabling


Tomography And Related Diagnostics In Synchrotrons - Offline
File Config Help
Reconstruct
MultiBunch
Start Datalog
Lock Scope
One Shot
Start
Event 1
Event 2
Event 3
Event 4
Front Page 1
Average Bunch
Emittance
Average Bunch - Output
Order m=1 Phase Power-Spectrum
Average Bunch - Input
Statistics
E_offset = 0.18 MeV σ_E = 7.43 MeV
t_offset = -0.01 ns σ_t = 1.57 ns
D = 3.02 × 10^-3 σ_D = 0.46 × 10^-3
82 Bunches Analysed
MAIN INJECTOR
Time: Wed Jan 15 08:19:27 2014
bunch 82/82
Initialized.


File Config Help
Reconstruct
MultiBunch
Start Datalog
Lock Scope
One Shot
Start
Event 1
Event 2
Event 3
Event 4
Front Page 2
Average Bunch
Emittance
Average Bunch - Output
Order m=1 Phase Power-Spectrum
Average Bunch - Input
Statistics
E_offset = -1.72 MeV σ_E = 6.70 MeV
t_offset = -0.02 ns σ_t = 1.54 ns
D = 1.77 × 10^-3 σ_D = 0.16 × 10^-3
82 Bunches Analysed
MAIN INJECTOR
Time: Wed Jan 15 09:50:03 2014
bunch 82/82
Initialized.



Front Page 3

Effect of having a bunch mid error using moments

	Moments Method	**Mapping Offset**	**Peaks Method**
ΔE offset (Mev)	0.18	-1.72	-1.78
ΔE spread (Mev)	7.43	6.70	6.66
Δt offset (ns)	-0.01	-0.02	-0.04
Δt spread (ns)	1.57	1.54	1.47
Mean D	3.02E-3	1.77E-3	2.03E-3
D spread	0.46E-3	0.16E-3	0.22E-3

4 Data Logging

With the Start Datalog button clicked TARDIS calculates the energy and time projection of the average bunch and updates ACNET devices with their offset and rms, it also updates the discrepancy. There are two data logging modes, *Normal* and *Slip-Stack* set through the canned parameters GUI. Normal mode

takes projections of the entire reconstruction area. In *slip-stack* the reconstruction area is cut along the ΔE=0 line and projections are taken on Hi and Low energy beam.

I:TAREOF	**Energy Offset**	**I:TAREOH**	**Energy Offset Hi cut**
I:TARESP	**Energy Spread**	**I:TAREOL**	**Energy Offset Lo cut**
I:TARTOF	**Time Offset**	**I:TARESH**	**Energy Spread Hi cut**
I:TARTSP	**Time Spread**	**I:TARESL**	**Energy Spread Lo cut**
I:TARDIS	**Mean Discrepancy**	**I:TARTOH**	**Time Offset Hi cut**
I:TARDSP	**Discrepancy Spread**	**I:TARTOL**	**Time Offset Lo cut**
I:TAREOH	**Energy Offset Hi cut**	**I:TARTSH**	**Time Spread Hi cut**
I:TAREOL	**Energy Offset Lo cut**	**I:TARTSL**	**Time Spread Lo cut**
I:TARDIS	**Mean Discrepancy**	**I:TARDSP**	**Discrepancy Spread**
I:TARAUX	**Current State of TARDIS**		

5 Conclusion

This is the end of the user manual. The next section is a more technical explanation of the pre-processing, referring to the source code and also gives an idea of the algorithms used.

TARDIS was built up from virtually no knowledge coding, c++, using the *M-V-C paradigm*, and also before a fuller understanding of Root's capabilities was gained (i.e. signals and slots, Root's Object management system, etc.). This means that the source code could be cleaned up, a lot. Even so, apart from in a few key areas it should be reasonable to add in new functionality and change algorithms without breaking too many other parts. (Hopefully.)

For those interested in "*Tomogrpahy - Under the Hood*" see Nick Evans' PhD and the source code ☺.

6 Pre-Processing "*Under the hood*"

This is a rough outline, it gives some ideas on how and why the code is the way it is. It was planned to be a separate note for people interested in the code.

Basic Terms

There are a number of structs that hold essential data for the pre-processing. The trigParam and scopeParam, hold data about the scope and trigger box set-up during acquisition. The machine parameters are held in a Machine_Config struct and the pre-processing parameters and options are held in a ppStruct. For convenience a DAQStruct contains a version number, title, time , scopeParam and trigParam. The relevant numbers from the Machine_Config are the RF voltage and frequency and the harmonic number of the machine (588 at 53 MHz), these are defined in the main part of TARDIS.

Trigger Box

There are 5 main parameters to consider:

Number of Traces	numTraces	The number of trigger events.
Turns Between Trigger	turnsDelay	The number of machine turns **between** each trigger event. I.e. a value of 1 would result in a trigger every other turn. A value of 0 would give a trigger every turn, however the fastest the scope can operate is every other turn.[footnote]
Beam Synch Offset (buckets)	beamSynchOffset	The offset, in buckets, from the A0 marker defining the injection position of the beam
Delay from Event (seconds)	trigDelay	The delay in seconds from the event being broadcast and the first trigger.
Event for Trigger	trigEvents	The event(s) for which triggers will be generated, i.e. $23 etc.

These can be set in the ScopeGUI or through ACNET <I34> subpage 12, etc.. The scope parameters can also be controlled in this window, only 3 are required for pre-processing and tomography, although the scopeParam struct contains other information about the scope set-up:

Number of Frames	numFrames	The number of Frames of Data the scope acquired.
Points per Frame	ptsPerFrame	The number of data points in a single frame.
Time scale	interval	The time, in nanoseconds, between data points

It is worth noting that numFrames and numTraces need not be the same. For example, the trigger box could be set up to give 100 traces on every injection event and the scope could be set up to record 600 frames allowing 100 frames of data of data at each injection to be recorded, then analysis of each injection for a single MI cycle could be done.

The tomography algorithms do not rely on the vertical scale of the data and so the data from the scope can be used in its raw, unsigned char type format, known as scope counts. These range from 0 to 255. It is very important that the signal is not clipped at the high or the low end, so the scope voltage range and offset must be set well to ensure this does not happen. Currently there is no auto-ranging module, although one is envisioned, and a few exploratory functions tried. A few frames of typical data is shown in Figure 5. The red trace shows the signal before beam is injected and the blue and the green show the beam signal. There are a couple of features that are typical to most data sets that are worth highlighting. The signal baseline changes from frame to frame. High frequency noise due to signal distortion in the detector and cables can be seen. In general F_i will denote the i^{th} frame of data, and B_j will denote the j^{th} bin of a particular frame, the j^{th} bin of the i^{th} frame could be represented by $F_{i,j}$ or $B_{j,i}$.

