A beam pickup signal analysis system for Atlas

Ezequiel Alfie
Universidad de Buenos Aires

September 10, 2006

Abstract

This document describes a software based acquisition and bunch analysis system intended for the LHC-provided, Atlas beam pickups. A model of the beam pickup and transmission line is presented. The system makes per bunch measurements. It was implemented using LabView, C and Octave. Tests with simulated data and with an arbitrary function generator and oscilloscope online acquisition were done. There is also an analysis on resolution requirements conformance

1 Introduction and objective

The LHC provides one pair (one per incoming beam) of beam pickups to each experiment, including Atlas. These beam pickups are called BPTX and are provided for timing purposes. The signal out of the beam pickup will have information about the actual bunch arrival to the interaction point.

On the other hand the Atlas clock, which comes[8] from the RF, is used by all the trigger modules, and also the detectors. Some of the Atlas detectors rely on a constant phase between the Atlas clock and the actual bunch crossing. The BPTX and clock signal can therefore be used for monitoring such a phase difference for possible drifts.

Atlas decided to use off-the-shelf oscilloscope acquisition and a software based system to address this objective. Since the BPTX signal will be sampled additional information can be obtained from the beam pickup. For instance, the length of the bunch and the intensity can be measured. Also there's the possibility to monitor for satellite (aka ghost) bunches. There is a satellite bunch whenever there is more than one bunch within a bunch crossing period (25ns).

The basic idea of the whole system is to acquire the pair of BPTX signals and the Atlas clock for the duration of an LHC turn once a minute. Using a model for the bunch charge distribution, beam pickup and acquisition system it is possible to make a fit for each bunch seen on the oscilloscope to find out the bunch parameters (time of arrival, length and intensity).

This fit should give precise information on bunch arrival and then this information can be used to measure phase diffence with the Atlas clock. The resolution on phase measurement must be better than 50 ps. This requirement came from the liquid argon calorimeter[12] detector, whose time resolution is 50 ps and therefore needs a phase measurement resolution better than their time resolution as it was informed[9].

This document describes the models used and a software system to acquire, analyze the beam pickup signals and make the mentioned fits on a whole aquired signal. It was programmed in Labview, C and Octave.

2 Model function

2.1 Introduction

To be able to make a fit from measured (sampled) signal by the oscilloscope for each bunch and extracting parameters from it, one has to be able to do the opposite process: being able to start from the parameters and reconstructing the expected signal.

This requierement is intrinsic to every measurement system which is based on the estimation of parameters by doing a fitting procedure: one needs an 'a priori' knowledge of how the system behaves as a function of the parameters, we will call this the *model function*.

2.2 The bunch

The bunch charge profile is assumed to be gaussian distrubuted (longitudinally), so let's first define the parameters to be fitted:

- t_0: the center of the gaussian, time when bunch was nearest to beam pickup.
- σ: corresponding to the standard deviation of the bunch charge's distribution, also corresponding to the bunch length and its value is related to the energy (higher energy $\implies$ shorter bunch).
- N: the number of protons carried by the bunch. Directly proportional to the pickup signal amplitude

A single bunch charge linear distribution will be then:

$$\rho(t) = \frac{Ne}{\sqrt{2\pi}\sigma c} \exp -\frac{(t-t_0)^2}{2\sigma^2} \tag{1}$$

For typical intensity and length parameters on the LHC see [5].

2.3 Button pickup model

The pickups to be used are of the electrostatic button type. A mathematical model on the button pickup can be found in [1] and [2].

To have a general idea on how this pickup works, let's first imagine a metal pipe and the beam passing through it. The charge of the beam induces an image charge distributed along the walls of the pipe. If the beam is centered then this mirror charge distribution does not depend on the axial angle.

As the beam travels along the pipe, a distribution of mirror charge also travels through the metal walls of the pipe, following the beam and mimics the beam's longitudinal shape. The idea of a button pickup consists of "collecting" this charge at a specified area in the pipe, covering some surface of the original pipe's wall.

