MARTe. Real time integration manual

Author : Llorenç Capellà
Company : Vitrociset Belgium

Task Plan/ ref IDM :

Document IDM reference :

Date : 20/03/2016

Contents

1 - Introduction

This document is intended to be a manual for the installation and the usage of MARTe for the real time integration tests.
Moreover the current document includes the steps of the real time integration process detailing the inputs, outputs and parameters of each GAM.

2 - Software installation

The whole software is distributed over three major components:
1. MARTe running the real time processing;
2. MDS+ providing the data archiving facility;
3. An Enclustra FPGA board providing the interface between the magnetics boards under test and MARTe.

The software must be installed on a PC with at least 4 CPU core running CentOS 7.

2.0 MARTe pre-installation

Before installing the software move to the directory where the software will be installed and create the following environmental variable:

```
export INSTALLATION_DIR=$(pwd)
```

Install the following extra packages. The next commands have to be executed as root

```
yum install gcc-objc++.x86_64
yum install ncurses-devel
yum install epel-release
yum install octave
yum install gtk2-devel
yum install svn
yum install git
yum install tuna
yum install xterm
yum install python-devel
```

2.1 - MARTe

To install MARTe move to the INSTALLATION_DIR

```
cd $INSTALLATION_DIR
```

Then checkout the repository to your computer

```
svn co http://efda-marte.ipfn.ist.utl.pt/svn/EFDA-MARTe
```

Compile MARTe:

```
cd EFDA-MARTe/trunk/
./compile.MARTe linux config.MARTe
```

2.2 - Magnetics diagnostic interface

The magnetics diagnostic interface stores the data on a special structure called “tree” provided by MDSplus.

2.2.1 - MDSplus

In www.mdsplus.org there is a yum repository for the MDSplus. Go to http://www.mdsplus.org/index.php/Latest_RPM%27s_and_Yum_repositories and choose your Linux distribution. For this particular tutorial the mdsplus-alpha-repo-7.0-96.el6.noarch.rpm is downloaded. Move where the file is saved and install the yum repository (run the command as root)

```
yum install mdsplus-alpha-repo-7.0-96.el6.noarch.rpm
```

Now install the required packages.

```
yum install mdsplus-alpha-kernel mdsplus-alpha-devel mdsplus-alpha-
java
```

When yum asks to install dependencies say yes.

2.2.2 - Real time integration tool.

To install the real time interface tool, first move to the INSTALLATION_DIR

```
cd $INSTALLATION_DIR
```

Create the diagnostic folder and the git sub-folder.

```
mkdir diagnostics
cd diagnostics
mkdir git
cd git
```

Inside git folder copy the remote integration interface repository

```
git config --global http.sslverify false
git clone https://vcis-
gitlab.f4e.europa.eu/aneto/MagneticsIntegratorInterface.git
```

If the last command does not work use this one

```
git clone https://user@vcis-
gitlab.f4e.europa.eu/aneto/MagneticsIntegratorInterface.git
```

Where the user must be replace by a valid user name.

Change to develop branch

```
cd MagneticsIntegratorInterface
git checkout develop
```

Executes setUpEnvironment.sh

```
./setUpEnvironment.sh
```

The following extra tools must also be installed

```
yum install mdsplus-alpha-python
yum install python-matplotlib
easy_install pip
easy_install MDSplus
pip install fpdf
```

Figure 1 shows the structure of the MagneticsIntegratorInterface folder

Figure 1 Folder structure of the MagneticsIntegrationInterface.

2.2.3 - Network configuration

In order to communicate with the FPGA board, the network has to be properly configured.
Open the configuration network windows. Open "**All settig**"--> "**Networks**"-->"**Wired**".

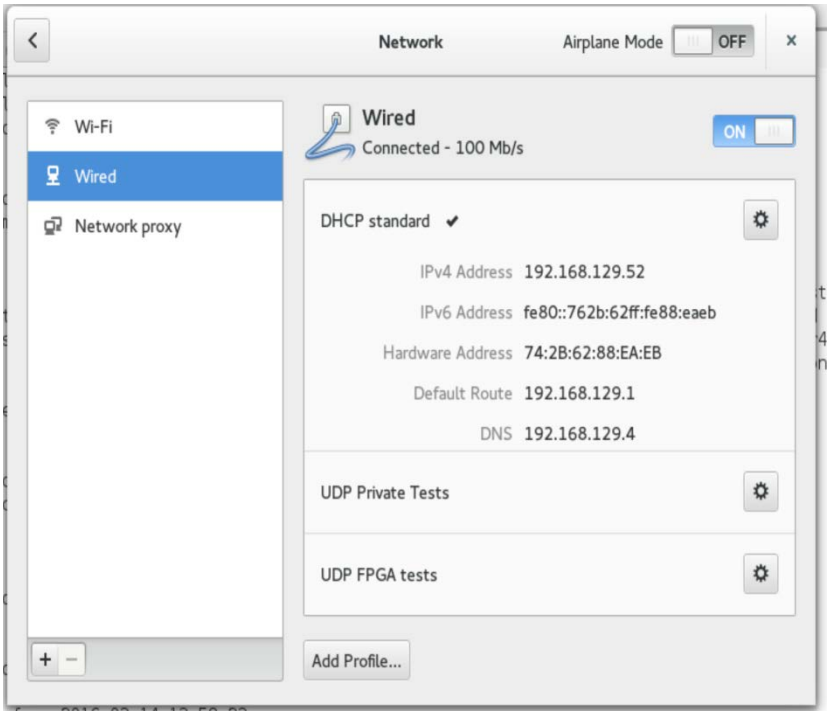

Figure 2 Configuration of the net connection. Adding a profile

Then click "**Add Profiles"**. Choose the "**Identity"** label.
Fill the field **Name** (e.i UPD FPGA tests) and the field **MUT** with 8192.