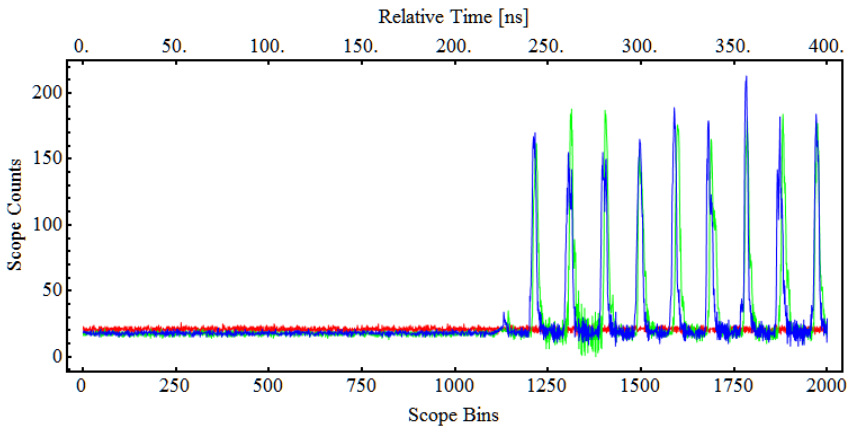


Figure 5: example raw data from scope.

Pre Processing

The data acquisition and pre processing is implemented through various classes that perform the different tasks. There are two main controller classes that the TARDIS master controller has access to, the scopeController and the ppController. The scopeController also has access to the ppController.[15] Both controllers have a GUI class from which various parameters can be set and different stages of the pre-processing can be performed in isolation for tuning. In addition the scopeController creates to objects that control the two pieces of hardware, the scope and the trigger box. Every time new data is acquired by the scopeController it passes it to the ppController which then creates a new ppDataObject. The ppDataObject knows how to calculate the various manipulations of the raw data, in general it does

[15] And to the master, for one function. To decrease dependency this should be re-written.

that by creating *finder* classes that can perform individual operations. Encapsulating in this way allows for different methods to be added to the various finders in a manageable way. A schematic block diagram of the classes and their dependencies is shown in Figure 6.

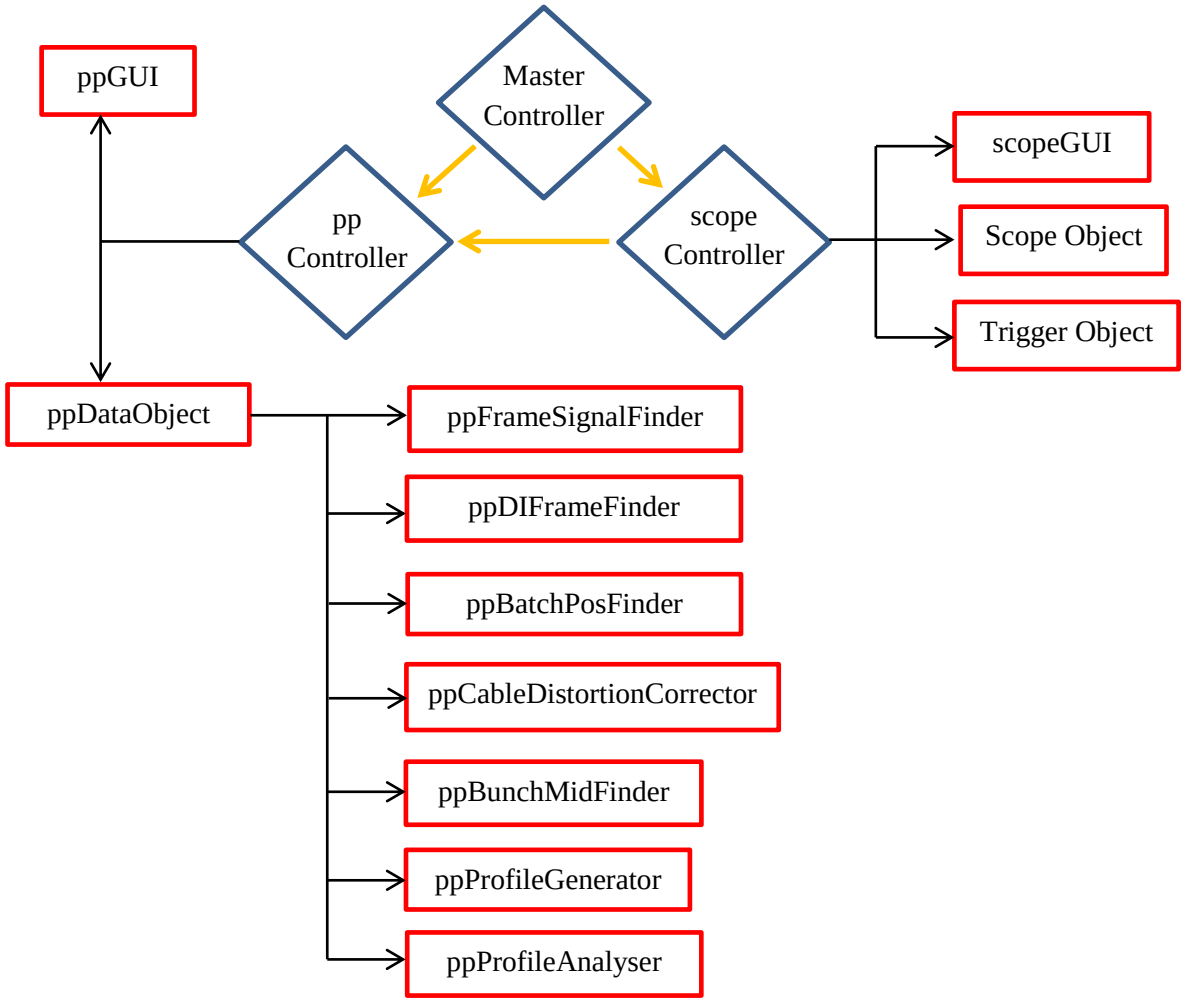


Figure 6: Block Diagram of Data Acquisition and Pre-Processing classes.