Now let there be a button pickup which is rectangular over the pipe wall, that means it is a rectangle on the $zeta - \phi$ plane, being z the longitudinal coordinate and ϕ the axial angle. In this case, this button would cover a surface over the pipe's equal to:

$$S = \Delta z \, \Delta \, \phi \, 2\pi b$$

being b the pipe's radius. If the longitudinal coverage of the button pickup (Δz) is small compared to the beam longitudinal length, mathematically this is:

$$\frac{d\rho(z,t)}{dz} \Delta z \ll \rho(z,t) \tag{2}$$

then the mirror charge, on the button pickup would be:

$$Q(t) = \Delta\phi \, 2\pi b \, \Delta z \, \rho(t) \tag{3}$$

rewriting in terms of the surface coverage over the pipe's wall:

$$Q(t) = \frac{S}{2\pi b} \, \rho(t) \tag{4}$$

Even in the case of the pickup not being rectangular on $zeta - \phi$ plane, if it satisfies the inequation 2, then the mirror charge on the button will be still that of equation 4.

Note that this will not hold true in the case of the LHC running with 7 TeV proton bunches since the button pickups installed[13] have a length of 61 mm ([13]) and the RMS bunch length at that energy will be 75.5 mm, see [5] .

So, one could use this approximation for a differential surface $dS = width(z)dz$ and integrating in dz, having then for each time t:

$$Q(t) = \int \rho(z - \beta ct) \frac{width(z)}{2\pi b} dz \tag{5}$$

Now we can calculate the current on the button pickup:

$$I_{img}(t) = \frac{dQ(t)}{dt} \tag{6}$$

To calculate the button voltage we need to know which impedance "sees" the button. Note that the current source of the button does not depend on the impedance, this correspond to the fact that we are assuming that the beam generates perfect mirror currents.

This impedance is composed of two parts, a (parasitic) capacitive part C_b, and a real part corresponding to the coaxial cable load ended in its own characteristic impedance (Z_0). These are two independent paths, so we can think of an equivalent circuit that has: an ideal current source with value $I_{img}(t)$ and two shunted impedances, one corresponding to the parasitic capacitance and one for the coaxial cable load. Regarding values, in this particular setup the C_b was informed[11] to be 16 pF and the Z_0 is standard 50 Ω.

In terms of frequency:

$$V_b(j\omega) = \frac{I_{img}(j\omega)}{C_b} \frac{1}{j\omega + \frac{1}{C_b Z_0}} \tag{7}$$

By now we have a complete model upto the "cable input". If we had a cable with no attenuation nor dispersion and we connected the cable from the button pickup to a load of value Z_0, we would measure over this load a voltage $V_b(t)$.

2.4 Transmission line model

The effect of 200 m transmission line is not negligible, specially at high frequencies. I used a model[4] which is based on the assumption that the attenuation is dominated by skin effect losses and these are proportional to the square root of the frequency. Such a transmission line, if ended at its characteristic impedance, has the following voltage step response (trivial propagation delay not included):

$$a_{cable}(t) = u_c(t) \ \mathrm{erfc}\sqrt{\frac{B}{t}} \tag{8}$$

This attenuation model has only one parameter B, which should be matched against the real datasheet of the cable. The procedure done was fairly simple, and not optimal: trying with different values of B and convolving a 1 GHz sinusoidal with $a(t, B)$ until getting an attenuation to the sinusoidal that matches the one specified on the datasheet[3] of the cable that corresponds to 1 GHz and 200m.

To add this transmission line model to the above mentioned button voltage, requires to convolve the button voltage coming from eq. 7 with the coaxial cable impulse response:

$$h_{cable}(t) = \frac{da(t)}{dt}$$

2.5 Oscilloscope analog input model

Finally the end part of the acquisition system is the oscilloscope analog input. Every oscilloscope has a rolloff and in fact it is needed to avoid aliasing. Oscilloscope rolloff characteristic hasn't been measured yet, but the procedure is simple: apply a sinusoidal to a channel of the scope and measure its amplitude/phase as a function of frequency, sweeping up to half the sampling rate. Fourier transforming the frequency response would yeld in another convolution kernel to stack.

3 Algorithm to evaluate the model function

The fact that models of button pickup, transmission line, and scope rolloff are linear models allowed for a separation of the expected signal onto two different parts: the differentiated gaussian containing the parameters (t_0, σ, N) and a stack of impulse responses that convolved together gives a global impulse response, which depends on physical model and acquisition system but not on bunch parameters.