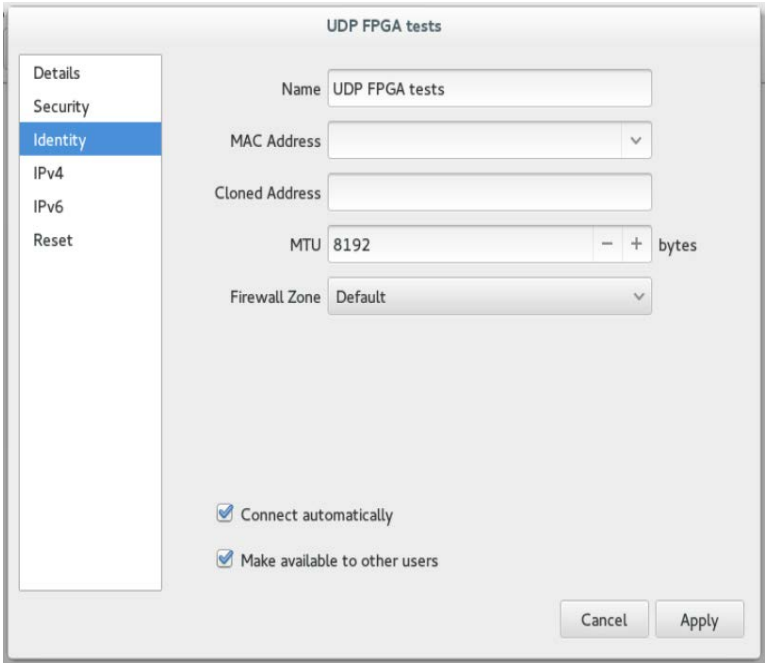

Figure 3 Configuration of the net connection. Identity label

On the IPv4 fill the **address** with 10.197.101.3, the **net mask** 255.255.255.0 and **gateway** 10.197.101.254 and click apply.

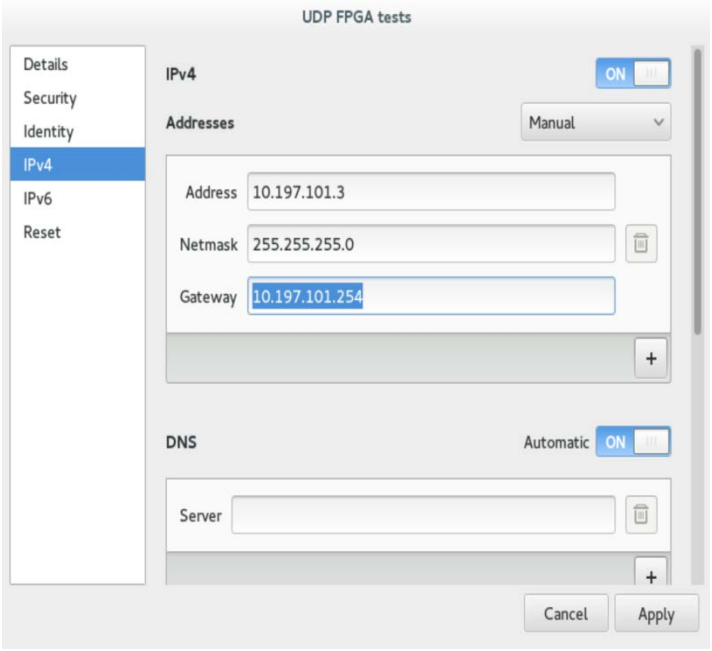

Figure 4 Configuration of the net connection. IPv4

3 – Real time integration GAMs

The real time integrator takes the UDP raw data as an input, processes it and generates the integral as an output. Moreover, additional functions are added, such as status and channel status monitoring and temperature acquisition.
The integrator is split in 8 processes. Each unit is implemented with a GAM. The names and a brief explanation of each GAM are listed below:

- StatusMonitoring: When AcquisitionStarted is 1 (data acquisition has started) this GAM checks some general alarms/information bits and when any of them changes the Status is saved on MDSplus tree.
- ChannelStatusMonitoring: When AcquisitionStarted is 1 this GAM checks some ADC alarms/information and when any of them changes the alarms are saved on the MDSplus tree.
- UnpackGAM: Takes the packed raw data which contains the chopper state. It generates two outputs: one with the raw acquired data and one with the chopper state information.
- EOCorrectionGAM: Calibrates the EO for a configurable period of time and then subtracts it from the raw data.
- DemodulationGAM: Takes the output data from the EOCorrectionGAM and the chopper state data and inverts when it is necessary. The DemodulationGAM also checks the correctness of the chopper state data.
- InterpolationGAM: Detects when a transitions occurs and interpolates the data around this point.
- IntegrationGAM: Calibrates the WO and integrates (trapezoidal algorithm) the output of the InterpolationGAM. The output is in V·s.
- TemperatureGAM. Reads the temperature in ADC units and converts them in C. The first temperature is taken as a reference and is compared with the other temperature values. When the temperature is beyond some boundaries the GAM throws a warning to JTLogger.

