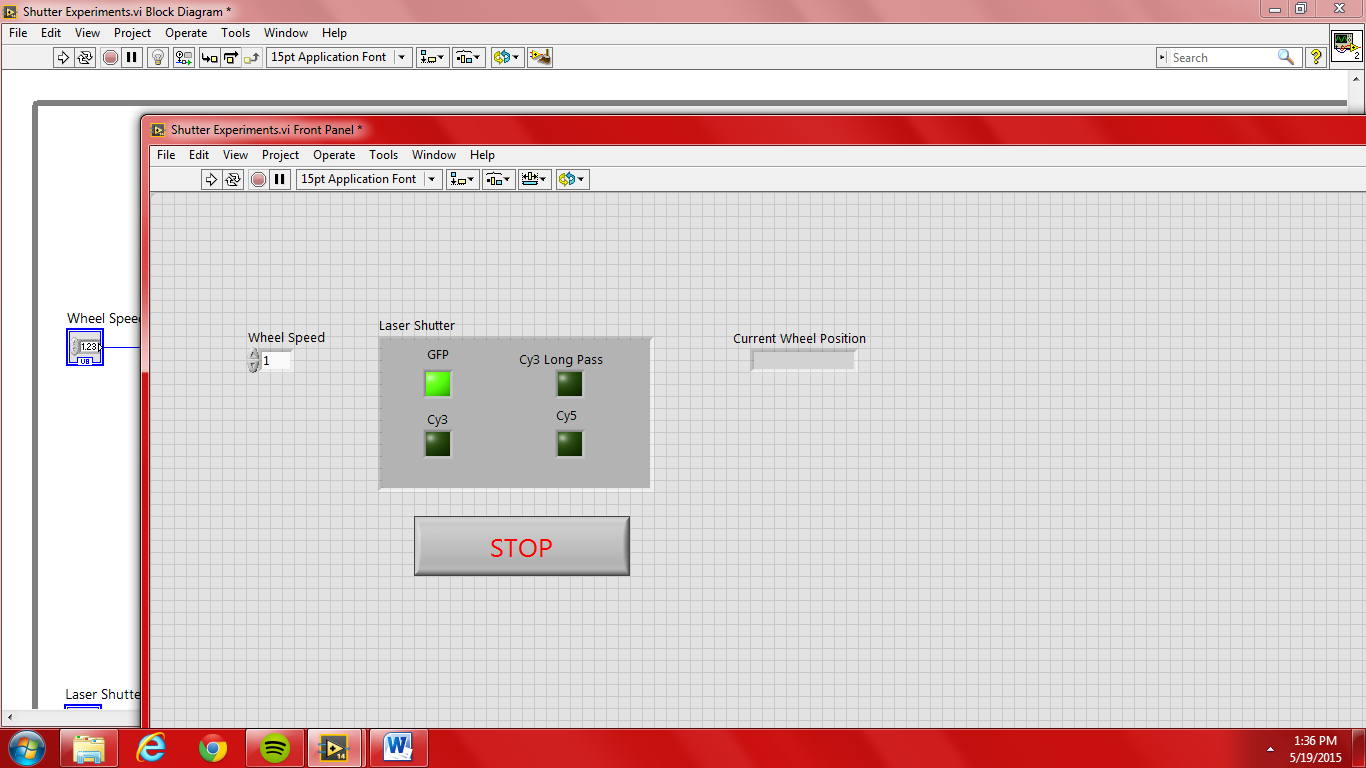
**Shutter Control.vi**

SubVIs:

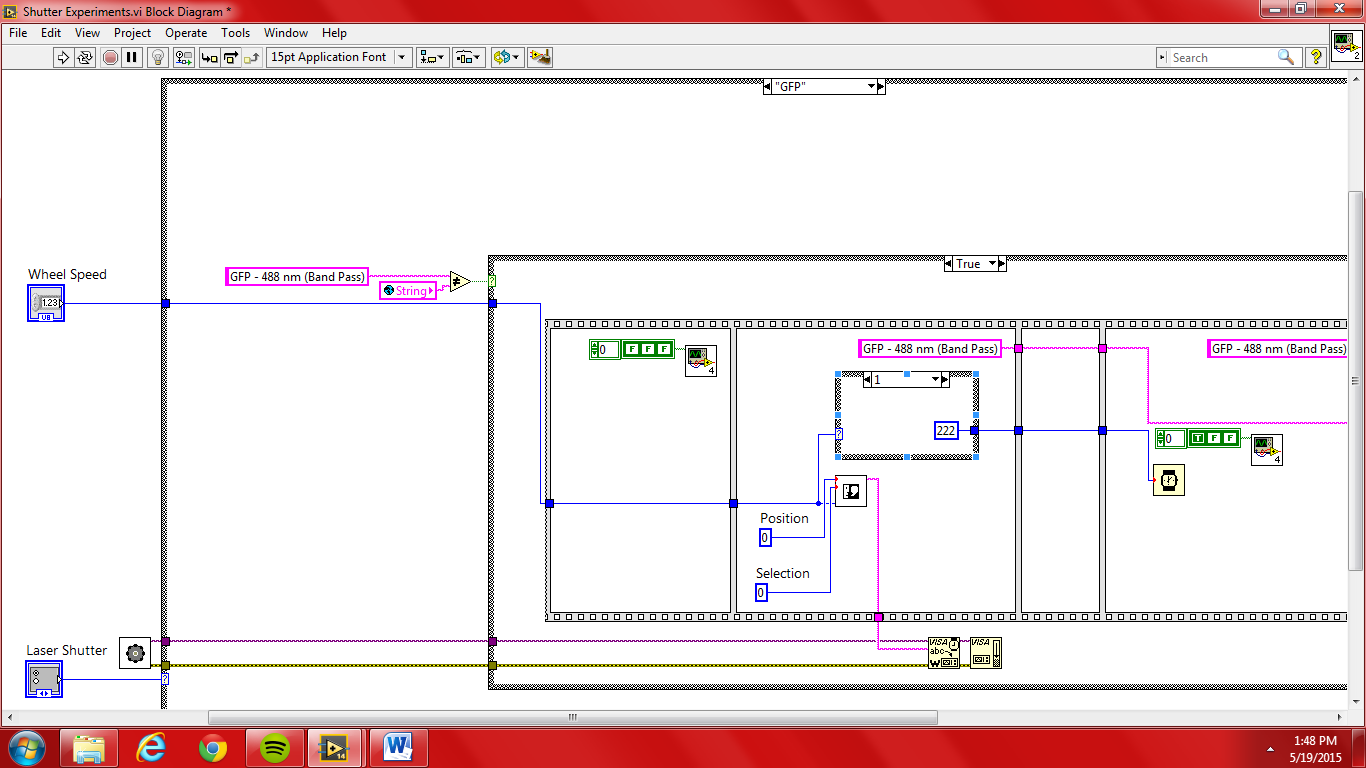
1. Laser Daq - SubVI
   1. Turns on/off shutters when selection is made
2. Generate-Command-SubVI
   1. Does something…
3. Global Shutter Variables - SubVI
   1. Global string variable indicating the current shutter
4. Initialize-FilterWheel-SubVI
   1. Initializes the filter wheel to be used

This vi controls/activates the laser shutters.



