August 12 2016

Online Monitoring Software

Lawrence Berkeley National Laboratory

Introduction

Online monitoring software for testing commercial electrical components in the ITS design. Its purpose it to view a live plot of the voltage output of ADCs during test beam. This lets us see immediately when hardware fails during test beam and allows for some form of check with the deeper-probing offline analysis. The code has been slightly modified for the upcoming test beam in September 2016. Issues with the plotting and mixing of firmware data were fixed. This document describes my limited understanding of the Monitoring Software.

Classes

The following subsections will briefly describe the classes of the monitoring software that process information from the firmware, as well as any edits made to them. The classes that make up the GUI were not worked on. Important or frequently used variables are in green. Edits are in red.

I. Masks.h

Defines several functions for determining what kind of firmware word is being read by looking at specific bits. The data structure of firmware words is in DataFormatITS.docx Masks.cxx is only 1 line simply sharp includes Masks.h

1. **IS_FWREG(w)** Determines if firmware word w is a firmware register word by checking that the 28th and 24th bit are both 0.
2. **IS_HEADW(w)** Checks if w is a header word by checking that the 28th bit is 0 and the 24th bit is 1.
3. **IS_PAYLW(w)** Determines if w is a Payload word.
4. **IS_TRAIL(w)** Determines if word is a trailer word
5. **IS_ADC...(w)** Checks if it is an ADC Header, or is ADC0 (number of words is 12) or ADC1 (number of words is 12*1000).
6. **IS_COP_" "_HEAD(w)** Determines what composite header the word contains according to the corresponding hexadecimal. EDIT: Added comments to the code describing the operation each composite header is associated with.
7. **TSTAMP(w)** Get timsestamp from firmware

II. PayloadADC.cxx

1. Calls the functions ("IS_HEADW" etc....) from Masks.h to see if data can be read correctly.
2. Uses vector fADC[n] (not to be confused with function "fADC()") as a buffer for payload data. The data is organized according to ADC sample and bit structure (ADC3_low,ADC2_low,ADC1,ADC0) described in the DataFormatITS.docx document. Another buffer variable, d, was mixing data between samples before fADC was filled. EDIT: Added conditional statements to prevent mixing (Line 98)

3. voltages are eventually determined in PayloadADC::GetADCf(), and then a vector of doubles, "v" is filled with voltages in PayloadADC::GetADCvf()
4. calls TSTAMP from Masks.h

III. Decoder

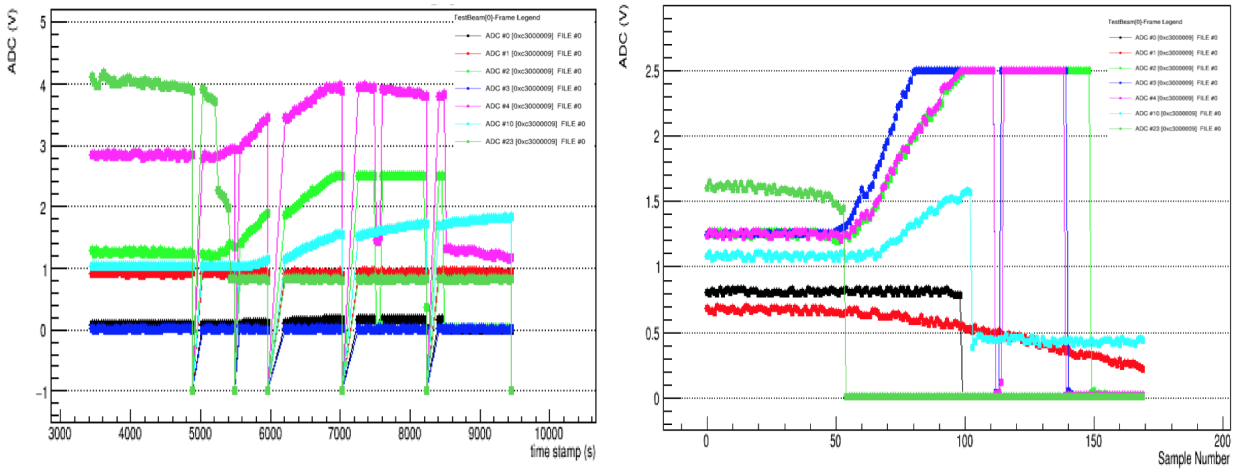
Decoder.h sharp includes PayloadADC (has access to voltages determined from payload words as well as timestamps). This particular class is quite long but much of it is just the same code under different conditional statements checking for different composite headers.

1. Fills vector fY[] with voltages
2. Fills vector fX[] with either timestamp or DAC Voltage depending on composite header word called.
3. gets a graph according to which composite header is called
4. EDIT: Decoder::InsertMarker() was making the graphs go to -1 at the end of each small run, disrupting a clean plot. This was commented out for Decoder::OnlineMonitoringADCRO, one of several functions that calls InsertMarker. (Line 543)
5. EDIT: The was also an issue with points not being connected well between small beam runs. A simple change from plotting time stamp to sample number was made (Line 488) Variable XMarker was made a global variable in Decoder.cxx (Line 11
6. Calls BinaryFileMonitor::Update() to update from the class FileMonitor.

IV. FileMonitor

A lot of time wasn't directed here. Class defines functions to read binary (or text) files from the firmware through fStream and passes it onto buffers so that classes like PayloadADC can interpret data.

Before & After



(a) Before

(b) After

ADC Scan plot before and after edits

Final Steps

The code needs to run live on a binary file that gets updated. I tried to get it to update ADCscans but the plot doesn't seem to update, just shows the same plot. The shell script test_update.sh works with AmplifierDC Scan (Composite header AF000002) but ADC scans (C3000009) seemed to loop back to the same graph. This was tested by duplicating the entire binary file, however, and not with new binary data, so it's very likely the plot is just looping. The software doesn't crash, and seems to update (the GUI has an ongoing timer). As a result, I think the ADC live test will work on a binary file that is updated in a way that the software would expect (updated from the firmware), rather than rapid duplication of the entire file.

EDIT DOCUMENT