The three main finder classes that find signal, injections and batch positions are described below. These procedures are mostly automated but each rely on one user defined trigger number, T_{sig}, T_{inj} and T_{batch}, these can be set in the ppGUI. As a *rule-of-thumb* the trigger values can be set at nominal values between 2 and 3, although low signal to noise ratios makes the finding process more difficult.

ppFrameSignalFinder

In order to reconstruct the first turn the beam is injected on it is advisable to record a few turns with no beam. Also, timing jitters often mean that the delay from the trigger event is not exactly the same each

time. The frame signal finder checks which frames in the raw data have signal, so that only frames with beam are used in the further analysis, it is the simplest of the finder classes. For a particular frame the algorithm checks if the ratio of maximum value to mean value is greater than the trigger value the with the trigger, starting with the first frame and then incrementing until the first frame that passes the test is found. Once the first frame of data is found the process is repeated starting with the final frame of data, again stopping with the first frame that passes. This procedure is fairly robust unless there is a low signal to noise ratio.

ppFrameSignalFinder pseudo-code

$i = 1$

$$\text{While } \left[\frac{\text{Max}[F_i]}{\text{Mean}[F_i]} < T_{\text{sig}}\text{, increment } i\right]$$

First frame with signal $= i$

$i = \text{Number of Frames}$

$$\text{While } \left[\frac{\text{Max}[F_i]}{\text{Mean}[F_i]} < T_{\text{sig}}\text{, decrement } i\right]$$

Last frame with signal $= i$

ppDIFrameFinder

This class looks for frames in which new beam has been injected in to the machine. The DI represents ΔI or, Δcurrent. It was planned to have one injection set-up for TARDIS that will record all six injections in the machine and then a particular batch, or batches, can be selected from this data set. To do this requires selecting the frames where the new batches are injected. First the total signal for each frame, FS, is found by summing the bin values:

$$\text{FS}_i = \sum_j F_{i,j}$$

Typically when beam is injected there is a step change in integrated signal, shown in Figure 7. The changing background level can distort the step change so plotting the absolute value of the differences (gradient) of successive frame sums, $\Delta\text{FS}_i = |\text{FS}_{i+1} - F_i|$ gives a cleaner signal to test, shown in Figure 8. (Of course the background for each frame could be subtracted, however accurately calculating that for each frame was deemed is relatively time costly and not necessary in general. The preferred method for background subtraction is histogram the background (not signal points) and fit a Gaussian, taking the mean as the background, i.e. non- integer value.

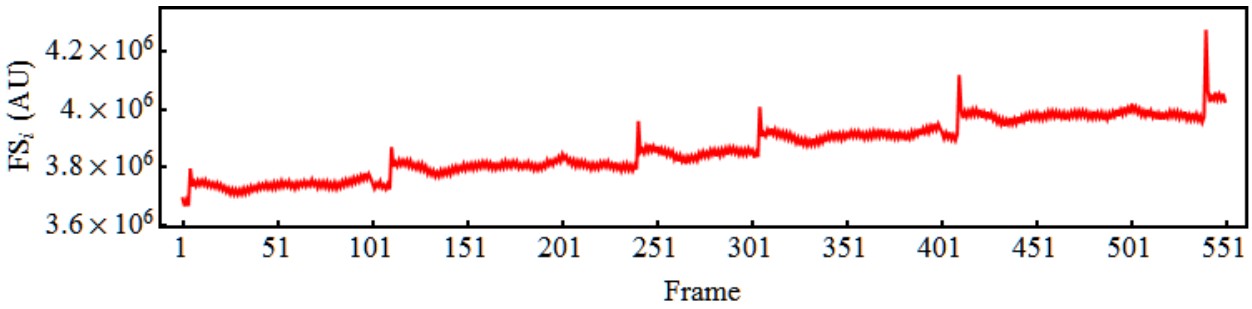


Figure 7: Example signal from summing each frame.

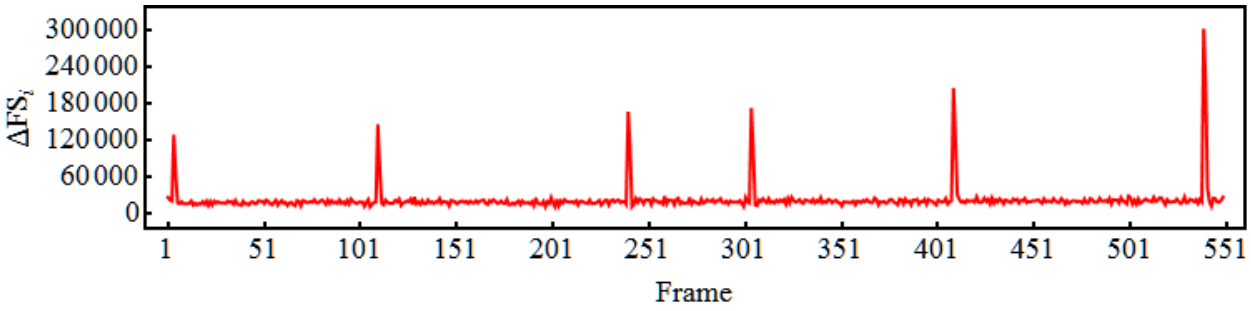


Figure 8: ΔFS_i for the data in Figure 7.

For the test to decide if a frame is an injection frame the trigger T_{inj} is weighted by the rms of the list of ΔFS_i values:

$$\mathrm{Test_{inj}} = T_{\mathrm{inj}} \times \sqrt{\frac{1}{n}\left(\Delta FS_1{}^2 + \Delta FS_2{}^2 + \ldots + \Delta FS_n{}^2\right)}$$

and a frame is selected as an injection frame if:

For $i = 1$ to (Number of Frames $- 1$)

If ($\Delta FS_i > \mathrm{Test_{inj}} \wedge \Delta FS_{i+1} > \mathrm{Test_{inj}}$, i is an injection frame)

The simple way to see if the algorithm has worked is to plot the frames that pass the test with the preceding one, as shown in Figure 9.