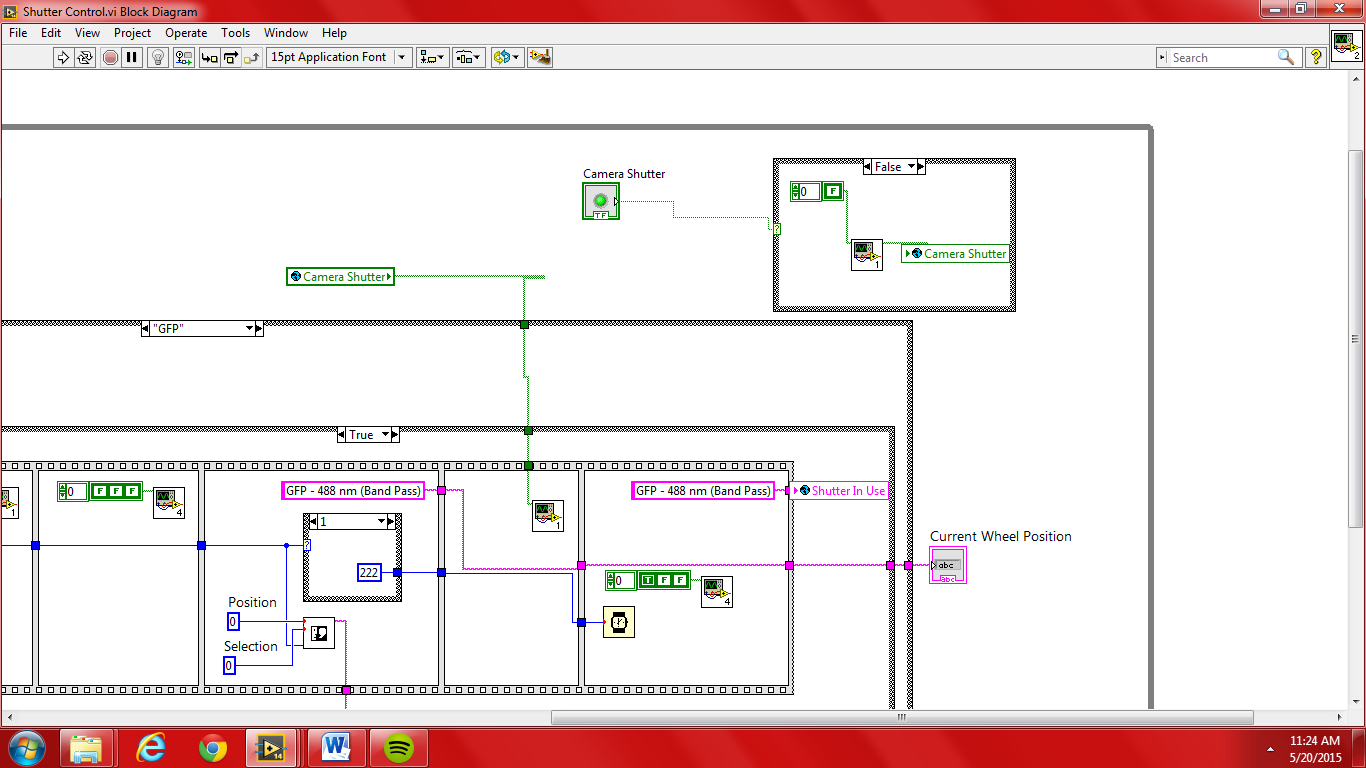
In the above figure:

* + Wheel speed: controls the speed of the wheel which in turn determines the delay when a different shutter is activated. Fast means low numeric values in the selection.
  + Current Wheel Position: The current wheel position
  + Laser Shutter
    - GFP: 488nm Band Pass
    - Cy3: 532 nm Notch Filter
    - Cy3: 532 nm Long Pass
    - Cy5: 637 nm Notch Filter



In the above figure:

* + The wheel speed determined by the user inputs into the case structure.
  + The laser shutter control determines the specific case based on the selection.
  + At each loop, the program checks to see if the user has made a different selection. Else, no signal is output to the filterwheel.

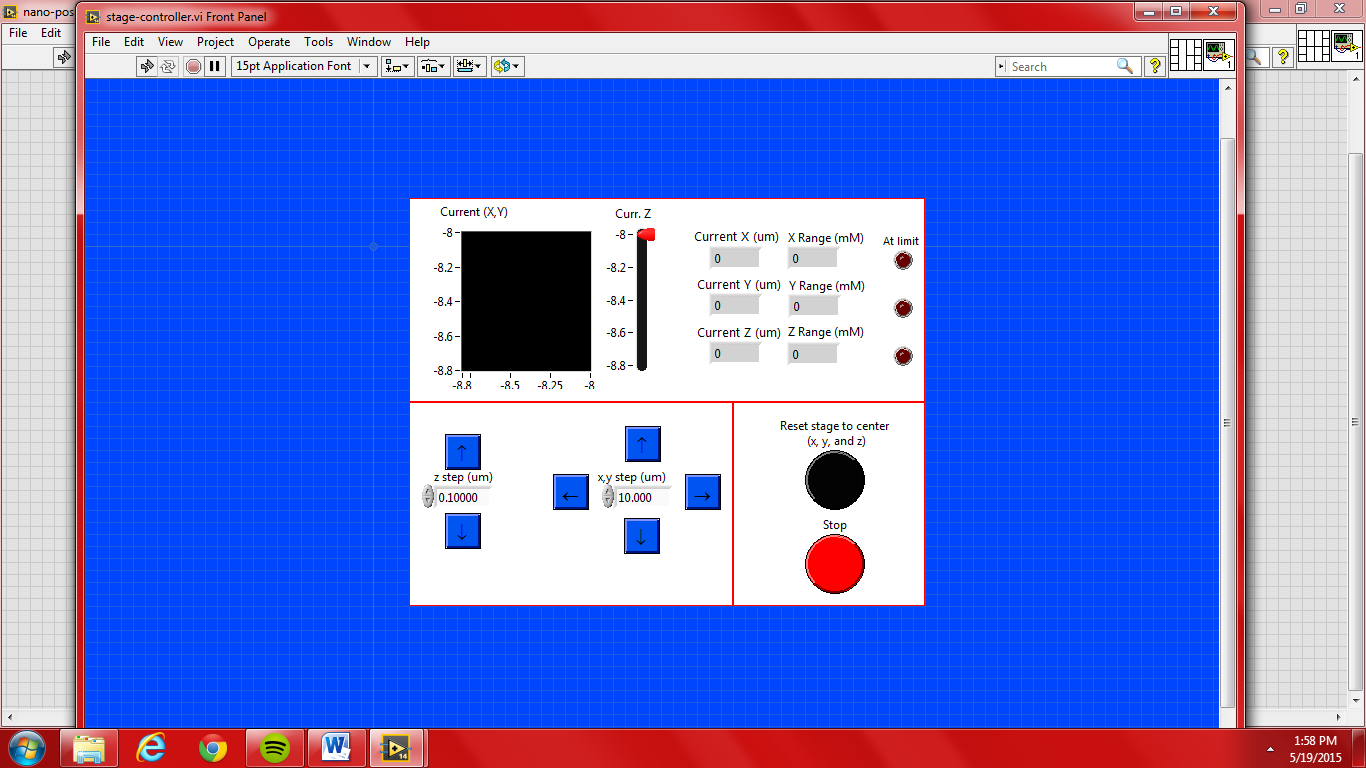


In the above figure:

The Boolean Camera Shutter activates/inactivates the camera shutter. This action sets the global variable Camera Shutter within the case structure accordingly.

