Triggering for Magnetic Field Measurements of the LCLS Undulators

Kirsten Hacker, Zack Wolf
SLAC

February 22, 2005

Abstract

A triggering system for magnetic field measurements of the LCLS undulators has been built with a National Instruments PXI-1002 and a Xylinx FPGA board. The system generates single triggers at specified positions, regardless of encoder sensor jitter about a linear scale.

1 Introduction

In order to map the magnetic field in a 3.4 meter long section of undulator with a Hall probe, the position of the probe for each measurement must be known. The position change is measured with a linear encoder with quadrature output from which forward and backward steps are counted. If the number of backward steps is subtracted from the number of forward steps, the position of the probe is known with an accuracy determined by the fineness of the linear scale grating. A system that generates triggers at specified positions (Fig. 1) would produce evenly spaced measurements if the read-head of the encoder never oscillated or jittered around a single position as shown in Fig. 2. Any oscillation would generate multiple triggers for a single position and there would be no way of knowing where these doubly triggered measurements occurred unless one had a way to record the position for each trigger or a way to block trigger generation for a position whose measurement has already been triggered.

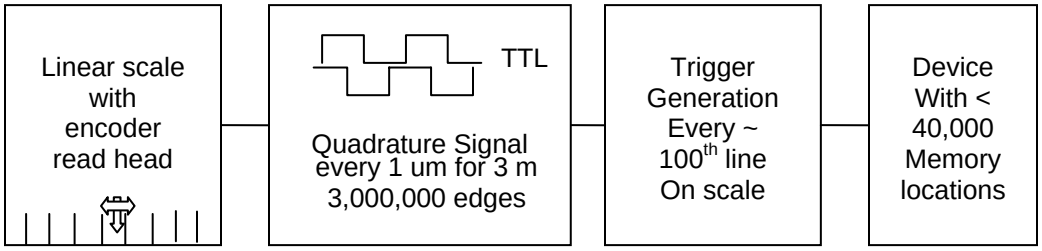

FIG 1. Generating triggers for a device with limited memory locations.

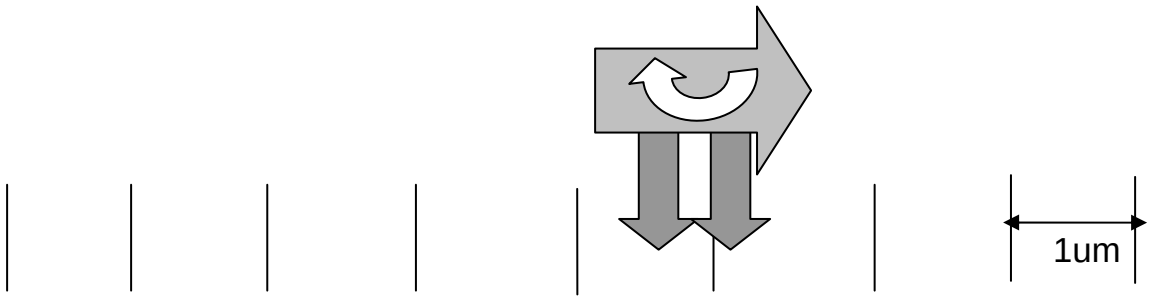

FIG 2. Twisting of the carriage as it moves longitudinally can cause multiple triggers for single positions on the linear scale.

Following the suggestion of Joseph Xu from Argonne, a Xylinx FPGA (Field Programmable Gate Array) module was programmed with LabView to use the encoder output to record the position of each and every trigger, thus providing a record of the location of every multiply triggered position. This system was adapted as shown in Fig. 3 to record the position of every trigger and also to prevent the generation of multiple triggers for single positions. The system allows for a PC running the main process control program to communicate with the PXI crate by reading and writing commands to a text file on a shared, networked drive. The PXI crate, in turn, reads from and writes to the FPGA module. This is ideal for our existing process control software and our measurement devices with limited memory resources.