Using the "electrical equivalent" of the beam pickup (see 7) and transmission line response, we may write the output of the system using Laplace's s variable like this:

$$Out(s) = \frac{I_{img}(s)}{C_b} \frac{1}{s + \frac{1}{Z_0 C_b}} H_{cable}(s) \tag{9}$$

The first factor corresponds to time derivative of 4, the second and third are together the transform of the following convolution:

$$h_{rc,cable}(t) = \int_0^t h_{cable}(\tau) \exp -\frac{(t-\tau)}{Z_0 C_b} d\tau \tag{10}$$

After obtaining an evaluated (sampled) $h_{rc,cable}$, which is called `modelkernel` on the implementation, it is possible to convolve it with the effects of the analog input of the scope and the correction for the finite size of the button pickup (note that the former will be likely obtained numerically and at discrete time or frequency points) to obtain a corrected `modelkernel`, which is not done on current implementation.

Then to evaluate the model function at runtime one has then to convolve in a discrete way `modelkernel` with a differentiated gaussian.

4 Implementation of the fitting

4.1 Overview

The `modelkernel` is generated at design time once. An Octave code (`make-modelkernel.m`) produces a C header file which declares `modelkernel` as a constant array of doubles. This C code is responsible of evaluating at runtime the model function: it gives the expected signal given the parameters t_0, σ and N. The name of this C function is `modelfunction`.

The reason for writing it in C is the speed since to make a fit, the model function has to be evaluated many (80-200) times. In fact it was first written in Labview and the speed improvement for having it on C was 25-30 times.

To make the fit itself I'm currently using the Labview's Levenberg-Marquardt fit block. Calling C code from LabView is really easy and it was used in other parts beyond model function evaluation. I have also tested successfully the fitting using a C implementation[6] of the same algorithm instead of the LabView one, which might be useful to further improve speed.

I have provided a Makefile which automatically creates `modelkernel` and compiles the C functions into a shared object.

4.2 Duration and number of samples

I had first to decide how long (in time) to sample the `modelkernel`. Taking into account a worst case of 375 ps for the σ, which corresponds to the LHC protons at injection[5] (450 GeV). I fixed the duration of the model function to 10 ns because all the signal contents where contained into that duration. An optimal value will be perhaps less than that (6 ns - 8 ns).

The length of `modelkernel` is 500 samples. the `modelfunction` samples 500 points a differentiated gaussian using the parameters given and convolves with `modelkernel`, but the convolution itself is evaluated at a submultiple of 500: for instance 100 samples would correspond to 10 GS/s if kernel duration is 10 ns. This process can be thought as making the convolution at a higher sampling rate, being more "similar" to continous convolution, and then subsampling.

However, number of samples and duration is parametrizable on the `make-modelkernel.m` Octave source file. Changing values is easy as long it is used consistently from the fitting part. That is why I provided additional functions to be called from LabView whose names are: `getmodelkernellength` and `getmodelkernelsecs`. The names are self-explanatory but it should be noted that `modelkernelsecs` equals also to the length in seconds of the fitting window.

4.3 Applying the fit to the bunches on the waveform

The waveform from the beam pickup signal acquired by the scope will have a train of bunches, in the final case this should correspond to a whole LHC turn[7] which has 3565 bunchs. The algorithm to apply the fit to each bunch is fairly simple:

Having a threshold defined, the program finds all the peaks of the signal. This peak finding is implemented in C, the function is named `peakindices` and is called from LabView. This results in an array of sample indices, then the program iterates over those indices and checks whether the difference in sample number between the current one and the previous is greater than the number of samples needed by the fitting. In that case it is "usable" index, if not then this would at least mean that a satellite bunch has been detected. Then for each usable sample index, the program "cuts" a window asimetrically around the index, starting from the index minus a variable called `leftsamples` and having the length needed by the "single window" fitting. This variable is closely related to the initial guess on time of arrival because it should be near the peak found.

Applying the fit to each window results in the three parameters, but the time of arrival estimated is relative to the start of the fitting window, so adding the start of the window time, a time of arrival of the bunch relative to acquisition start is obtained.