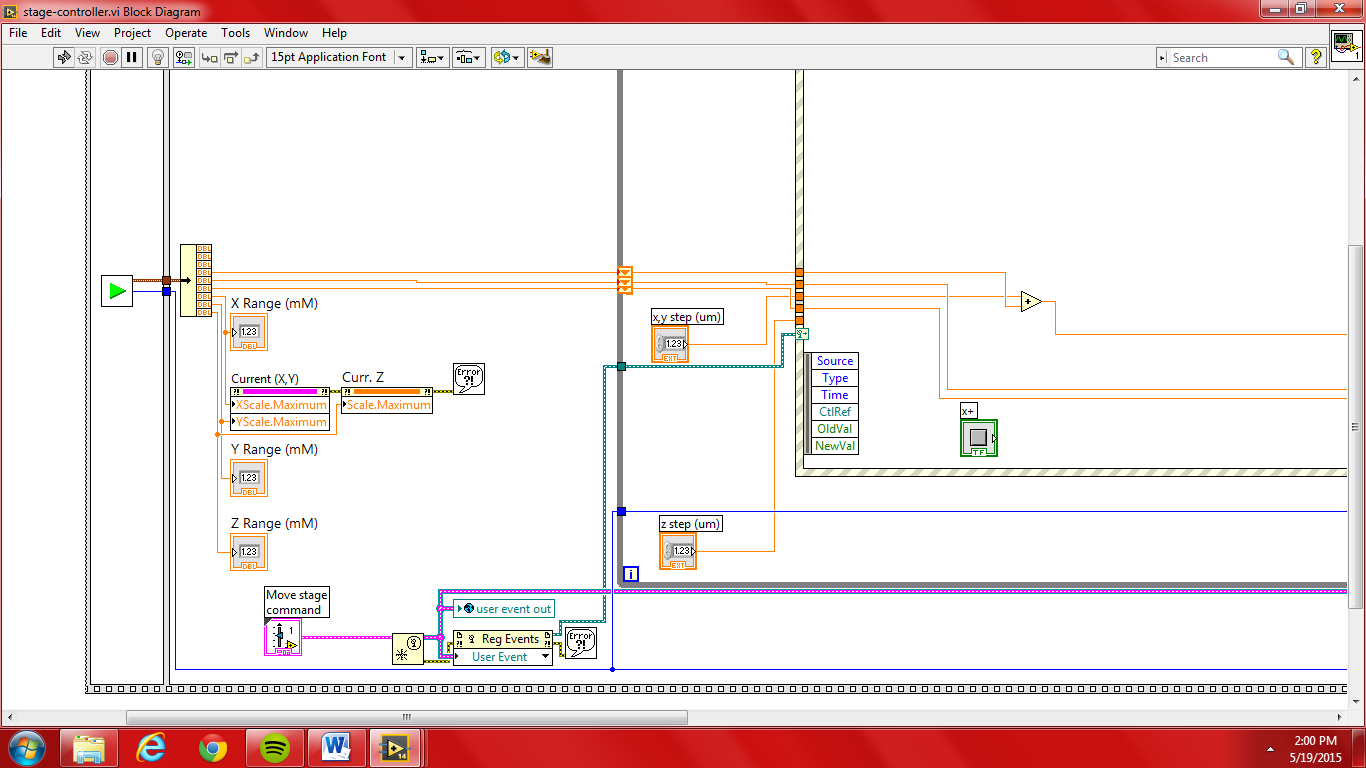
**stage-controller.vi**

This vi serves to adjust the stage’s position according to the user’s specifications. A plot for the x & y axis is shown as well as a slide plot for the z-axis. The values of the axis can be incremented in varying degree, depending on the user’s choice. Current axis values and their range is on display. The black button resets the position to the center of the stage while the red stop button quits the user out of the stage controller.



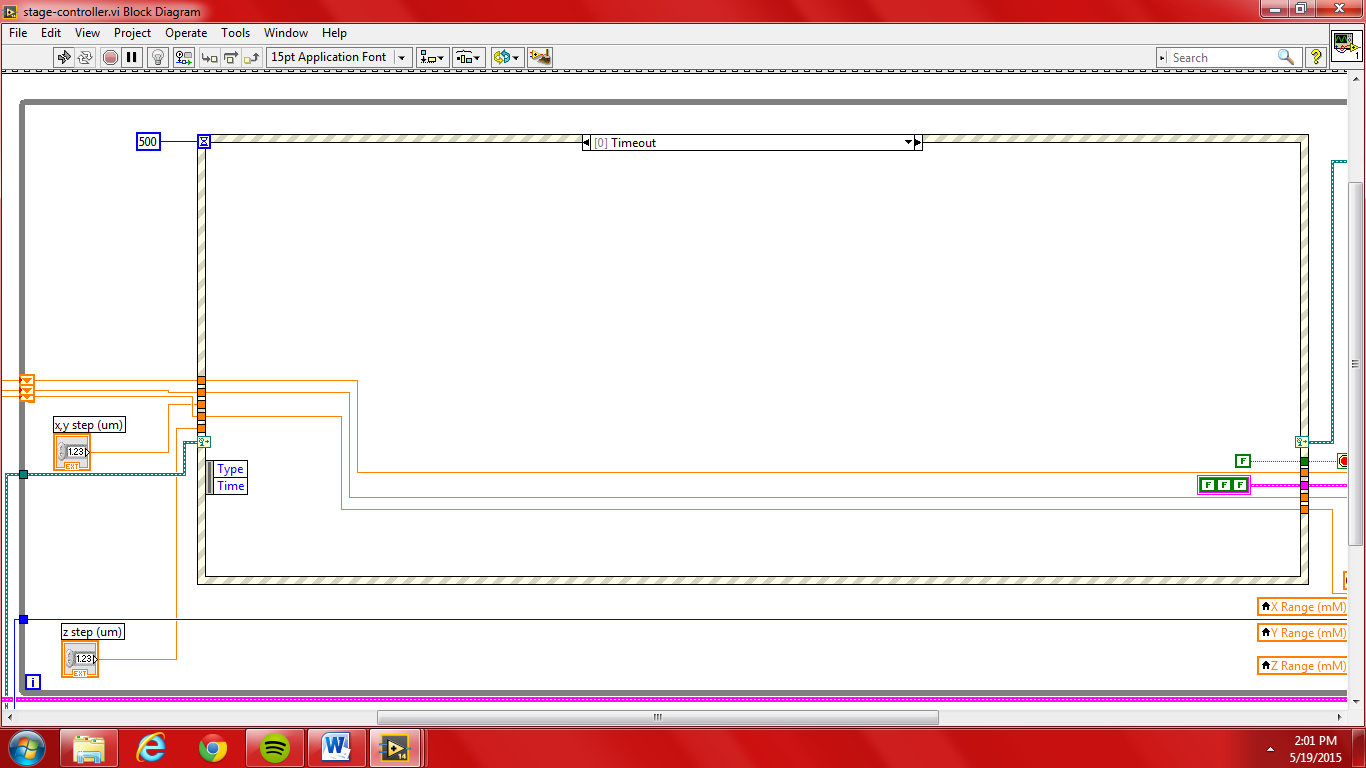
SubVIs:

1. Initialize-NanoDrive-SubVI
   1. Initializes the stage/nanodrive
   2. Determines the range of the axis (x,y,& z)
2. Move stage command – Global - SubVI
   1. This vi was taken from the Glimpse VI stage controller
   2. Adjusts the stage depending on the user’s step selection
3. NanoDriveInfo-SubVI
   1. This is actually a subVI of Initialize-NanoDrive-SubVI
   2. Obtains information from the nanodrive/stage
4. Nano-drive-usb-SubVI
   1. This VI was copied from Glimpse stage controller
   2. Outputs the current values of the axis to the nano drive
5. nano-positioner
   1. This VI was created prior to the creation of the stage-controller
   2. No longer relevant - can be disregarded/discarded
6. user event out globals - SubVI
   1. Global variable
   2. Obtained from Glimpse global stage configuration
   3. Purpose/functionality uncertain