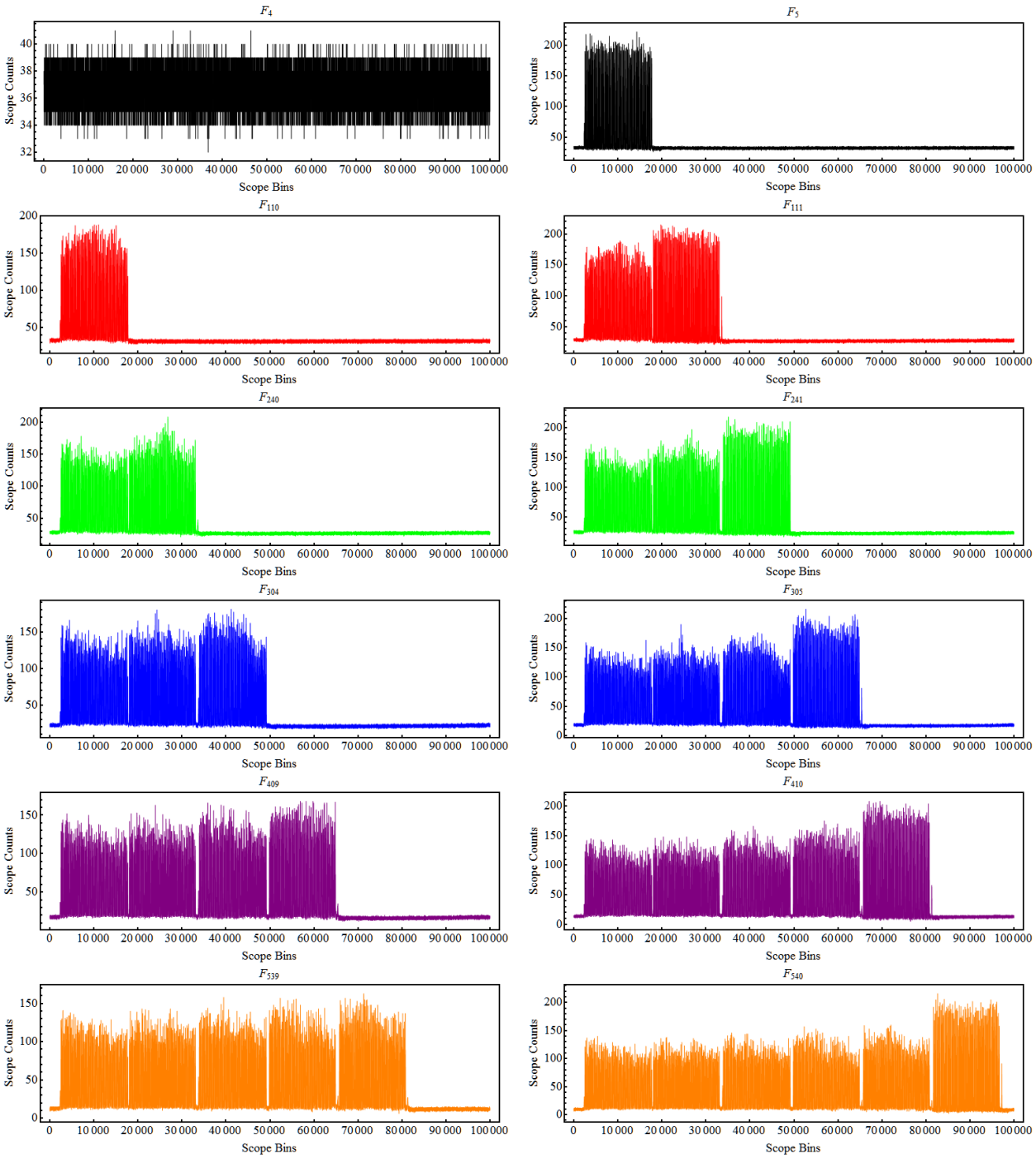


Figure 9: Example data showing a successful finding of injection frames.

ppBatchPosFinder

This class looks for the start and end positions of batches in the data. It uses another user defined trigger T_{batch} that is scaled by the range of the data in the frame to give $\mathrm{Test}_{\mathrm{batch}}$:

$$\mathrm{Test}_{\mathrm{batch}} = \mathrm{Min}[F_i] + \frac{\mathrm{Max}[F_i] - \mathrm{Min}[F_i]}{T_{\mathrm{inj}}}$$

Only the injection frames found with the ppDIFrameFinder are tested. The algorithm steps through each $F_{i,j}$ looking for adjacent bins that are below then above the trigger, indicating that a bunch edge has been found

For $j = 1$ to Number of Points per Frame

If ($F_{i,j} < \mathrm{Test}_{\mathrm{batch}} \wedge F_{i,j+1} < \mathrm{Test}_{\mathrm{batch}}$, j is the position of a bunch; save)

The saved js are then compared to one another, if they are more than twice a bucket length apart it is assumed that is a gap between batches.

For $k = 1$ to Number of saved j

If ($j_{k+1} - j_k > 2 \times$ bucket length, j_{k+1} is the position of the start of a batch)

To find the positions of the end of batches the algorithm is run starting at the last frame of the bin and working backwards. Often there are small partial bunches at the start and end of a batch that have been clipped due to mismatches in kicker firings (or some other effect) therefore the found batch start and end positions are shifted by 2.5 bucket lengths to account for this (whilst making sure to not go below the first position or above the number of bins in the frame). This algorithm tends to be the one most sensitive to the input trigger, this is because there can be a wide variation in the peak signal for a bunch, as shown in Figure 11.

Data Structure encoding.

Once the number of batch start and end positions has been found the structure of the raw data is encoded in a two digit number. This is to simplify handling a user to request to reconstruct a specific batch. For any given set of data (that is not empty) there are only a finite number of patterns the batch structure can take. These can be constructed from 4 primitives one of which is a special case, shown in Figure 10. The first digit of the encoding defines the pattern of the data and the second digit gives the number of complete batches. Care must be taken: there is a big difference between a '*partial start'* of a batch and a '*partial batch at the start'* of the acquisition. It is simplest to visualize with some examples, shown in Figure 12.

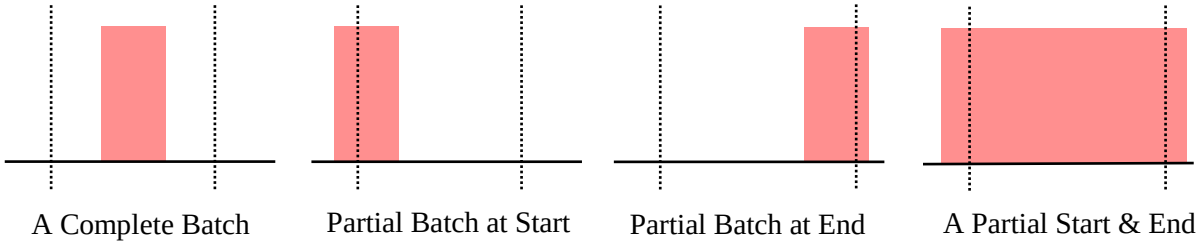


Figure 10: Schematic of the 4 batch structure primitives, dotted lines show the acquisition window and red blocks are complete batches in the machine.