5 Testing with simulated data

Using the model function I generated a bunch train waveform containing 400 bunches with 3% gaussian white noise added.

All the bunches had the same parameters and where separated by 25ns. The parameters chosen where t_0: 3.23 ns (corresponds to first bunch at 3.23 ns, second at 28.23 ns, etc), σ: 0.25 ns (corresponds to 7 TeV), and unit intensity.

Then I applied the global fit to the waveform. A phase difference is calculated between time of arrival of bunches and a virtual 25 ns clock, which is equivalent to the remainder of the division of time of arrival by 25ns.

After that process I had an array of phases, of lenghts and intensities. Figures 1, 2 and 3 shows the resulting histograms. No special correlation was found between successive fits results, as it is expected.

The mean for phase measurement was 3.23032 ns and the corresponding standard deviation was 5.3 ps, obviously this is conformant to the requirement of better than 50 ps.

The mean σ was 249 ps, with 10 ps standard deviation implying 4% resolution on length. Mean intensity was 1.00125 with a 2% resolution.

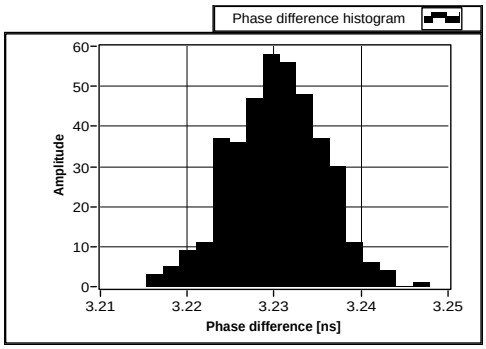

Figure 1: Histogram for phase measurement using simulated data

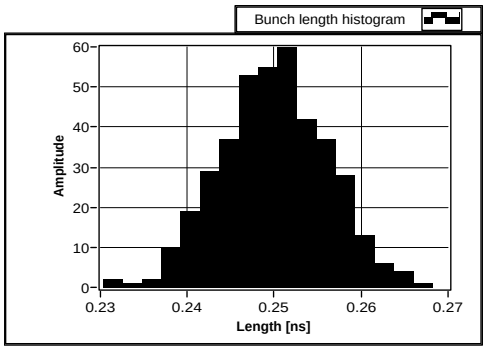

Figure 2: Histogram for length (σ) measurement using simulated data

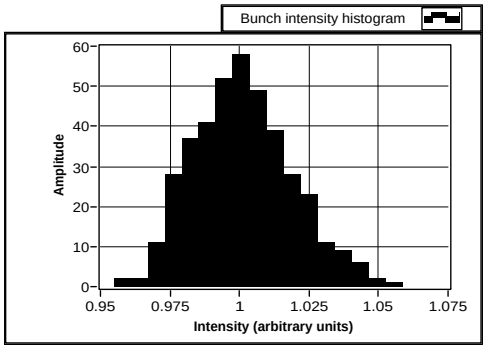

Figure 3: Histogram for intensity measurement (arbitrary units) using simulated data

The resulting correlation matrix is on equation 11. The order in this matrix is t_0, σ, N.

$$\begin{bmatrix} 1.0000 & 0.0802 & 0.1189 \\ 0.0802 & 1.0000 & 0.6396 \\ 0.1189 & 0.6396 & 1.0000 \end{bmatrix} \tag{11}$$

6 Testing with hardware

6.1 Overview

Having a model for the expected signal it made sense to use an arbitrary waveform generator and load the signal onto it and then testing the analysis system (the "global" fitting) to a signal coming from an oscilloscope, instead of being simulated locally on the PC. The idea is that not only the fitting procedure is tested but also the oscilloscope setup and acquisition part of the software developed.

Even more, it was useful because unexpected results where obtained and this lead to an important conclusion.

It should be noted that no *real* test can be done until mid of 2007, when the first beam test will happen. Only whith a real beam the model itself can be put into test.

6.2 Setup

The instruments used where:

- Oscilloscope Tektronix TDS3054B 4 channels 5 GS/s 500 MHz
- Arbitrary waveform generator Tektronix AFG3252 dual channel 2 GS/s 240 MHz

Maximum sampling rate of the waveform generator is 2 GS/s. This is not enough to output an expected pickup signal and a bunch crossing period of 25 ns. So the time was "scaled down" 1000 times. This means that the new BC period is 25 microseconds.