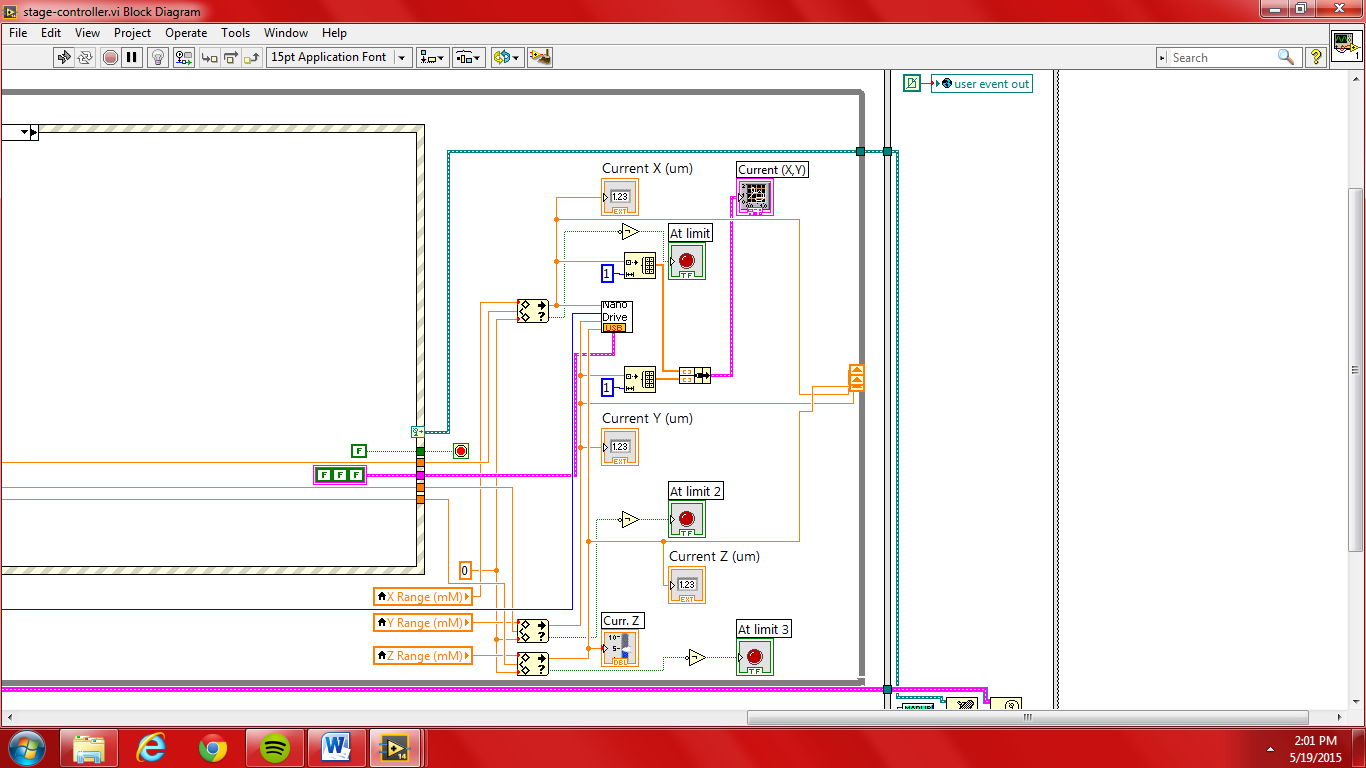
In the above figure:

The nanodrive is initialized by Initialize-NanoDrive-SubVI which outputs the handle as well as the current x,y, & z axis and their range. The range values are immediately sent to the x-y and z plots to set the max values. The move stage command contains default values to be used in the “Move Stage” event within the main for loop’s event structure.



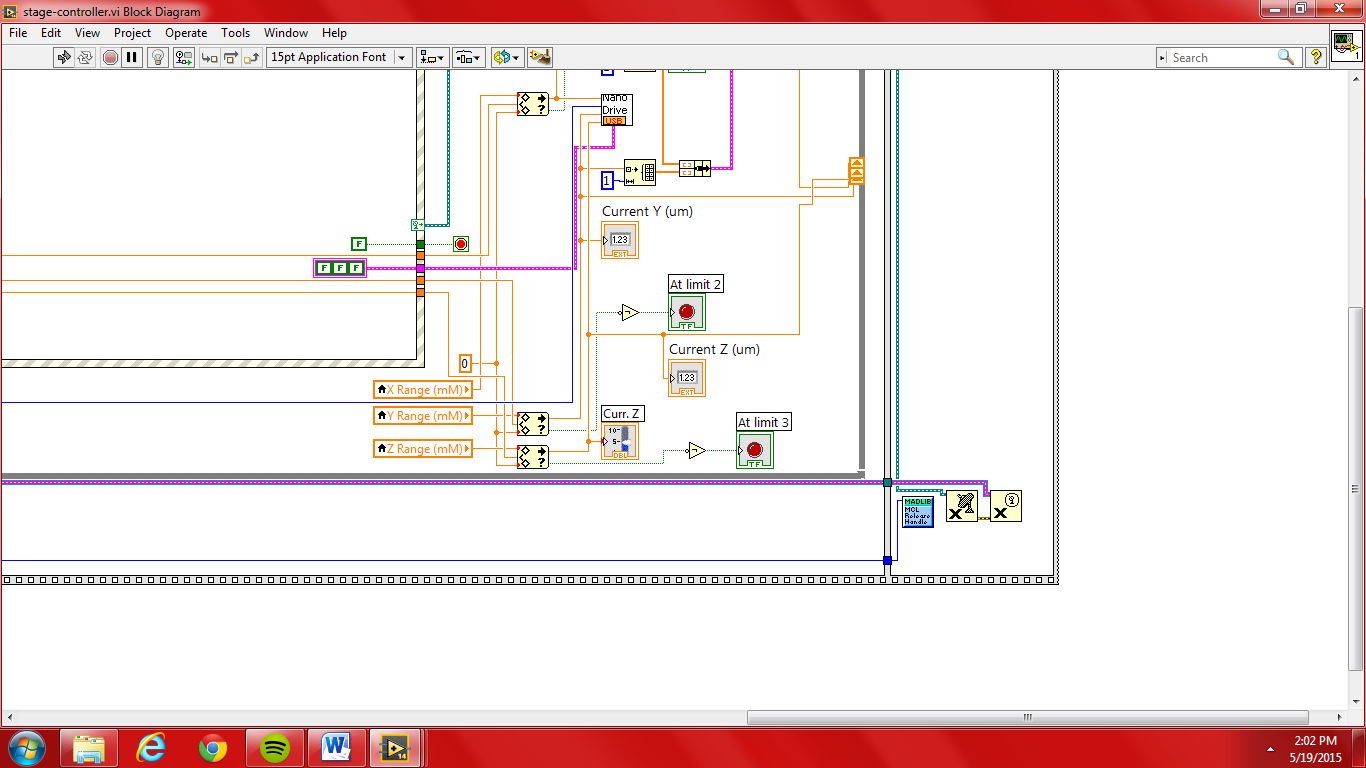
In the above figure:

The current values, obtained from Initialize-NanoDrive-SubVI, are output into the for loop and used as shift registers. Within the for loop, the user’s step increments in either axis is accounted for then output to the nanodrive which can be seen in the image below.



In the above figure:

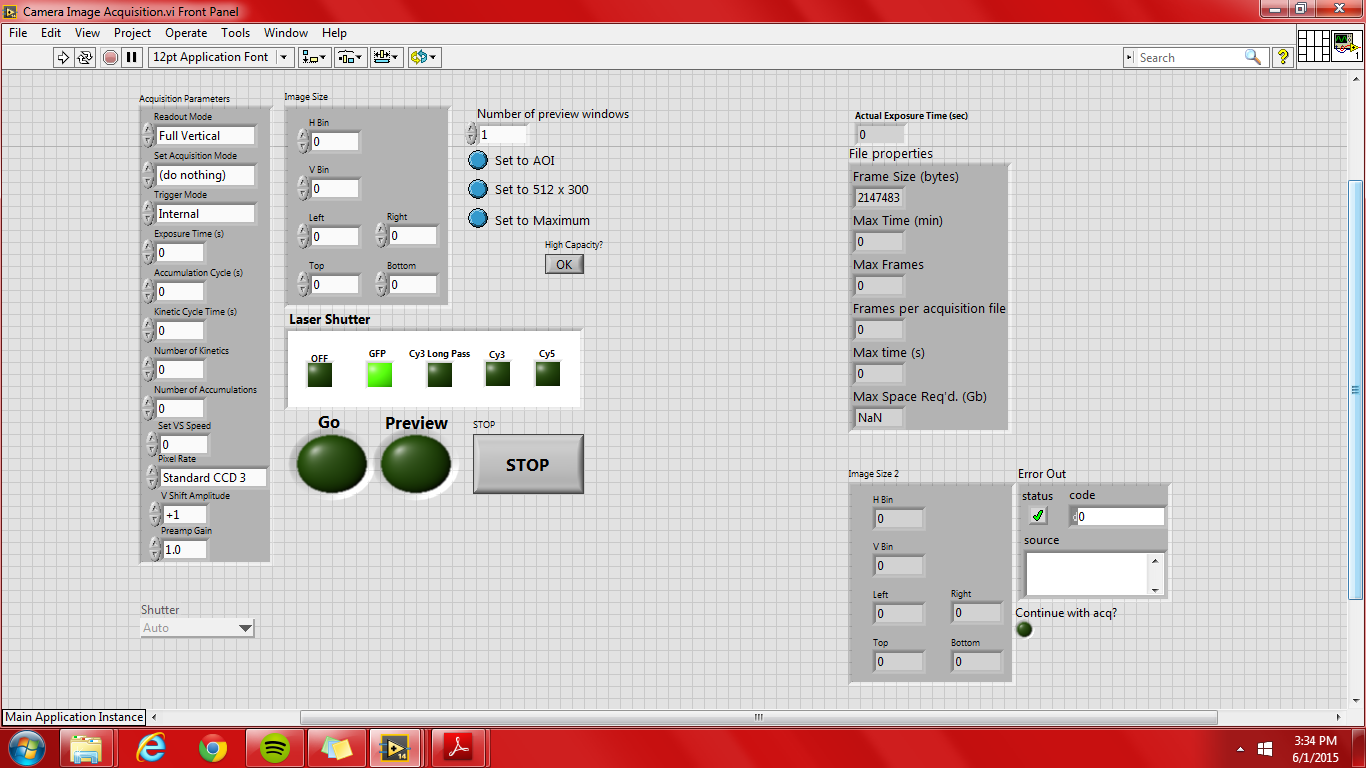
The value which has been altered by the user it sent to the nanodrive-lambda subVI which adjusts the stage accordingly. The current stage values are displayed. If a current value is at the max of the stage, lights are flashed for the appropriate axis. The altered axis values can be seen in the plots. The user event global variable is not understood but it seems to confirm that the user has ended their control of the stage.