Code	Pattern
0	Partial start and end of batch with zero full bacthes
1	Symmetric Structure with partial start and partial end
2	Symmetric structure with only complete batches
3	Asymmetric data set with partial end
4	Asymmetric data set with partial start

Table 1: Batch pattern codes.

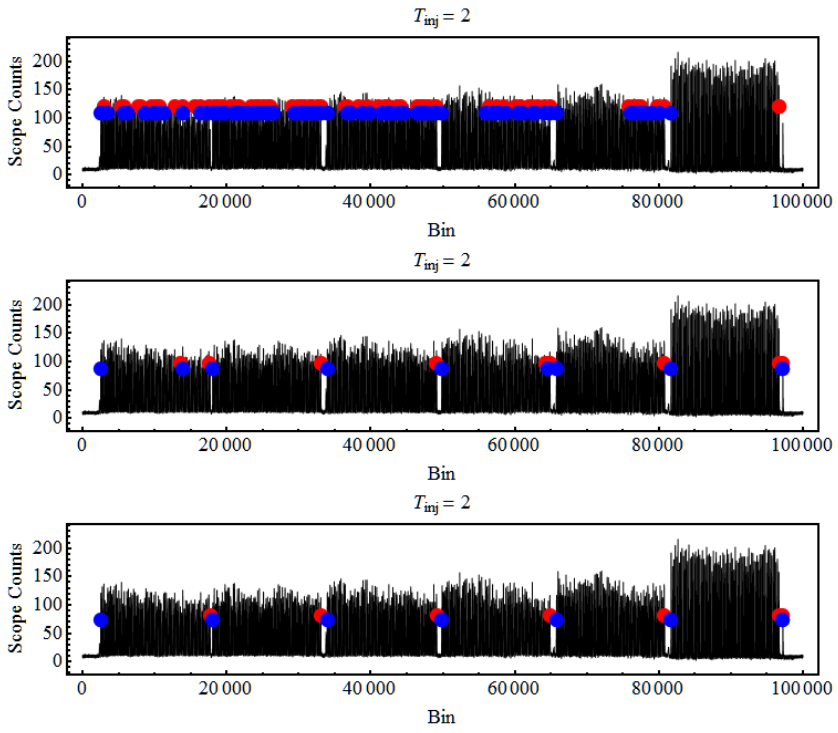


Figure 11: Example batch start and end positions (red and blue) for different T_{inj} values.

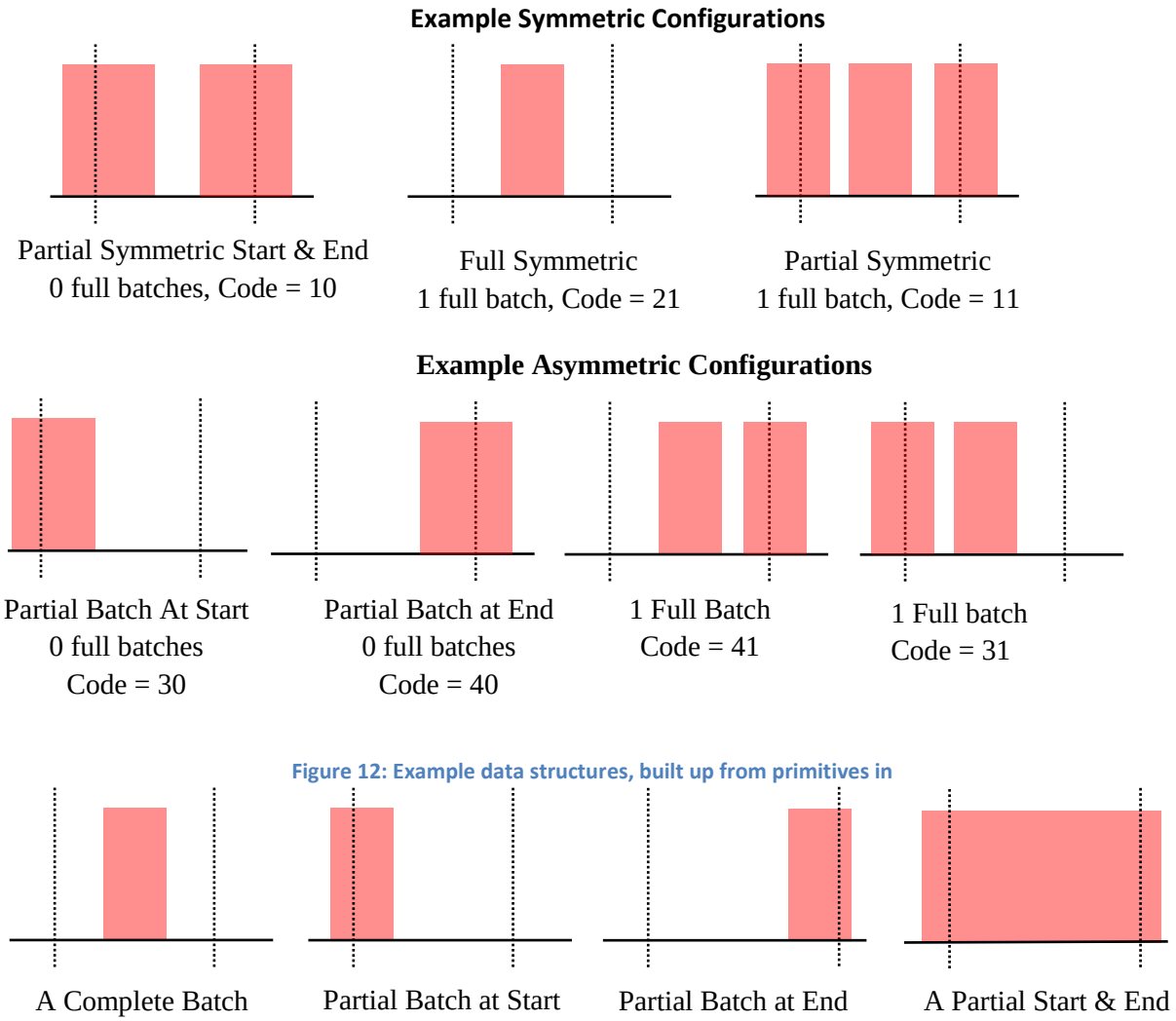


Figure 12: Example data structures, built up from primitives in

Figure 10, dotted lines show the acquisition window and red blocks are complete batches in the machine.

Signal Processing

Once the batch positions are found a particular batch can be selected and then some signal processing can be done, if desired. There are currently two signal processing features, correction of distortion from the transmission of the signal down coaxial cables and some simple smoothing.