Note that one instance of each GAM is allocated for each channel, so that there will be four UnpackGAMs, four EOCorrectionGAMs, …

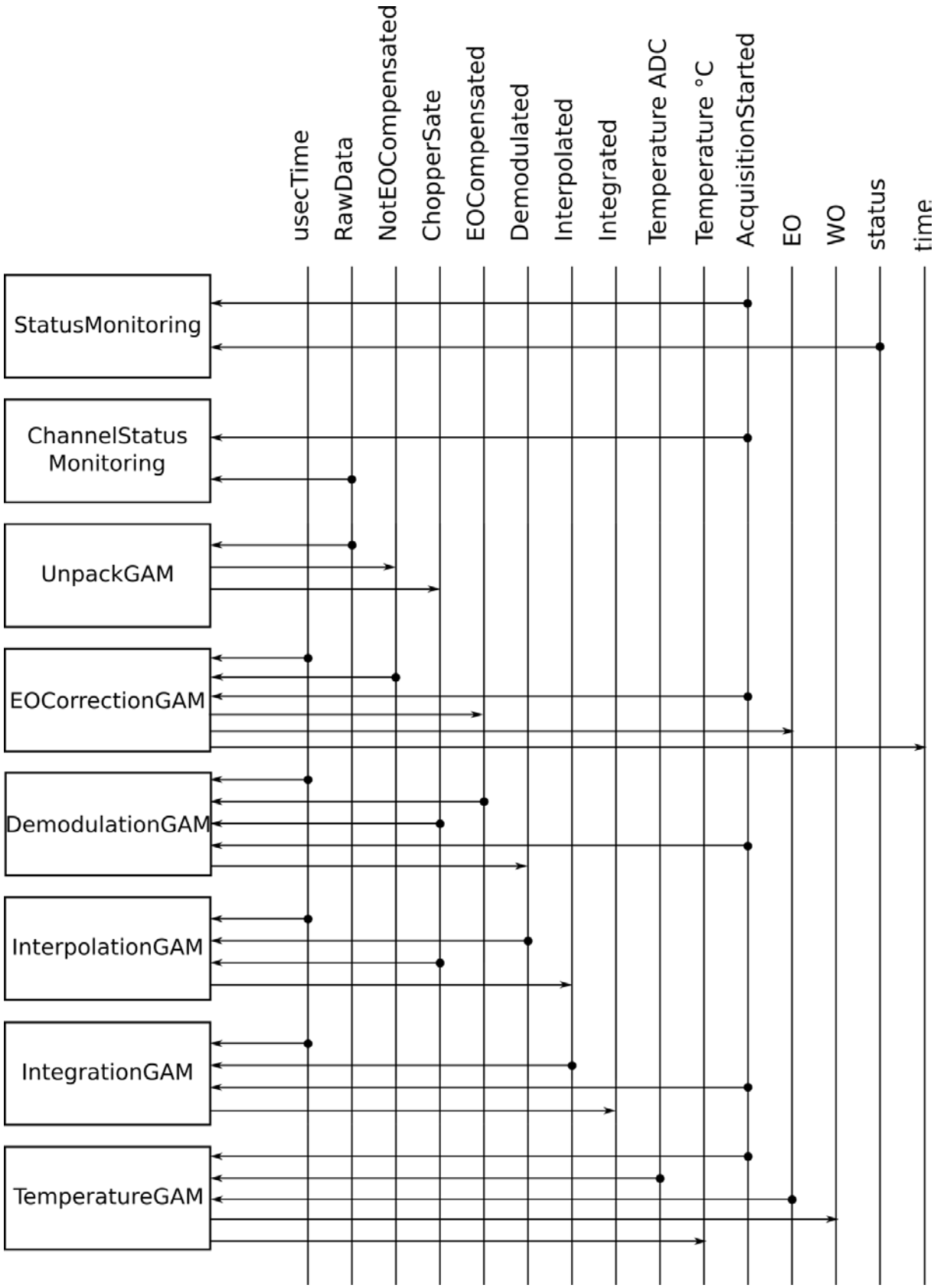

Figure 5 Steps of the real time integration

3.1 Components

3.1.1 StatusMonitoring

The StatusMonitoring checks the Status bits when the acquisitionStarted is 1. If some alarm is set to 1 the StatusMonitoring save the status bits on the MDSplus tree and send an error message to the logger. The status bits are shown below

Status	Acquisition Started? (1 bit)
	FPGA Clock Locked (1 bit)
	FIFO full (5 bits)
	AXI FIFO full (1 bit)
	Spare Bits (24)

The GAM has only one parameter:

- MDSPlusWriterLocation: Indicates where the Status has to be saved

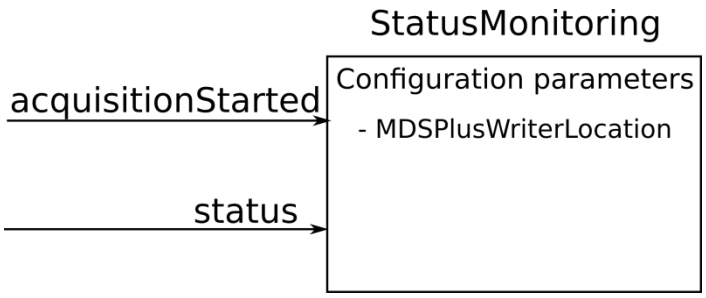

Figure 6 Status monitoring GAM

3.1.2 ChannelStatusMonitoring

The ChannelStatusMonitoring checks the ADC information bits, added to the RawData, when the acquisitionStarted is 1. If some of these bits are set to 1 the ChannelStatusMonitoring saves the information bits on the MDSplus tree and sends an error message to the logger.
The ADC information bits are shown below:

ADC information	FPGA PLL loss of lock
	Jitter Attenuation PLL loss of lock
	Jitter attenuator PLL loss of sync
	ADC digital filter out of range
	ADC analog input out of range
	ADC pattern not synced

The GAM has only one parameter:

- MDSPlusWriterLocation: Indicates where the ADC information has to be saved

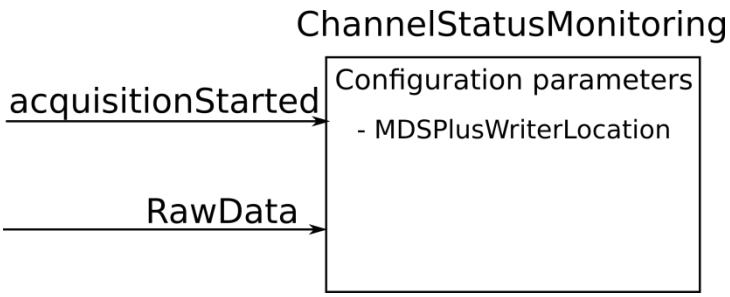

Figure 7 Channel status monitoring GAM

3.1.3 UnpackGAM

The UnpackGAM takes an array of packed raw data, which contains the data from the ADC and the chopper state. It generates two outputs: one is the raw data called NotEOCompensated and the other is the chopper state called ChopperState. Both formats are int32.
This GAM has one configuration parameters:

- ADCbits. Indicates the number of bits of the ADC used

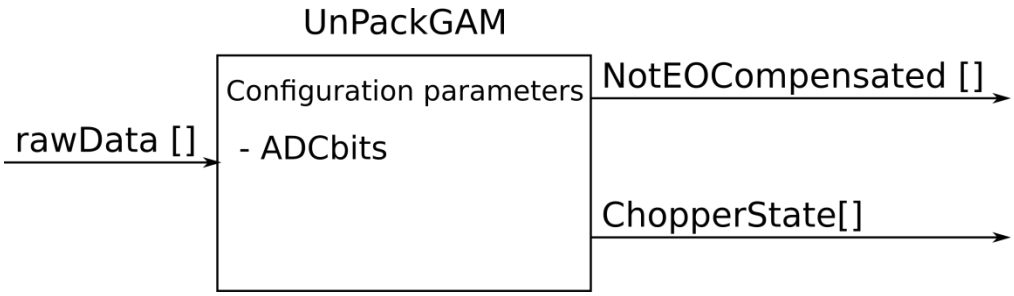

Figure 8 Unpack GAM

3.1.4 EOCorrectionGAM

The EOCorrectionGAM has two “different states”.

- First the GAM calibrates the EO. During this period the output (EOCompensated) is always 0.
- Then, when the EO is calibrated, the GAM subtracts the EO from the NotEOCompensated signal.

Moreover the EOCorretion implements an internal “Reset” function which will reset the EO value when usecTime is 0. This allows running the GAM even when there is no input rawData and then, when data and time are being produced, the GAM resets itself. Based on the usecTime (which is a counter) and the sampling frequency, this GAM provides time in seconds.
The GAM has five configuration parameters:

- FreqSamp. It is the sampling frequency of the ADC producing the NOTEOCompensated data..
- EOTime. Is the time length during which the EO value is estimated.
- EnableEO. Indicates if the EO will be calculated or not (if not EOCompensated will have the same value of NotEOCompensated)..

- MDSPlusWriterLocation. Inticates which interface will be used to save the EO value. Usually this value will be "MARTe.DriverPool.MDSWriter"
- Channel number. Indicates in channel the EO value will be saved. E.g., if channelNumber = 1 the EO value will be saved in the CH1:EO node.

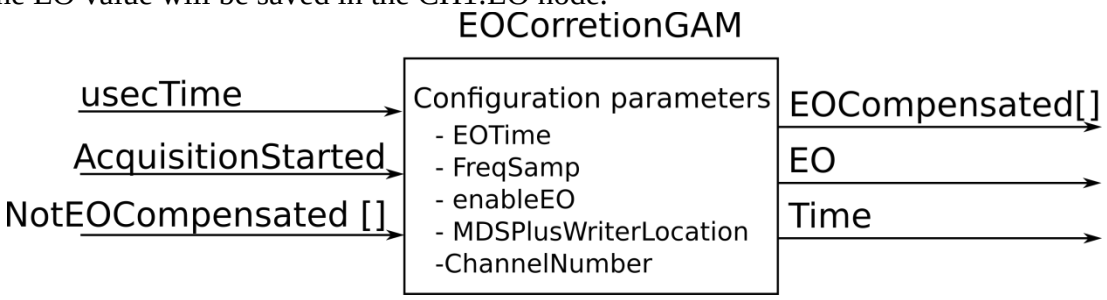

Figure 9 EO correction GAM

3.1.5 DemodulationGAM

Using a square or sinusoidal demodulation, the DemodulationGAM demodulates the EOCompensated. This GAM also has the function reset, which sets to zero all the internal counters and variables.

Moreover this GAM checks the correctness of the chopper state (only if the sample frequency/chopper frequency is an integer and EnableControlChopper = 1) checking the number of ones and zeros every N samples, where N is the ratio between the sampling frequency and the chopper frequency , i.e. 2000 in the case of SampFreq = 2000000 samp/s and FreqChopper = 1 kHz.

