A Matlab Interface to Acquire and Analyse Beam Spectra Using a Large Memory Digitizer

A. Välimaa

CERN

August 27, 2014
Let Me Present Myself

WHO
Alpo Välimaa

FROM
Finland, Aalto University

STUDYING
Engineering Physics and Mathematics

HERE FOR
CERN Summer Student Program June 16 - August 29, 2014
For Operations Research there exists a system (OASIS) which satisfies the users. For MD, there is potential for a flexible data acquisition tool that can provide more information than it is needed to Operate at known parameter region, to have a better view when trying new things.

The idea was to develop a user interface for digitizer to gain w.r.t. some specs if compared to the oscilloscope in use and to have an environment where to start data post-processing.
Data Acquisition Tool

What? A digitizer by SP Devices
Model ADQ214
Quality 14-bit
Memory 64 MSamples
Sampling frequency $800/n$ MHz, $n=2,3,4,...,20$
Flexibility Great
Resolution At 400 MHz one achieves a frequency resolution of 1 kHz for an event lasting 1 ms.
More Info Here
With ADQ214 we are fully covering the basic cycle length in PS which is 1.2 seconds.
Data Acquisition Tool

One can think of ADQ214 digitizer being a camera with 64 megapixels. A camera producing such images for which exposure time can be changed afterwards.
Oscilloscope vs Digitizer

Device  Tektronix MDO4054-3
Strength  All-in-one package
😊  More tools than necessary for our purpose

Device  SPD ADQ214
Strength  High-quality, fast rate and large memory
😊  Requires application development
Oscilloscope vs Digitizer

Quality 12-bit 😞
Memory 4 ch × 20 MS 😞
Analog BW 500 MHz
Signal processing Analog 😊
Spectrogram No 😞

Quality 14-bit 😊
Memory 2 ch × 64 MS 😊
Analog BW 850 MHz
Signal processing After digitization with computer
Spectrogram Interface for it now developed 😊
Oscilloscope vs Digitizer

Upgradability  Buy a new one 😞
Price  More 😞

Upgradability  Upgrade device, computer or modify code.
Price  Less 🙂
Matlab Interface - GUI

**GUI** Controlled by Matlab class. Consists of a spectrogram view, slider widget and file open button.

**View** $512 \times 1024 \approx$ half MP. Use click+ctrl and click+shift to zoom in and out. Adjust spectrogram parameters to the view.

**Slider** Use slider to balance with time and frequency resolutions.
The Code (more in the appendix)

classdef myClass < matlab.System
properties
  gui_h % GUI handle structure
  in % Input data structure
  out % Output data structure
  layerNumber % This indicates of the layer
  layer % Use this to navigate
  checkTimer % Timer object
end

properties (Logical)
  debug=false;
end

methods
%--------Main functions---------------------
end %Methods(Static)
end %Classdef
Σ signal. **Whole cycle** shown from injection to extraction. Note constant betatron frequency but varying content of high frequencies during cycle.
nToF Cycle Spectrum

Overview of TOF Cycle as Seen in H Signal. RF Clock.

Corresponding spectrogram of nToF cycle but now with H signal.
nToF Cycle Spectrum (zooming)

Fluctuation of high frequency content can be seen during the first ms after injection. It implies oscillation of bunch size.
Transition is a critical point in the cycle regarding beam losses. This was seen also in H signal as sudden rise of highest frequencies.
nToF Cycle Spectrum

Extraction

For nToF cycle we see the so-called sharp extraction. The whole beam is extracted at once.
We had an algorithm to find the nearest significant peak (nearest in the frequency domain).
We developed it further to follow a curve.
One can use it f.ex. to extract information of the synchrotron frequency.
With external clock we lose the information of time but we can record the beam position w.r.t. RF clock and gating is possible.
In this presentation I reported the work done during my summer studentship at CERN.

- I developed a Matlab class and a Matlab GUI for a fast and easy post-processing of the data acquired with the ADQ214.

- I show few examples using the $\Sigma$ and $H$ signal of the nToF cycle: one single data measurement the “64 MP” image can deliver a lot of information and give almost a complete overview of the beam passage in the cycle.

- To spot something more local, one can use the gating.
Conclusions

In this presentation I reported the work done during my summer studentship at CERN.

- I developed a Matlab class and a Matlab GUI for a fast and easy post-processing of the data acquired with the ADQ214
- I show few examples using the $\Sigma$ and $H$ signal of the nToF cycle: i one single data measurement the “64 MP” image can deliver a lot of information and a give almost a complete overview of the beam passage in the cycle
- To spot something more local, one can use the gating.