Cable Distortion Correction

The distortion correction is explained in more detail here.[16] Simply, the finite resistance of the conductors in the coaxial line cause attenuation of the signal. The attenuation varies (to a good

[16] D.J Scott Distortion in Resistive Wall Current Monitor Signal Transmission Lines *Beams-Doc-4436*

approximation) as the square root of the signal frequency with higher frequencies attenuated more. Knowledge of the length and attenuation constant of the cable means that this distortion can be corrected for. Generally this is only necessary for short bunches that generate higher frequency signal. This class can be slow for many frames it is doing forward and backward transforms on lots of data,

Smoothing

The smoothing is a simple moving average window that computes the local average of N points.

Finding the bucket middles

The next to last step in the preprocessing is finding the RF bucket centres relative to the signal. This is perhaps the most difficult step as it is hard (impossible?) to arrive at a definitive answer.

The tomography algorithms used in TARDIS depend upon the assumption that a bunch oscillates about a fixed point in longitudinal phase space. This fixed point is the centre of the RF bucket the bunch is captured in and represents the synchronous particle, one with the correct energy and phase (time) relative to the RF. The bunch profiles used for tomography must be centered on these fixed points, however, there is no simple way of knowing where these are relative to the RWCM signal. Therefore, we use a number of different algorithms that make educated guesses based on three main assumptions:

- Over long enough periods of time averages of the motion are at a fixed point.
- Beam loading effects are negligible
- There is a single RF frequency that is known precisely

The first assumption implies that for preprocessing (for tomography a minimum of half a synchrotron period is required) WCM data must be taken over enough synchrotron periods to give a '*good-enough*' average. The second assumption means that TARDIS does not include any beam loading effects, these could be included in later versions if deemed necessary. The third assumption enables the precise time between bucket centres to be fixed.

The average of the motion is found by combining multiple frames of data into a single frame, S. The j^{th} part of S is the sum of the j^{th} bin of each F_i:

$$S_j = \sum_i F_{i,j}$$

Example data, with clear bunch oscillations, and its summation are shown in Figure 13 and Figure 14.

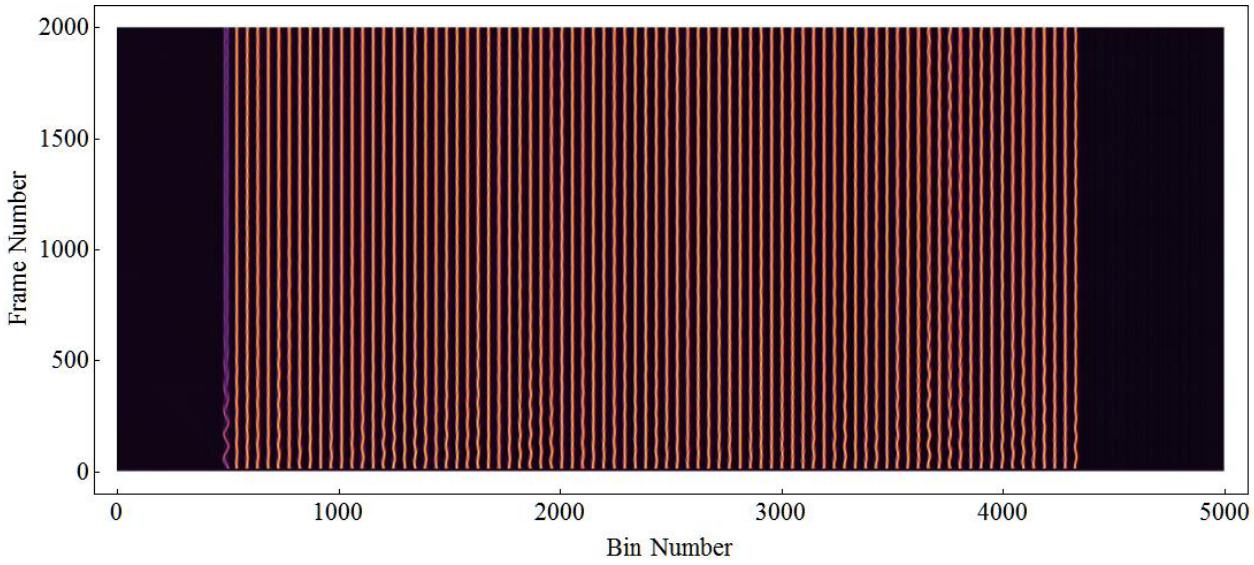


Figure 13: WCM data of a single batch for a few thousand turns. Individual bunches can be seen oscillating about their bunch centres.

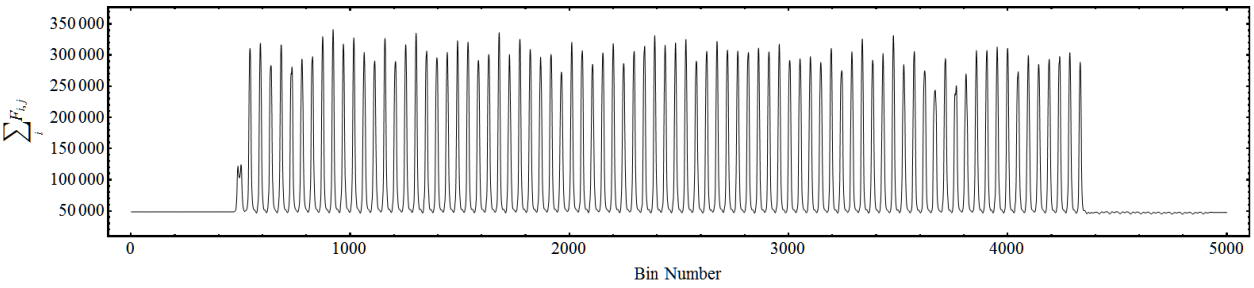


Figure 14: Summation of WCM data along the frame number axis.