I first generated an expected signal containing 25000 samples corresponding to a duration of the new BC period, that is using the waveform generator at 1 GS/s. The parameters of the bunch loaded where $t_0 = 3.23ns$ and $\sigma = 0.41ns$.

The procedure for generation is conceptually the one described on 3, but it is different from the `modelfunction` function, since number of samples is higher and duration is longer. This generation was made using Octave and the output is a plain text file containing voltage values for each time.

The function generator has an USB port for use with memory sticks. It can load waveforms from it but it only understands a special binary format. I used the Tektronix ArbExpress software to open the text file and saving in the binary format which has a `.tfw` extension.

After loading the files the function generator was setup to repeat on one of the channels the bunch waveform at 40 kHz, and the offset was adjusted. The output was conected to one channel of the scope. Also the synchronous output of the function generator was used as an input channel. The lengths of the cables going from the generator to scopes were the same.

The synchronous output is a square signal which has a rising edge at the same time the signal output channel is at the beginning of the loaded waveform. Therefore, it has a frequency of 40 kHz and is phase aligned with the output channel. This signal was used as an "Atlas clock". So bunch-to-clock phase measurement is calculated from the results of the fit applied to the bunch signal and the edge location of the sync signal.

The oscilloscope was setup by the software to acquire from both channels 10000 points at 10 MS/s, triggering on the sync output at half peak voltage. The rise-time of the sync signal was around 7 ns which is much less than the 100 ns sampling interval.

6.3 Test and results

For each acquisition the "global fitting" returns a matrix of parameters, one row per bunch. The acquisition is repeated and the history for each parameter is recorded and a histogram made. Histograms made for intensity and bunch's length didn't differ significantly from those on the test with simulated data.

The phase measurement, calculated as the time difference between bunch's time location relative to acquisition start and nearest edge location of sync channel, showed a strange behaviour. Figure 4 shows the history over many acquisitions of the phase measurement results. It is possible to see at least two effects: one is the trends, constrained to each acquisition, and the other are the jumps between acquisition.

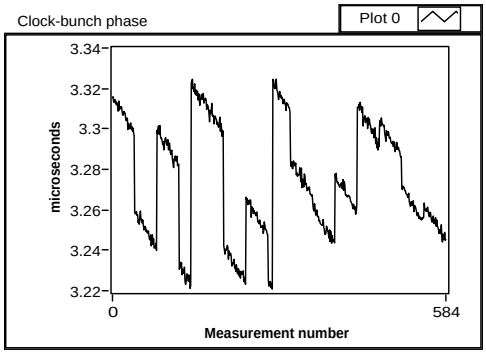

Figure 4: History of the phase measurement on the harware test, many acquisitions

The "trend effect" was strange at first because, in principle, there should be no correlation between differnt phase-measurements. This effect was identified as a misscalibration of the oscilloscope internal clock, it is possible to see a delay of 20 ns in 1 ms (20 ppm). The reason for doing so was that the function generator has 12 significative digits on the selection of the frequency and in fact, setting 40 kHz - 1.7 Hz on the function generator flats this trend.

The jump effect can be understood as an uncertainty in acquisition start. When the oscilloscope is waiting for a trigger to start an acquisition it is already acquiring. It is sampling at the same sampling rate that the user asked for the acquisition, and whenever it detects that the current sample discretized voltage value is greater than the threshold then it starts the acquisition using the sample before that one. So there is an uncertainty in acquisition start because it is not possible to know when within the sampling interval happen the "real threshold".

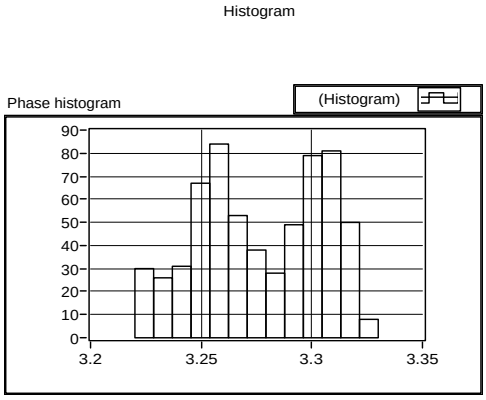

Figure 5: Histogram of the phase measurement on the hardware test, many acquisitions

It is true however, that these effect affected the phase measurement because the clock signal had a very sharp rise, compared to the sampling interval. Figure 5 shows the histogram for the bunch-clock phase measurement. Resolution in this case is 57 ps, and is much worse than in the case for simulated data.