In the above figure:

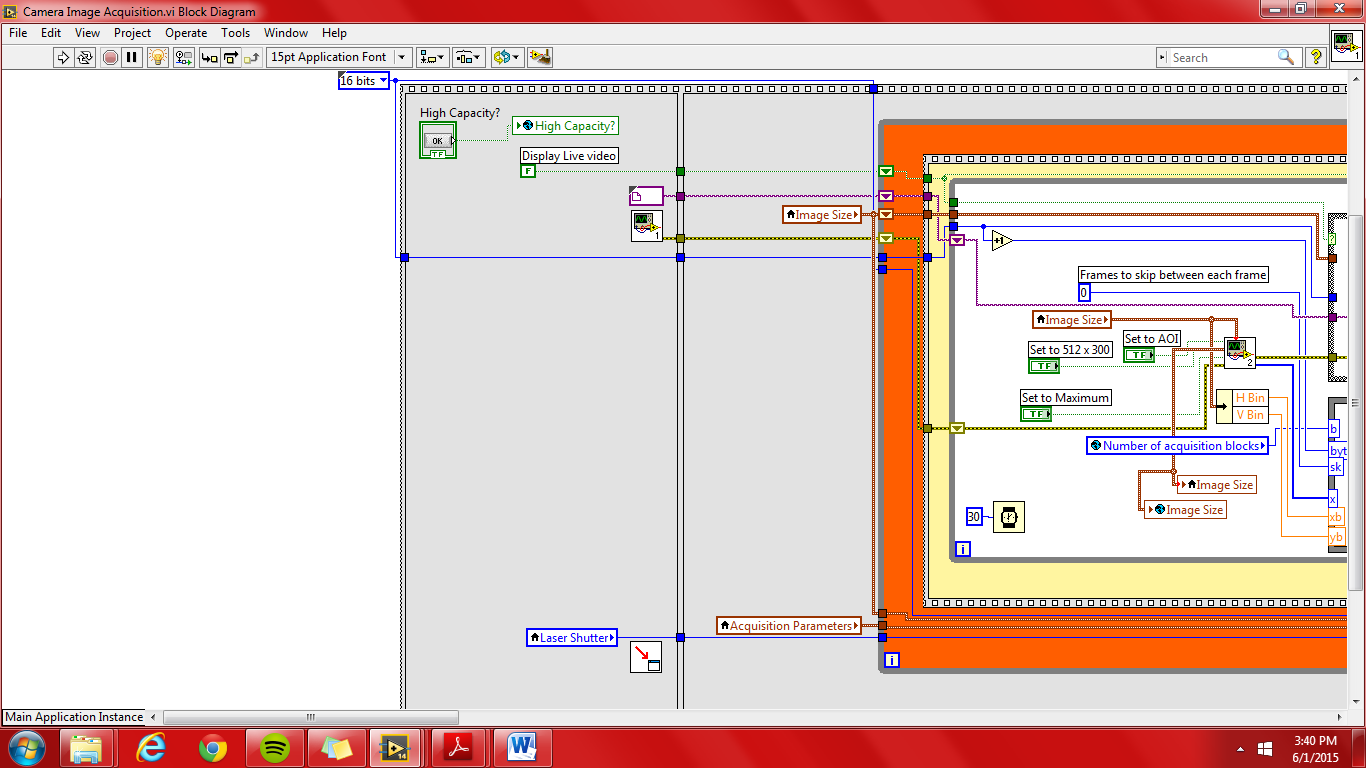
The handle value from Initialize-NanoDrive-SubVI is sent to the mad city library function, release all handle. The other two functions are not well understand and were taken from Glimpse along with user event out.

**Camera Image Acquisition.vi**



This VI is found within camera test folder of Lambda Testing. The purpose of this VI is to acquire an image(s) during the live feed while a laser(s) is turned on.

**\*VERY IMPORTANT\*** Make sure that the settings are adjusted to your specifications prior to running the VI, especially the Image Size parameters. However, *the Boolean Variables maybe altered whenever* during the VI run.

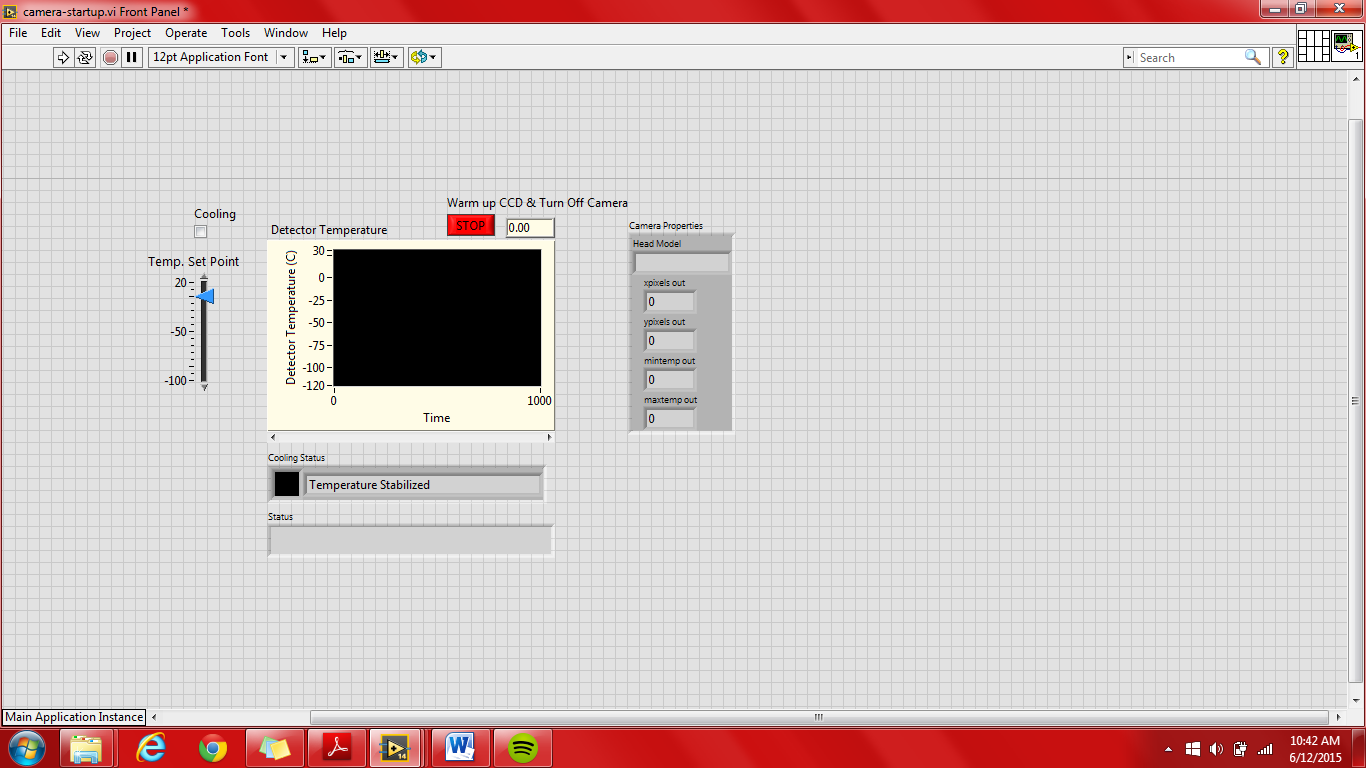


In the above figure:

* + The first event in the program, only ran once.
  + The Boolean High Capacity is accounted for then sent to the Initilize Globals which is subsequently used in Initialize Andor.vi.
  + The Initialize Andor.vi initializes the Andor camera. This only needs to be done once.
  + The Position calling VI to…vi chains the order of VI’s opened during the course of the program’s run.
  + The Laser Shutter local variable outputs the user’s selection of which laser to use.
  + An image file to be used in case of default is ouput to the remaining events. Only used for preview.
  + The display live video Boolean is hardcoded to false. If this should be changed is not understood.
  + The default bit size of the images to be acquired is hardcoded to 16 bits. Although additional bit sizes are listed within the array, only the 8 bit size could possibly work.

**camera-startup.vi**

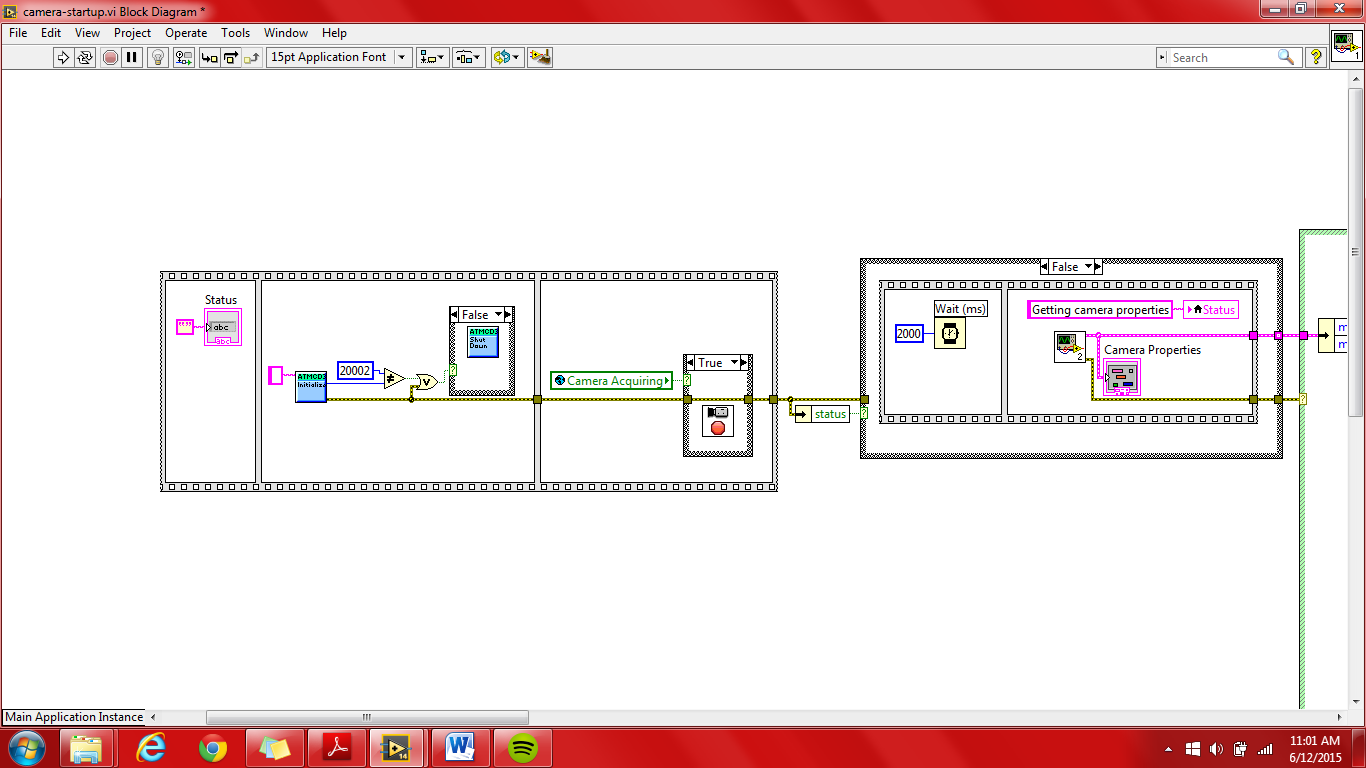
This VI functions to control the temperature of the Andor camera. This VI should be used prior to performing any acquisition and at the completion of an analysis in order to properly turn off the camera by it to room temperature (~20°C).



Sub-VI’s:

* Andor Vi’s
  + Cooler On.vi
  + Cooler Off.vi
  + Initialize.vi
  + Shut Down.vi
  + Set Temperature.vi
  + camera stop.vi
* camera-get-info-subi.vi
* get camera temperature.vi