The configuration parameters are:

- FreqChopping. Is the chopping frequency.
- FreqSamp. Is the sampling frequency.
- EnableControlChopper. Enables and disables the chopper checking.
- TypeDemodulation. There are two types of demodulation. Square demodulation where the data is inverted when the chopper state is 0. And Sinusoidal demodulation where the data is multiplied by a sine wave.
- FrequencyDemodulation. When TypeDemodulation = “Sinusoidal” the FrequencyDemodulation indicates the frequency of the demodulation sine.

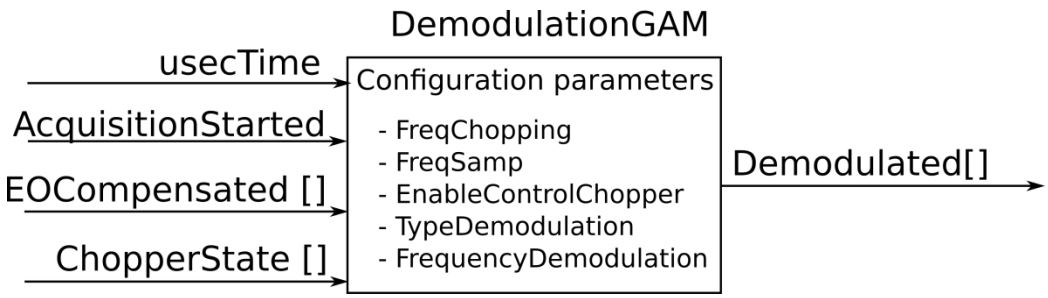

Figure 10 Demodulation GAM

3.1.6 InterpolationGAM.

The Interpolation GAM copies the input data to the output until it detects a change in the chopperState, upon which it uses an algorithm to reconstruct ***n*** samples (where ***n*** is a configurable parameter).
There are 4 algorithms implemented:

- Hold 0. In the transitions this algorithm substitutes n samples by 0.
- Do nothing (No interpolation). When this options is chosen the reconstruction algorithm does nothing, i.e. it uses the original samples.
- Hold the last value. When a change is detected the algorithm holds the last sample during n samples.
- Linear interpolation. When there is a transition the algorithm saves the last value, looks forward to the n+1 samples and using these two points interpolates linearly **n** samples.

Due to the linear interpolation this GAM delays the output ***n***+1 samples, where ***n*** is the number of samples to interpolate.
The configuration parameters are:

- TypeInterpolation. -1 ☐ hold 0, 0 ☐do noth interpolation.
- NumberOfSamplesInterpolated. Is the number of samples to interpolate.

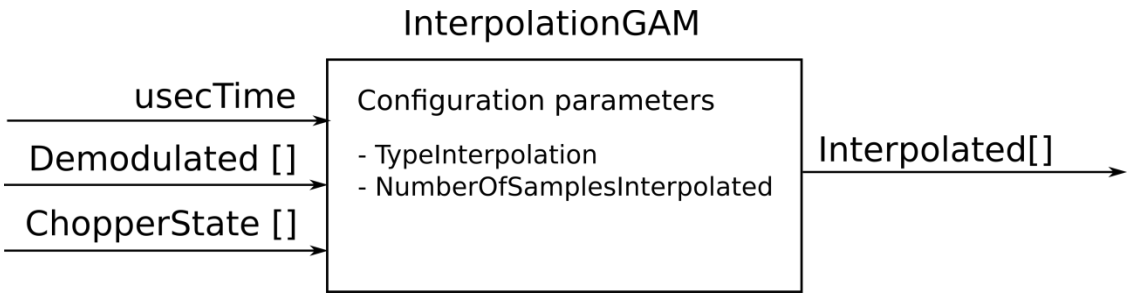

Figure 11 Interpolation GAM

3.1.7 IntegrationGAM

The IntegrationGAM has “three different states”:

- First the GAM waits for the EO calibration to be computed and forces the output to 0. Following this period, the IntegrationGAM waits one extra second (arbitrary value expected to be much larger than the interpolation delay) before starting to compute the WO calibration. This allows taking into account the delay of the signal in the InterpolationGAM, since otherwise the IntegrationGAM would be calibrating the WO with some zero values.
- Second, the GAM calibrates the WO. During this period the output is the integral without WO correction.
- Then, when the WO is calibrated, the output of the GAM is the integrated data WO corrected. The integration uses the trapezoidal integration algorithm.

The output of the GAM is in V·s. Internally the ADC units are converted to volts.
The calibration parameters are:

- SampFreq. Sample frequency of the ADC.
- EOTime. Time needed to calculate the EO.
- WOTime. Time needed to calculate the WO.

- Gain. This is the gain from the input of the module to the input of the ADC.
- InputRangeADC. Dynamic range of the ADC.
- ADCbits. Number of bits of the ADC.
- enableWO. Allows enabling and disabling the WO calibration. 1 ☐ WO calibr
  calibration (WO = 0).
- MDSPlusWriterLocation. Inticates which interface will be used to save the WO value. Usually this value will be "MARTe.DriverPool.MDSWriter"
- ChannelNumber. Indicates in which channel the WO value will be saved. E.g. if channelNumber = 1 the EO value will be saved in the CH1:EO node.