Bucket Centre (mid) Finding Algorithms

As the RF frequency is precisely known so is the spacing (in bins or time) between bucket centres along S_j. With no beam loading effects these centres are equidistant, so defining one bucket centre position defines them all, generally the *offset* from the zero bin is used to define all the bucket centres, C_k, where k is the number of buckets in S_j. Methods that use this assumption are called '*fixed*' to reflect this fixed spacing between buckets.[17] Assuming fixed spacing allows different positions to be iteratively tested against the data to arrive at an optimum. There are two main tests that are used, one looks at the moments (or expectation values) when of S_j is portioned into subsets each the length of a bucket, the other looks at the regions close to the individual peaks. Before these algorithms are described an initial guess must be made and regions with no beam that can span many buckets must be accounted for. In general an interpolating function of S_j is used as the length of a bucket, l_{bucket}, in units of bins, given by:

[17] *Local* methods do not assume the bucket centres are equidistant and are described later.

$$l_{\text{bucket}} = \frac{1}{f_{\text{rf}}\,\Delta t_{\text{scope}}}$$

is not a whole number of bins, Table 2 gives some common numbers. The interpolation function assumes that the value of a bin is contained in its centre, i.e. the value for bin 0 to 1 Δt_{scope} is at 0.5 Δt_{scope}, for bin 1 Δt_{scope} to 2 Δt_{scope} is at 1.5 Δt_{scope} etc.

f_{rf}	Δt_{scope}		
Hz	0.1	0.2	0.4
52809176	189.353	94.677	47.34

Table 2: example bucket lengths in bins for different scope intervals

Initial guess for C_1

A good initial guess for the bucket centre position is given by the bin containing the maximum value in S_j, $j_{S_{\max}}$. C_1 is then:

$$C_1 \equiv j_{S_{\max}} (\text{mod } l_{\text{bucket}})$$

from which all C_k can be found. It is assumed that this initial guess is within a few bins of the optimum offset for any method. From this initial guess the *bucket occupancy* can be calculated. This is a measure of if there is beam in the bucket, calculated by comparing integrated signals between each C_k. A ternary system is used to define if a bucket has *no beam*, a *partial bunch* or a *full bunch*. This occupancy is then used to weight the results of the below methods. For example, if the bucket occupancy is *no beam* any method weights that bucket by a factor zero and stops it contributing to the optimum. Whether or not to include partial bunches is entirely up to the user.

Once the initial guess and occupancy of each C_k has been made the optimization of C_k can start. Historically the first method that was tried considered the mean of the expectation values of each C_k using the method of moments.

Moments

For each C_k values of S_j are taken half a bucket length in the positive and negative direction giving $\{x_p, y_p\}$, where p is the number of points the bucket is divided into. The expectation values, E_k, of each one bucketlength distribution is taken, this is a weighted average, or the balance point of the distribution, found using a simple sum:

$$E_k = \sum_p \frac{x_{k,p} y_{k,p}}{y_{k,p}}$$

the mean value is the mean of E_k, omitting each E_k whose occupancy is zero, (and is a partial bunch if desired etc.). Now all that is required is to find the optimum offset that matches E_k with C_k, i.e. the offset that gives zero rms difference between E_k and C_k. Continuing the example using data from Figure 14, the maximum value is in bin 922, giving the initial offset as 22.57 bins, from this a series of profiles, each a bucket length can be generated, shown in Figure 15 with their expectation values. For completeness the mean expectation values for offsets over a complete bucket are shown in Figure 16. There are two zeros for this curve, corresponding to the two sets of distributions shown in Figure 17, these can easily be distinguished by including the gradient of $\langle E_k \rangle$. In practice the initial offset is very close to the optimum answer and the local curve is a straight line, so a simple loop moves up (or down) the curve until a positive and negative value for $\langle E_k \rangle$ is found and then a straight line fit used to find the optimum offset.

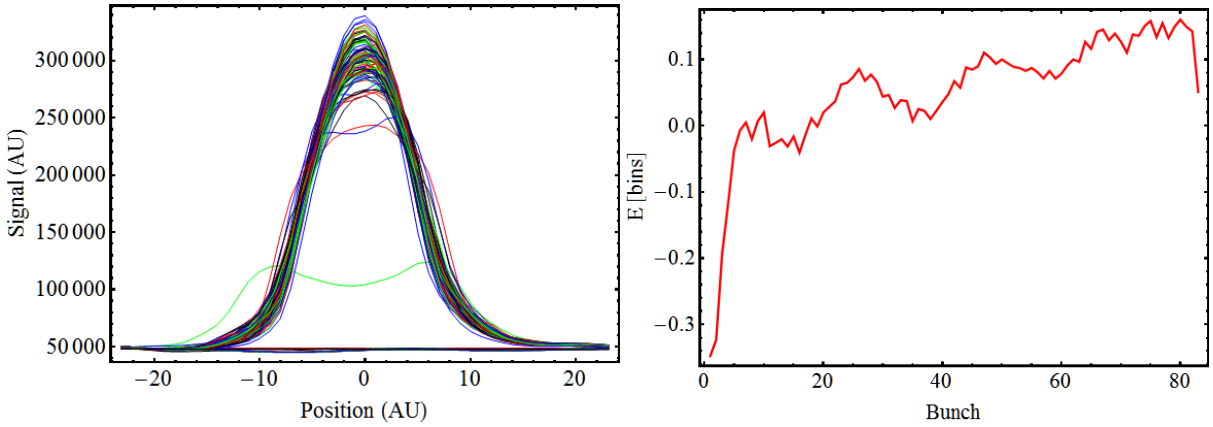


Figure 15: example set of profiles generated from the initial offset and their expectation values..

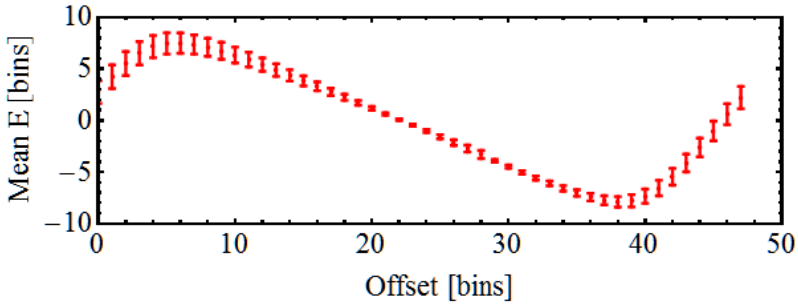


Figure 16: Mean expectation value for increasing offsets, (and standard deviation, shown as an error bar).