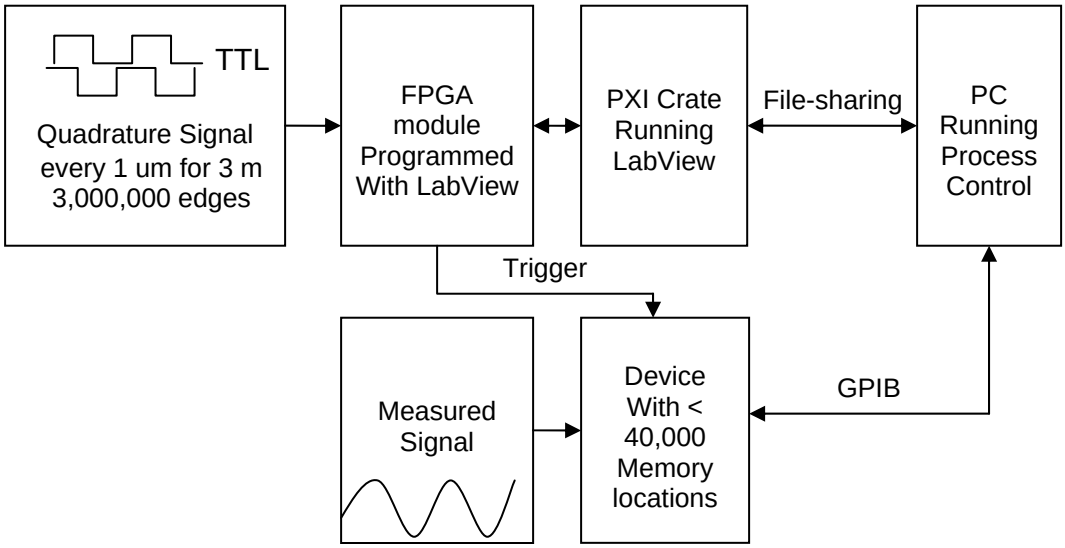

FIG 3. Triggering system data flow.

This FPGA method of generating evenly-spaced triggers and recording their positions was applied to linear scales in x, y, and z. The z scale is attached to the granite table and the x and y scales are mounted adjacent to stages moving in x and y that ride on the carriage moving in z (Fig. 4). Measurement device DC drifts are a concern, as well as overall measurement time, so the carriage cannot be brought to a halt and carefully positioned with a servo motor for each measurement.

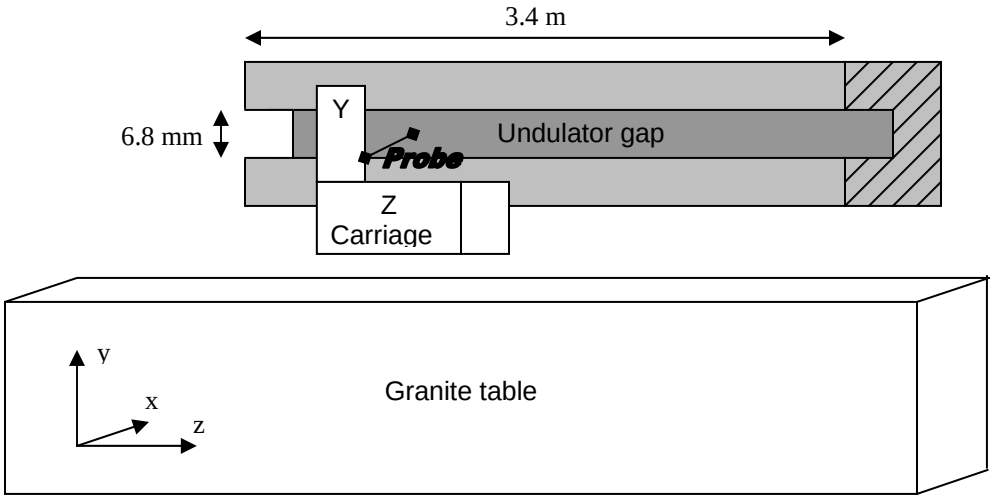

FIG 4. Control of probe motion. The Z carriage moves over an air bearing in the z direction. It carries stages that move in x and y.

The x and y linear scales are used for Hall probe scans around selected pole-pairs to find each pair's magnetic center, thus allowing for alignment of the undulator to the table. The scans in x give the yaw of the magnetic center of the undulator with respect to the table and the scans in y give the pitch. Once the magnetic axes of the undulator are aligned to the table, the z scale is used for Hall probe scans that give information about differences in the strength of poles along the undulator and allow for correction of these differences with the application of shims that attenuate the magnetic field. For all of these measurements, it is important to know the position of the measurement probe to better than 40 μm in x, 20 μm in y, and 3 μm in z (LCLS-TN-04-8).

2 Principles of Operation

A Xylinx FPGA chip on an NI PXI-7831R board was programmed with LabView FPGA 7.0 such that for every edge of the quadrature signal (shown as A and B in Fig. 5) the direction of motion is determined and used to increment or decrement a counter. There are three different counters that use this process.

- The first counter starts when the FPGA chip is initialized or reset by the host program running on the PC. It gives a position relative to the arbitrary starting position. Each count corresponds to a position equal to the count number times the distance between marks on the linear scale.
- The second counter starts and stops counting following the detection of an adjustable number of index pulses. An index pulse is shown as Z in Fig. 5. This counter gives a position relative to the selected starting index position.