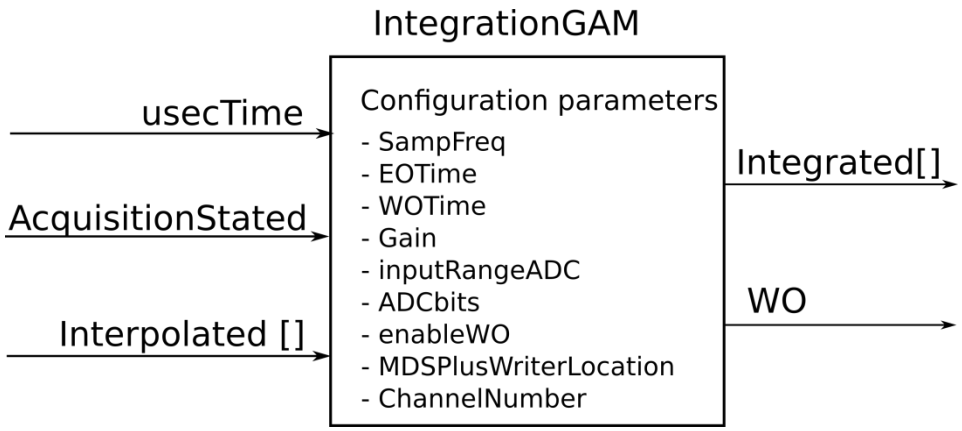

Figure 12 Integration GAM

4 - User's Manual

4.1 - Using MARTe for the experiments

Connect the Ethernet cable from the PC to the FPGA board (make sure the board is powered). Choose the internet connection configured in "2.2.3 - Net configuration".

From the command line execute the $INSTALLATION_DIR/diagnostics/git/runMARTe.sh as root. This will launch the configuration tool for MARTe (Figure 13), choose the desired options and then click OK. After that, the test form tool should appear (Figure 14). At this point MARTe is running and waiting for the start acquisition trigger. Pushing the start button of the FPGA board MARTe will trigger the start of data saving in MDSplus.

In the test form tool, a set of fields can be filled (such as notes, temperatures and author). Files can be load as well. Moreover, it has a countdown which stops MARTe when it arrives to 0. All the data is saved in the Tree when the OK button is clicked.