6.4 A "differential" phase measurement

The effects seen on the hardware test gave rise to many questions. In particular if it is possible to have sub-sampling rate resolution (one order less, for instance) to the phase measurement. So a simple test was done: since the arbitrary waveform generator has 2 channels I set up both channels to output a bunch train signal one delayed to the other 45ns (45% of the sampling interval).

Then the fitting was applied to both channels and a phase measurement was made by calculating differences between time of arrival of bunches of the two channels. The history recorded for this measurement is on figure 6. The 45ns delay is recovered with a resolution comparable to the one of the simulated data test.

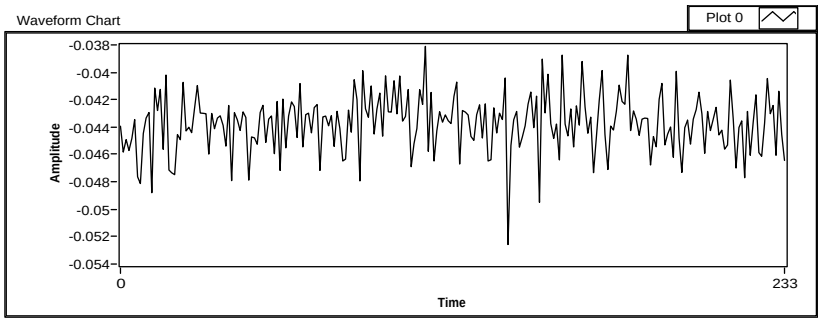

Figure 6: History for phase measurement on the differntial hardware test

7 Conclusions

A LabView and C software to analyze the beam pickup signal using and measure bunch parameters was made, based on the models cited and algorithms described. This software is functional and interacts with the oscilloscope to acquire online data and process it to estimate the parameters. Also, besides the online software, a "framework" was made to work with the expected signal using Octave scripts and prepare data for the arbitrary function generator.

This measurement software was put into test. Intensity and length measurement performed well in both the test with simulated data and the hardware tests. But phase measurement resolution is limited by clock edge detection uncertainty. This uncertainty on the order of one sample interval is present at least with clock signals with rise-time shorter than the sampling interval, as it was the case on the first hardware test.

However it was shown on the "differential" phase measurement that subsampling interval resolution is possible. In fact in that measurement resolution was 2 orders of magnitude better. This possibility came from the fact that "a priori" knowledge of the signal was used. The knowledge in this case is the information that signals where generated with a model with certain parameters.

For the clock signal there isn't such "a priori" information. The possibility of having extra knowledge will depend on how exactly the clock signal will be and what is known in advance. One example of extra knowledge will be the following, perhaps ideal, case: if the clock signal has a maximum frequency component which is less than half the sampling frequency then full reconstruction of the analog signal is possible using sampling theorem. Then to obtain clock edge timing information apply a voltage threshold and see which time point crosses the threshold. Note that in this case the time points can be arbitrarily separated (less than sampling interval, of course).

8 Code and files

Here there is a listing of the source code files with a brief description. Not all files are listed since many were for specific testing purposes.

8.1 C/Octave files

On subdirectory `modelfunction/`

- `modelfunction.h` Header with declarations of functions in `modelfunction.c`.
- `modelfunction.c` Source code to evaluate the model function at run time.
- `make-modelkernel.m` Octave source to generate the system impulse response.
- `hrccable.m` Used by `make-modelkernel.m`.
- `modelkernel.h` Header containing the system impulse response.
- `modelfunction.so` Shared object containing functions inside `modelfunction.c`, dynamically linked from LabView.
- `Makefile` to build the shared object `modelfunction.so` from sources.

On subdirectory `auxiliar/` there are some helper functions used from LabView which are not related to evaluating the expected signal, see inside source files for more information.