Thank you!
classdef myClass< matlab.System
    properties
        gui_h % GUI handle structure
        in % Input data structure
        out % Output data structure
        layerNumber % This indicates of the current ... layer
        layer % Use this to navigate through ... different zoom layers
        checkTimer % Timer object
    end
    properties (Logical)
        debug=false;
    end
    methods
    %--------Main ...
    functions-----------------------------------------------
        %Class and GUI
function obj = myClass(varargin)
function createGui(obj, varargin)
function initializeMyClass(obj)
function delete(obj)
function switchChannel(obj)
function closeFigureFcn(obj, varargin)

% Data Processing
function ...
    [nfft, noverlap, nWindows] = giveParametersForFFT(obj, varargin)
function mySpectro(obj, varargin)
function myMatrix = addWindow(obj, myMatrix, varargin)
function plotSpectro(obj, hAxes)

% Image
function updateGraph(obj, hAxes)
function updateLabels(obj, varargin, callerFcn)
function ...
    [h, ds_data] = dsimagesc(obj, data, hAxes, method)
function computationalZoom(obj)
function select_pts_callback(obj,\textbackslash \textbackslash,\textbackslash \textbackslash)
function zoom_in_callback(obj,\textbackslash \textbackslash,\textbackslash \textbackslash)
function zoom_out_callback(obj,\textbackslash \textbackslash,\textbackslash \textbackslash)
function calculate_callback(obj,\textbackslash \textbackslash,\textbackslash \textbackslash)
function view_callback(obj)
function axesButtonDownFcn(obj,hObj,\textbackslash \textbackslash)
function ZoomInCallback(obj,\textbackslash \textbackslash,\textbackslash \textbackslash,\textbackslash \textbackslash,\textbackslash \textbackslash,\textbackslash \textbackslash,\textbackslash \textbackslash)
function showRegion(obj,fCenter,fWidth,t0,tend)

%Peaks
function ...
    [locMatrix,locMatrix_i]=findHarmonics(obj)
function ...
    [line,line_i,ind]=findNearestLine(obj,varargin)
function ...
    [line,line_i]=loopAlongTheLine(obj,method,startingPoint)
function [peak,peak_i]=getNearbyPeak(\textbackslash \textbackslash,data)
The Code IV

function ...
    [peak,peak_i]=enhancedPeakPrecision(obj,peak_i,values)
function ...
    [loc,loc_i,amplitude]=findNearestPeak(obj,varargin)
function ...
    [peaks,peaks_i]=findPeaks(¬,type,value,varargin)
function ind=findNextBunch(obj,i0)

%Slider
function updateSliderValue(obj)
function slider_callback(obj,hObj,¬)
function ...
    [smin,smax]=EvaluateSliderValueRange(obj)

%Gate
function ...
    getBaseFreqAtBunch(obj,baseFreq_i,baseFreq,nBunch)
function ...
    gateMatrix=gateSignal(obj,nTurns,clockType,varargin)
function loopGatedSignal(obj,gateMatrix)
function plotTransition(obj)
function plotInjection(obj)
function plotExtraction(obj)

%Slices - amplitude plots
function ...
    [timeMatrix, freqMatrix, nfft, nWindows, times, freqs, accuracy] = constructSlices(obj)
function plotSlices(obj, varargin)
function ...
    varargout = amplitudePlot(obj, type, value, nfft, nWindows)

%Event Handling
function ...
    listenToExtTrigger(obj, myDevice, timePeriod)
function listenToSwTrigger(obj, myDevice)
function listenToDataSavings(obj, myDevice)
function respondToTriggered(obj, src, ¬)
function respondToDataSaving(obj, myDevice, ¬)

%File Handling
function reLoad(obj)
function LoadMostRecent(obj)
function LoadFile_Callback(obj,~,~)
function saveData(obj,myDevice)
function figureName=saveFig(obj,varargin)
function openInMyClass(obj,myDev)

%--------Short ...

functions---------------------------------------------------

%Position
function ...
    [coord,str,varargout]=position2Coordinates(obj,pos,varargin)
function ...
    value_i=coordinate2Position(obj,varargin)
function value=getCurrentCoordinate(obj)
function value_i=getCurrentPosition(obj)
function [iLower,iUpper]=getCurrentIndex(obj)

%Notation
function ...
    [str,exponent_value,preval]=value2ScientificNotation(...)
function ...
    str=getValuePlusMinusAccuracy(obj,value,type,accuracy)

%Indexes
function index=lim2ind(−,limit)
function limit=ind2lim(−,limit)

%Marks
function MarkPosition(obj,position,varargin)
function ClearMarks(obj)

%Constrain
function [x0, xend, ...]
    shifted=shiftPositionNearEdges(obj,x0,xend)
function rect=constrainRect(obj,rect,varargin)
function ...
    TF=isWithinLimits(−,hAxes,point,varargin)
The Code VIII

%Get Value
function [nfft,nWindows]=getNFFT(obj,N_data)
function fNumber=getFNumber(obj)
function mySignal=getSignal(obj,type)

%Test
function testConsistency(obj)
function testMyClass(obj)

%View
function cleanView(obj)
function normalView(obj)

%Image Colors
function fadeColors(obj)
function returnColors(obj)
function brighten(obj)
function darken(obj)

%Other
function printDebug(obj,str)
function stopTimer(obj,varargin)
end
methods(Static)
function a = example()
function ...
    [tickLabels,tickPlaces,counter]=FindMyTicks4(physicalLimits,indexLimits,exponentValue)
function myCompress(data)
function myCompress16(data,smallest_increment)
function S=myCompress8(data)
function data=unCompress8(S)
function ...
    [offset,varargout]=calibrate(signal,type,varargin)
end %Methods(Static)
end %Classdef