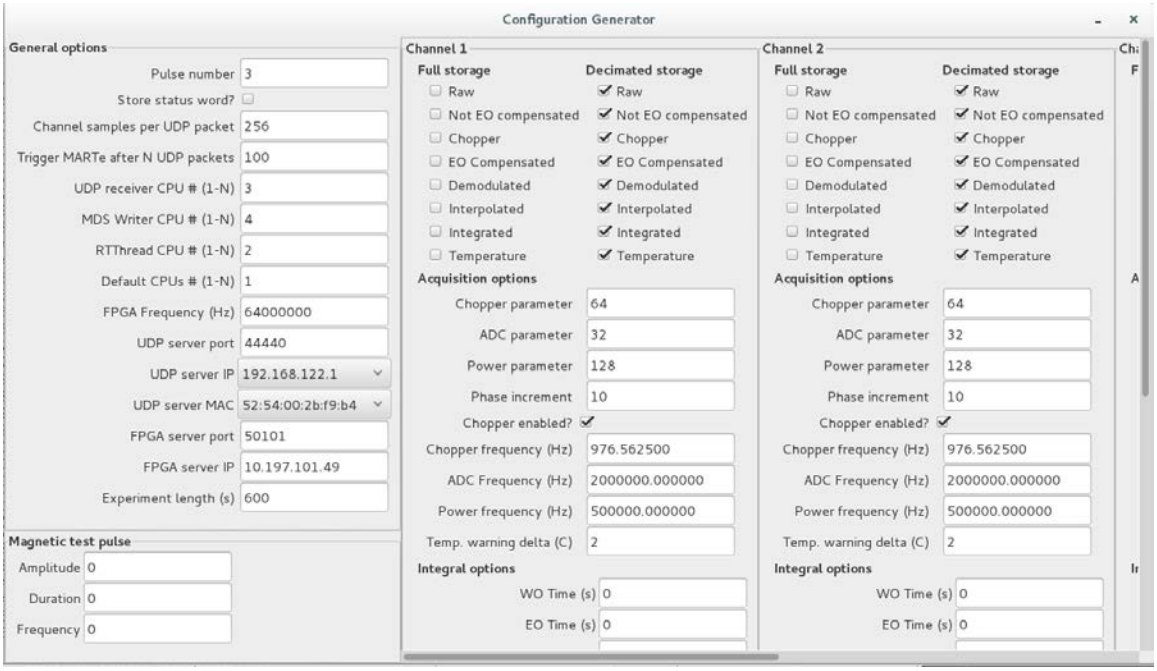

Figure 13 MARTe configuration window

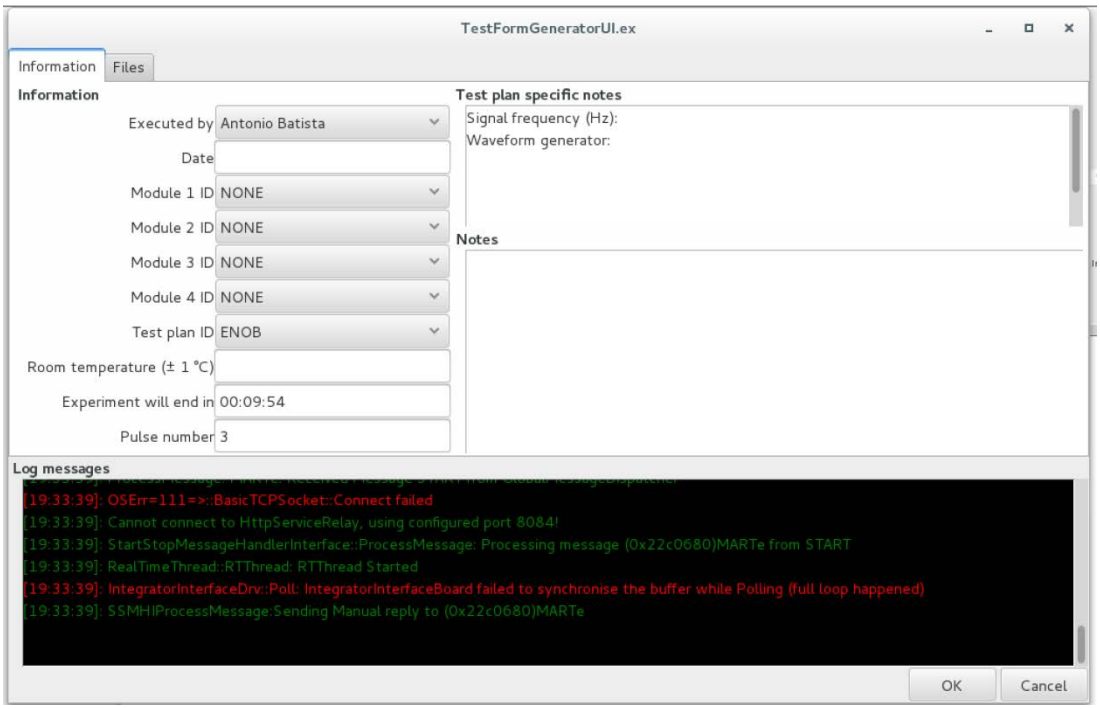

Figure 14 Test form tool window

4.2 Extra tools

4.2.1 Plotting tools

There are two ways of plotting the data: the jScope and the MARTe web service.

jScope

For plotting decimated data, when the experiment is running, move to the $INSTALLATION_DIR/diagnostics/git/MagneticsIntegratorInterface/ and run launchRealTimeJScope.sh

```
cd $INSTALLATION_DIR/diagnostics/git/MagneticsIntegratorInterface/
./launchRealTimeJScope.sh
```

It launches a predefined plot window (Figure 15) with 4 subplots. To change the signal, right click on the plot and choose "**select data source**", choose the desired decimated signal and the pulse number.

The plot window waits for an event, which is send by MARTe when is acquiring data. When an event is received, the plot window is refreshed with last ten seconds of data. The frequency of the event is configured in MARTe and by default is 5 seconds.
Note that this requires that the user has enabled the storage of decimated data in the configuration tool (Figure 13).

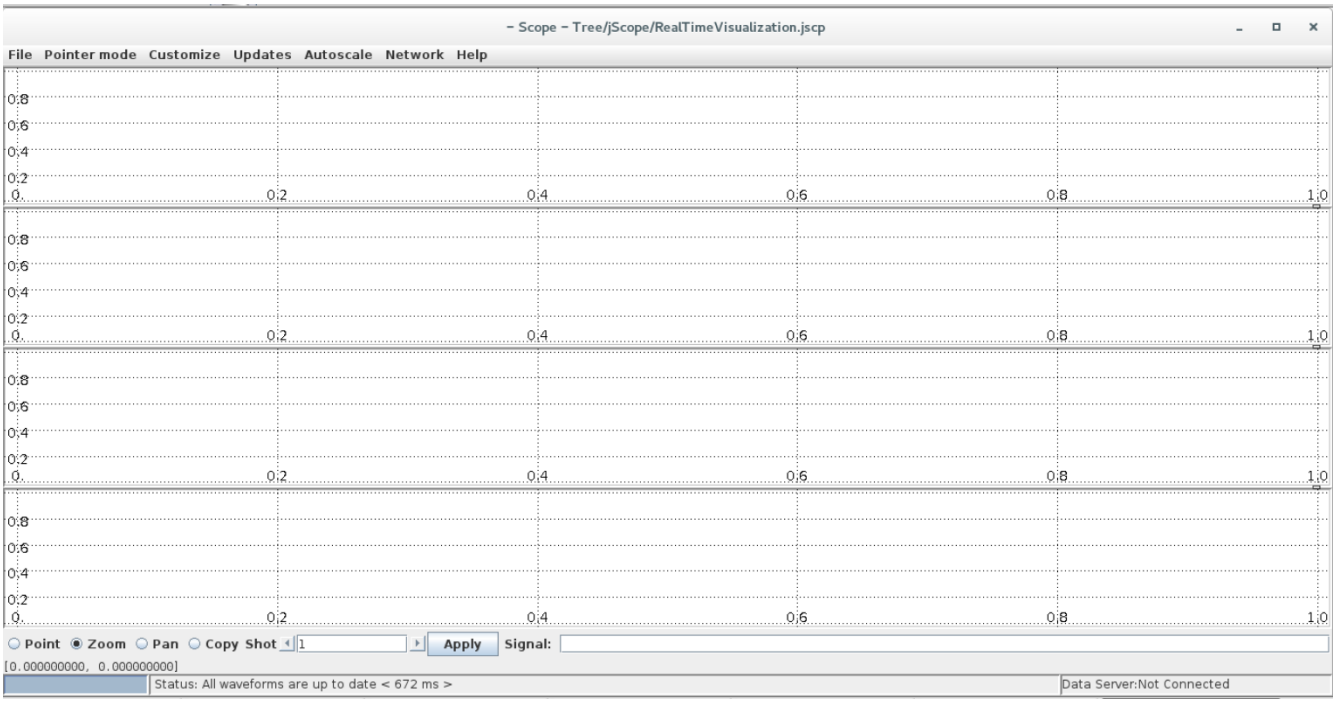

Figure 15 jScope window example.