**\*\* Note:** Any values of 20002 within the VI are codes to confirm that the andor vi’s were successfully performed.

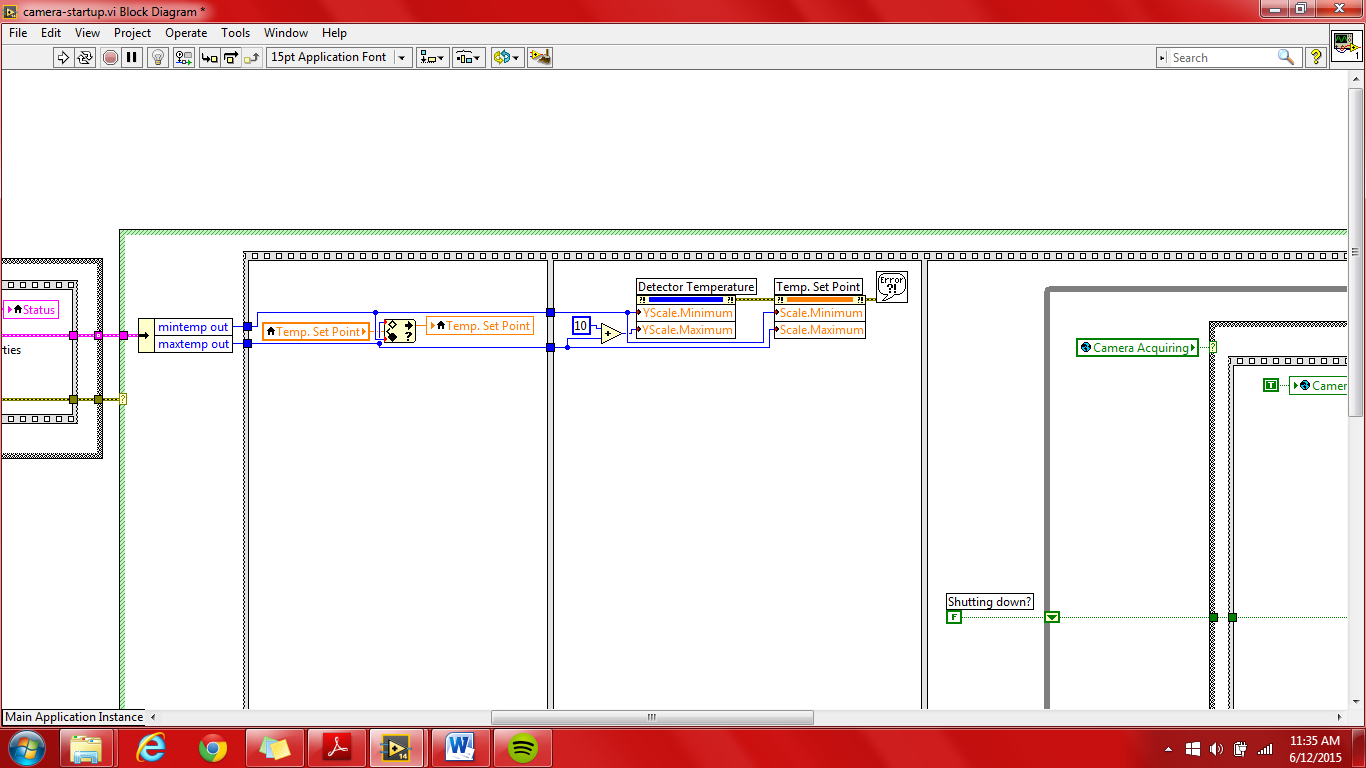


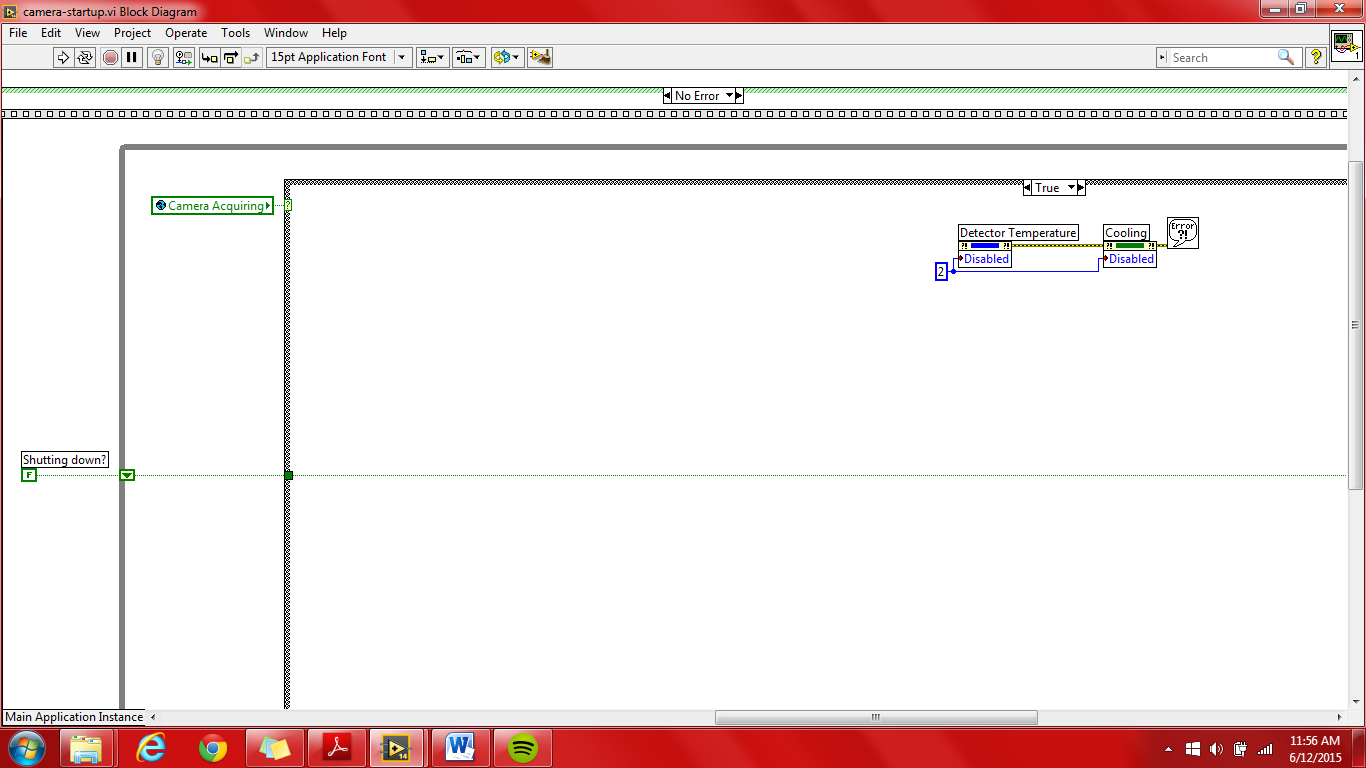
In the above figure:

The Andor camera is first initialized via the *Initialize.vi*. If initialization was unsuccessful for whatever reason, the camera shuts down via the *Shut Down.vi*.

If the global Boolean Camera Acquiring indicates that the camera is already acquiring, then the vi *camera stop.vi* will force it abort acquisition.

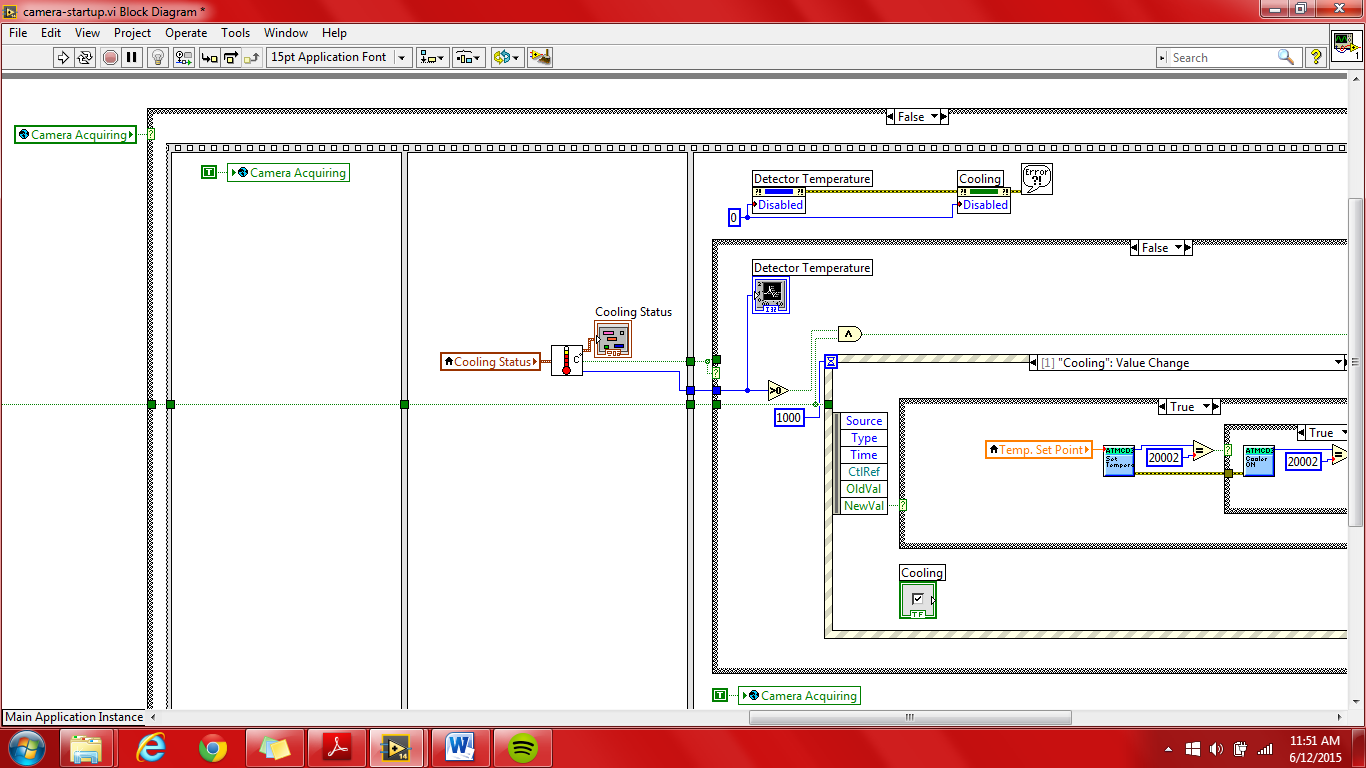
In the far right True/False structure, if no error was generated in the previous Vis, there is a 2 second delay prior to the initialization of the *camera-get-info-subvi.vi*. In the camera-get-info-subvi.vi, the Head Model of the camera, X & Y pixel values, and temperature range are obtained. These values are clustered and ouput from the VI. The range from this VI is used to establish the min and max of the Temp. Set Point numeric slide bar and the detector graph as shown in the image below.