- `auxiliar.h` Header with declarations of functions in `auxiliar.c`.
- `auxiliar.c` Source code to evaluate the model function at run time.
- `auxiliar.so` Shared object containing functions inside `auxiliar.c`, dynamically linked from LabView.
- `Makefile` to build the shared object `auxiliar.so`

8.2 LabView files

Inside `bunchtrain-analysis/`:

- `pulsemodel.vi` Implements the model function by linking to the `modelfunction.so` shared object.
- `fixedwindowfit.vi` Performs a fit on a window corresponding to a single bunch.

- `pulsetrain.vi` Samples a train of simulated bunches, it takes a matrix of bunch parameters.
- `bunchrain-analysis.vi` Performs the global fit on a beam pickup waveform and gives the matrix of parameters.
- `repetitive-acq.vi` A basis block for setting up scope, acquire or simulate, fit and make histograms from results.
- `mypulsetrain.vi` A block to sample a simulated train of bunchs with given parameters matrix.
- `fit-simulated-data.vi` Block to perform the global fit to a simulated train of bunchs. It outputs histograms, mean values and covariance matrix.

Inside `oscilloscope/`, names are self-explanatory.

- `oscilloscope-setup.vi`
- `oscilloscope-acq.vi`
- `oscilloscope-close.vi`

8.3 Arbitrary waveform files

I have prepared a set of files corresponding to one bunch with differents parameters. They have 25000 points, in ASCII format and in the binary `.tfw` format, which the function generator understands. All of them have unit intensity.

- `awf-323-025-1` t_0: 3.23 ns, σ 0.25 ns.
- `awf-323-041-1` t_0: 3.23 ns, σ 0.41 ns.
- `awf-327-041-1` t_0: 3.27 ns, σ 0.41 ns.
- `awf-329-041-1` t_0: 3.29 ns, σ 0.41 ns.

and the following Octave scripts built the above mentioned waveforms:

- `genawf.m` Builds an expected signal, outputs in ASCII format, depends on the files below.
- `hrccable.m` Same as used for `make-modelkernel.m`.
- `diffpulse.m` Script to sample a differentiated gaussian.

9 Related Projects

Other experiments of the LHC will also use the beam pickups, they may have different requirements, but anyway it is worth looking how is their monitoring system. Particularly LHCb's Beam Phase and Intensity Monitor (BPIM[14]) is an online, hardware based, made on a custom VME board system.

But also from the machine side there are going to perform and online publish beam profile related measurements[15], for instance satellite bunch detection is included.

References

[1] Beam Position Monitor Engineering, Stephen R. Smith, Stanford Linear Accelerator Center

[2] Signal processing for Beam Position Monitors, Giuseppe Vismara, Cern

[3] Technical Specification for a low loss cable with cellular PE and Cu screening tube, CERN-LHC

[4] Transient analysis of coaxial cables considering skin effect, R.L. Wiginton and N.S. Nahman, Proc IRE, vol 45, February 1957

[5] `http://edms.cern.ch/cedar/plsql/navigation.tree?top=CERN-0000020013`

[6] Levenberg-Marquardt nonlinear least squares algorithms in C/C++ (`http://www.ics.forth.gr/ lourakis/levmar/`)

[7] Nominal Proton Bunch Pattern in the LHC for 25nS Spacing (`http://lhcp.web.cern.ch/lhcp/ab-ltc/ltc_2003-02a.pdf`)

[8] `http://ttc-upgrade.web.cern.ch/ttc-upgrade/New_system/RF2TTC.htm`

[9] Philippe Farthouat, private communication (`Philippe.Farthouat@cern.ch`)

[10] Christian Boccard, private communication (`Christian.Boccard@cern.ch`)

[11] Thilo Pauly, private communication (`Thilo.Pauly.cern.ch`)

[12] Liquid Argon Calorimeter website: `https://twiki.cern.ch/twiki/bin/view/LAr/WebHome`

[13] Schematics of the installed Beam Pickup for Experiments (BPTX): `/afs/cern.ch/user/e/ezequiel/references/BPTX`

[14] http://lhcb-online.web.cern.ch/lhcb-online/TFC/html/BPIM.html

[15] https://edms.cern.ch/file/328145/2.0/LHC-B-ES-0005.00-rev2.0.pdf