To plot the full data **after** an experiment, use the launchJScope.sh script. It launches a predefined window. To select the signal, right click on the plot "**select data source**", choose the desired signal and the pulse number.

```
cd $INSTALLATION_DIR/diagnostics/git/MagneticsIntegratorInterface/
./launchJScope.sh
```

The data shown on the plot is automatically decimated by MDSplus , however when zooming, the plot window will automatically increase the time resolution (adding more points and reducing the decimation).

Beware on the size of the signal to be visualized using the launchJScope.sh given that this process requires huge amounts of RAM for large size signals. For short experiments this method is advisable, however is not recommended for long experiments and FULL data visualization.

Web MARTe plotting

When MARTe is running go to the browser and write the following URL:
http://localhost:8084/INTEGRATOR_PLOTS
From the menu, choose the desired plot.
Note that this service is only available when MARTe is running and the decimation is very high (it depends on the MARTe cycle), consequently only low frequencies can be visualized.

4.2.2 Extract data from MDSplus

After an experiment, for further analysis, the data can be extracted from MDSplus in text or binary format.

Extract to binary file

To extract data to a binary file move to INSTALLATION_DIR/diagnostics/git/MagneticsIntegratorInterface and executes convertMDStoBin.sh specifying the path of the tree, the name of the tree, the pulse number, the signal to extract and the path of the output file (optional).

```
cd $INSTALLATION_DIR/diagnostics/git/MagneticsIntegratorInterface
./convertMDStoBin.sh Tree marte_mds 1 CH1:FULL:RAW_DATA .
```

If the output path is not specified the file is saved in Components/ConvertMDStoBin/

Extract to text file

To extract data in a text file move to INSTALLATION_DIR/diagnostics/git/MagneticsIntegratorInterface and executes convertMDStoText.sh specifying the path of the tree, the name of the tree, the pulse number, the signal to extract, the path of the output file (optional), initial time (optional) and the final time (optional).

```
cd $INSTALLATION_DIR/diagnostics/git/MagneticsIntegratorInterface
./convertMDStoText.sh Tree marte_mds 1 CH1:FULL:RAW_DATA . 0.001
1.23
```

If the time interval is not specified all data is converted.

Extract using python

The following example shows how data can be extracted using python: Note that the following environment variable must be exported before executing the python script:

```
export
marte_mds_path=$INSTALLATION_DIR/diagnostics/git/MagneticsIntegrator
Interface/Tree
```

```
#!/usr/bin/python

import os
from MDSplus import *
import matplotlib.pyplot as plt

mdsTreeName = 'marte_mds'
mdsPulseNumber = 34
try:
    tree = Tree(mdsTreeName, mdsPulseNumber)
except:e
    print 'Failed opening ' + mdsTreeName + ' for pulse number ' +
mdsPulseNumber

channelNumber = 1
signalType = 'NOT_EO_COMP'
#signalType = 'RAW_DATA'
signalName = 'CH' + str(channelNumber) + ':FULL:' + signalType
try:
    node = tree.getNode(signalName)
    print node
    #Typically there will thousands of segments (each with some
milliseconds of data)
    nSegments = node.getNumSegments()
    print nSegments
    segment = 0
    while segment < nSegments:e
        time = node.getSegment(segment).getDimensionAt(0).data()
        data = node.getSegment(segment).getValue().data()
        plt.clf()
        plt.plot(time[0:len(data):1], data)
        plt.title(signalName)
        plt.show()
        segment += 1
except:
    print 'No data available for signal ' + signalName + ' in pulse
number ' + str(mdsPulseNumber)
```

4.2.3 Replaying an experiment offline with different GAM parameters

The software infrastructure allows replay an experiment using as a data source the values which were stored in a previous experiment. This can be very helpful to easily test different integration algorithms and/or different GAM parameters (e.g. number of samples to interpolate).

Infrastructure to be explained in the next release of the manual.

4.2.4 Test form generator and summary tools.

Another useful tool is the runTestFormGenerator which automatically creates a test form using the information of the tree.

To run the program type on the line command

```
cd $INSTALLATION_DIR/diagnostics/git/MagneticsIntegratorInterface
./ runTestFormGenerator.sh pulseNumber
```

First, the test form generator window will appear and the user is allowed to add or modify information stored during the execution of the experiment. Clicking OK the report is generated and is saved in the $HOME directory.

The *listAllExperimentsScreen.sh* script looks on the Tree folder and lists on the screen the most relevant information of the experiments providing a quick overview of all the experiments.

The *listAllExperimentsCSV.sh* gives the more information than the listAllExperimentsScreen.sh, and the information is saved on a CSV file. The script needs an output file name which is saved on the directory $INSTALLATION_DIR/diagnostics/git/MagneticsIntegratorInterface /Components/TestFormGenerator/

4.3 - Golden rules

- Execute runMARTe.sh as **root** mode. This script needs root privileges to isolate cpus.
- When MARTe is running **never** execute launchJScope.sh. Even with the cpus isolated the RAM resources are shared.
- When the real experiments start (ENOB tests, drift test and so on) **do not remove shotid.sys** (usually in Tree/) otherwise MARTe will start counting from 1 removing/mixing the existing data.
- When the local disk is full, the whole pulse experiments files (which are three files with the extensions *.characteristics*, *.datafile* and *.tree*) can be copied to an external disk and can be removed from the local disk. However the **shotid.sys cannot be removed** from the local disk otherwise MARTe will start counting pulses from 1.