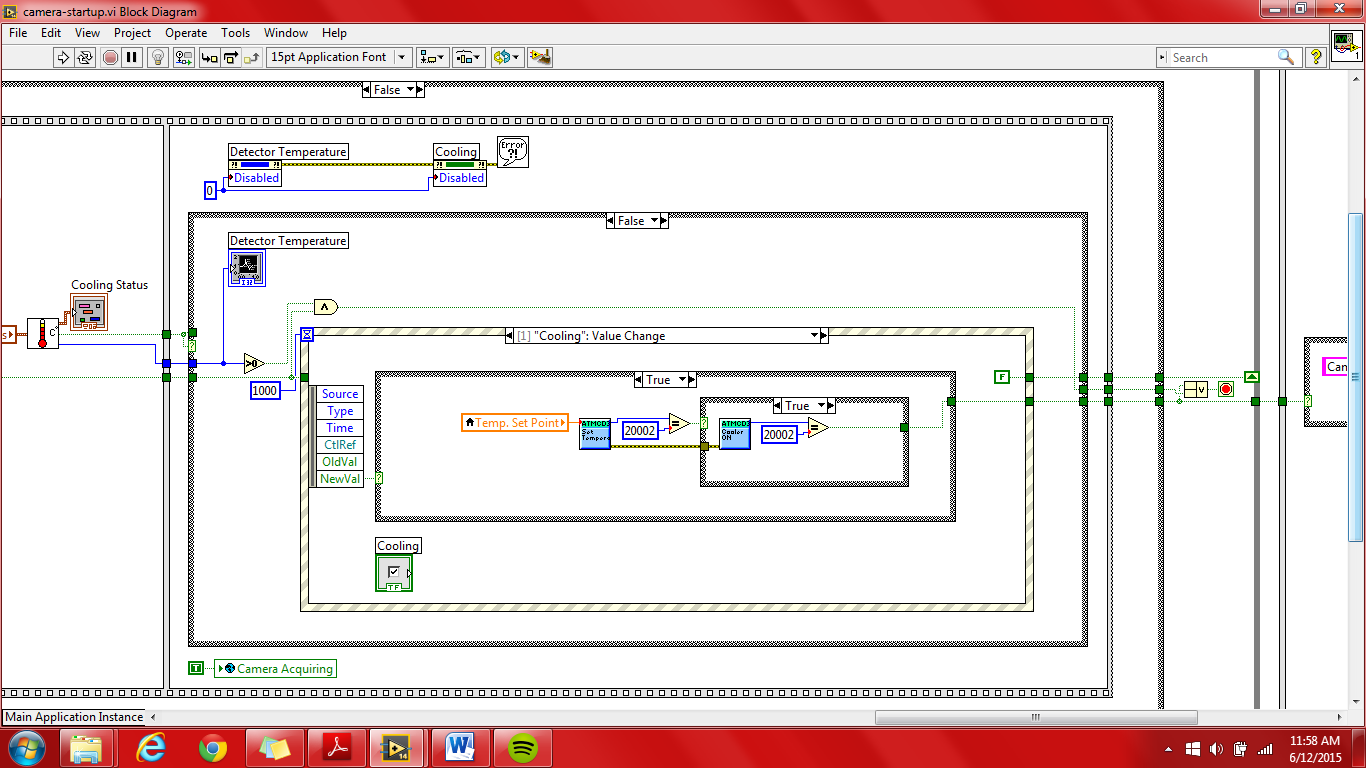
In the above figure:

The figure above shows the case structure if the camera is currently acquiring. If the camera is currently acquiring, then the detector temperature graph and cooling status are disabled.



In the above figure:

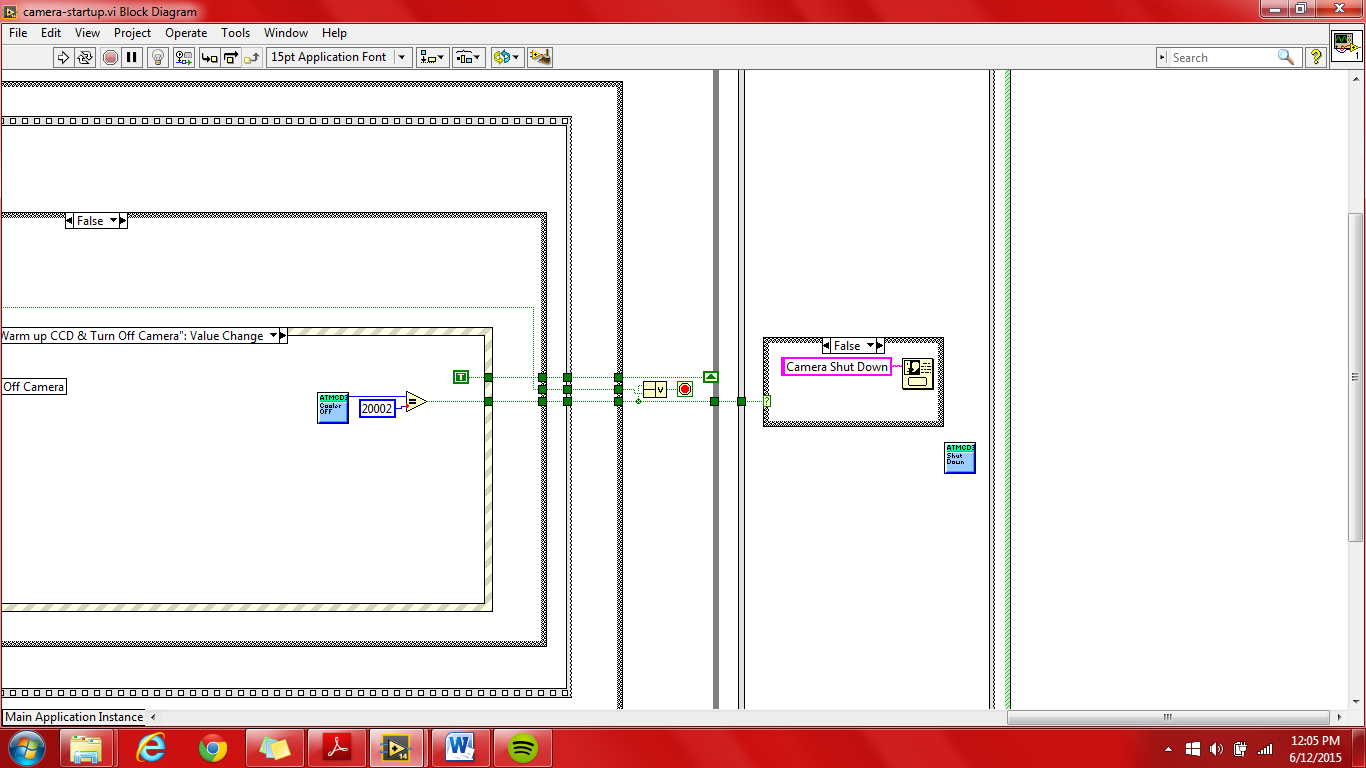
The current cooling status is input into the *get camera temperature – subvi.vi* where the current temperature and cooling status are output. A Boolean indicating if an error occurred is also an output.



In the above figure:

This inner case structure controls the display on its front panel as well as setting the temperature of the Andor Camera via the event structure.The event structure accounts for the following:

* Timeout
* Cooling Value Change
* Temp. Set Point Value Change
* Warm up CCD and Turn Off Camera Value Change



In the above figure:

The user is shown a message upon the termination of the VI, depending on if an error occurred or shut down was successful. The Shut Down.vi shuts down the camera.