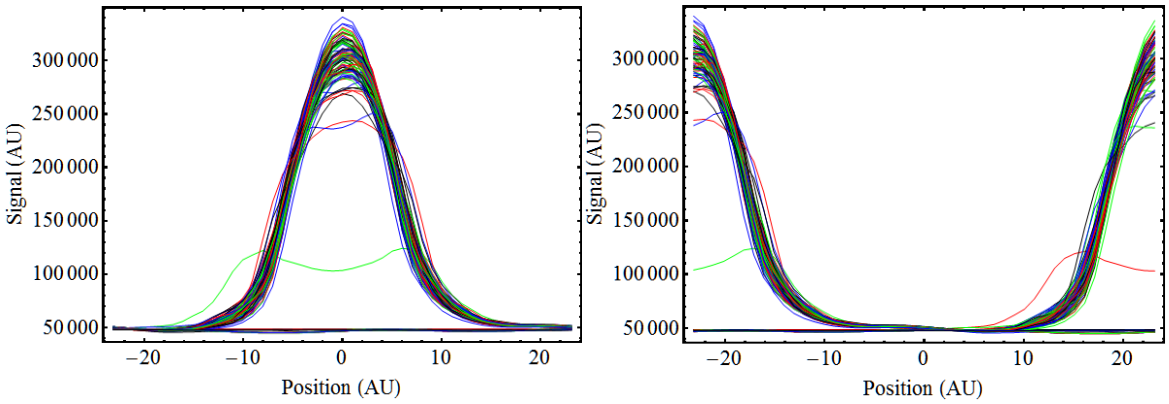


Figure 17: Distributions with offsets of 22.17 and 45.62 that give $\langle E_k \rangle = 0$.

Issues with methods using moments

The principals used in this method are sound, however they assume reliable data and this is not necessarily the case. For example, it has been found in data from the Main Injector RWCM that artifacts are often present that can't be removed easily. These artifacts shift the expectation value by a few bins and this has consequences for the tomography algorithm, an example is shown in

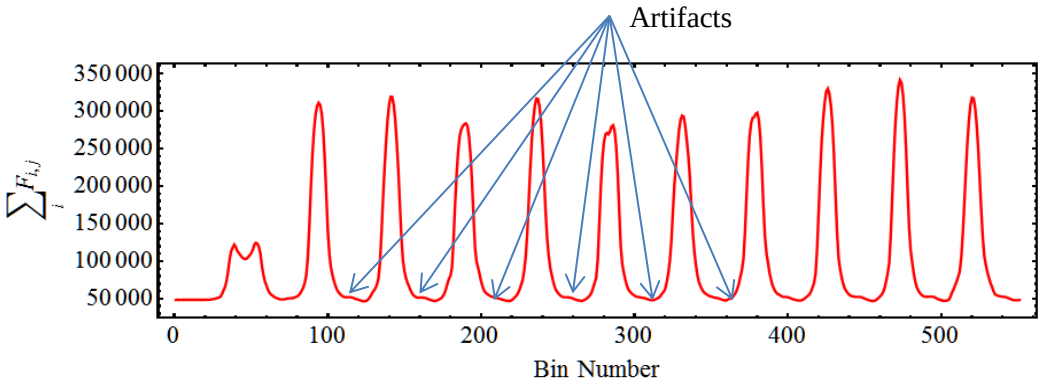


Figure 18. The artifacts could be due to signal transmission, the detector response or something else, it may be possible to correct for them at a later date. For this reason another class of methods looking at the Full width at some fraction of the maximum has been implemented.

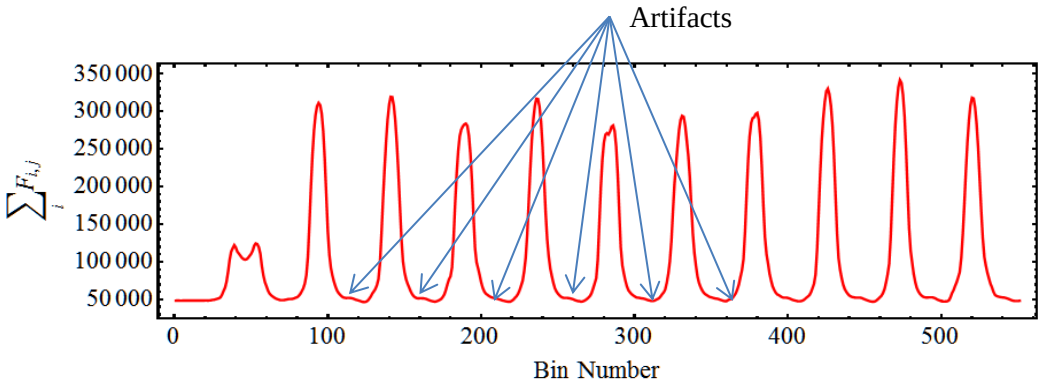


Figure 18: Detail of Sum Frames showing artifacts after each bunch.

Peak Full Width Methods

This method is simpler than using expectation values and has proved to be more robust. The positions of the local peaks of each bunch in S_j are found, there is no enforcement of equidistant spacing between bunches. For bunches that have large oscillations the local peak might not be the correct position to take, for example the first bunch in Figure 18. For this reason the centre of the left and right hand edges (full width) of some cut-off is taken. The cut-off is some fraction of the maximum, for example 90% as shown in Figure 19.

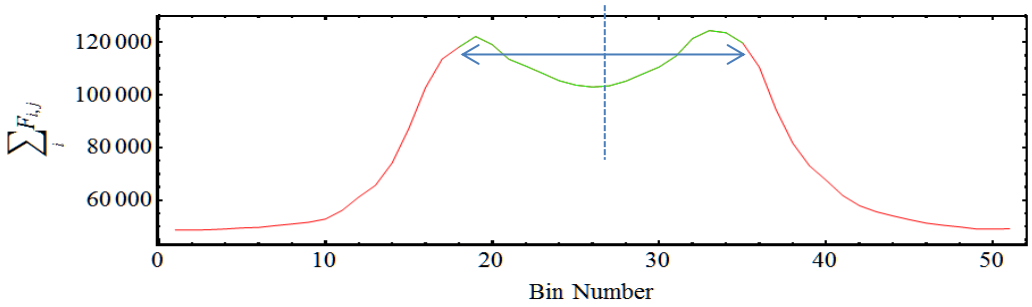


Figure 19: Example of finding the centre of the full width at 90% maximum.

This is a method on its own, called '*local peak.*' It has been found that assuming equidistant bucket spacing gives more precise results (not necessarily more accurate but the tomography algorithm has a way of moving the centres to improve accuracy) and so similar to the moments method the position of the local peaks relative to series of equidistant C_k can be used to find an optimum C_k that minimizes this difference. The robustness of this method can now be seen because artifacts tend to appear after the bunch and are well below the cutoff used in finding the full width.

Profile Generator

This takes the raw data and interpolates each from, over the range required nad then using the bucket mid positions takes half a bucket length either side, then interpolates the number of points required for each profile. It is relatively simple. Then it fills the profile_header and passes the data back to the mastercontroller.