- A third counter starts and stops at the same beginning and ending positions as the second counter, but it repeatedly counts up to 100 and is reset to 0 so that it can count up to 100 again and again until the final index position is reached. This counter is limited to 100 because that is the maximum array size that can be encoded on the FPGA chip and it is used as a pointer to a location within an array which serves as a mask to determine whether or not a measurement at a specific position has already been triggered. If there is a 0 in the mask array at a position, a trigger can be generated and a 1 is placed into the mask array at that position. If the encoder read-head crosses that position again, there will be a 1 in the array, ensuring that a trigger cannot be generated again. The array is continually re-used; when one pointer replaces a 0 with a 1, a pointer 50 counts away replaces a 1 with a 0.

A' XOR **A**	**B'** XOR **B**	**A'** XOR **A** OR **B'** XOR **B**
T	T	T
F	T	T
T	F	T
F	F	F

A'	**B**	**A'** XOR **B**
T	T	F
F	T	T
T	F	T
F	F	F

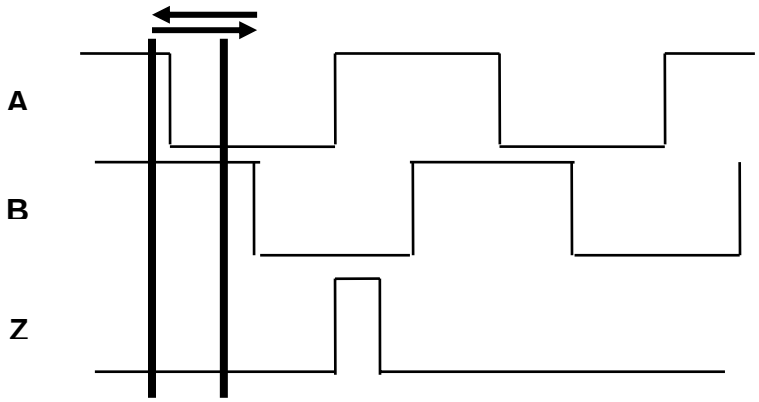

FIG 5. Direction and motion information from a quadrature encoder output. Each vertical line, in bold, represents a measurement of the quadrature signal. If an edge of the quadrature signal is detected, then the carriage has moved one step either forwards or backwards. An edge is detected if A' XOR A is true OR B'XOR B is true (where the prime denotes a sample taken in a previous iteration of the FPGA looped execution). The carriage has moved forwards if A' XOR B is true and backwards if it is false. After an adjustable number of Z rising edges are detected, the triggering is started and stopped.

A trigger is generated when the value of this third counter is equal to a multiple of an adjustable input parameter AND when 0 is the value of the mask array at the pointer equal to the value of the third counter. The purpose of the multiple is to allow for an

adjustable distance between triggers. This is useful because the distance between triggers needs to be different for the x, y, and z scans.

An error associated with starting and stopping the edge counting is generated due to the variability in the time it takes to detect the rising edge of the index pulse but since this error occurs at either the beginning or the end of the measurement, and since it is always less than 4 edges or one mark on the linear scale, it does not affect the repeatability of the trigger locations. There are always 4 edges per count, so an error of plus or minus 3 edges is always less than one cycle of the linear scale

It is possible that if the stage were moved too rapidly across the linear scale that the FPGA module could not sample fast enough to count all of the edges of the encoder signal. The sampling speed of the FPGA is greater than 40 MHz and this translates to a maximum carriage speed on a scale with 1 micron/count of 10 meters per second. Since the carriage travels 4 meters in approximately 40 seconds, this speed issue will not be a problem. For our application, the trigger frequency is limited by the voltage measurement device.

3 Equipment

The z scale on which the FPGA triggering system was tested is the Heidenhain LIDA 205 incremental linear encoder with 1 micron resolution. The index pulses are 50,000 counts apart and can be activated by attaching a 3 KGauss magnet next to the linear scale with the north and south poles on either side of the desired index line. The y scale has .5 micron/count measuring steps and 40,000 steps in between index pulses. The presently installed x scale has no index pulses. Unfortunately, the FPGA triggering system requires index pulses for operation and the x scale will need to be replaced or artificial index pulses will need to be generated.

The HP3458A multi-meter can take 39,000 triggered measurements with extended memory. This means that for measurements taken over 4 meters with .25 micron/count, the distance between triggers needs to be 100 microns or 400 edges.

The National Instruments PXI-1002 crate with a PXI-7831R reconfigurable I/O (RIO) device is connected to an SCB-68 pin shielded connector block. By programming the FPGA on the RIO device with LabView FPGA, one can define the digital and analog functionality and timing of the device. The advantage of programming with LabView FPGA instead of the FPGA Xylinx language is that LabView can speed up development time through a user friendly interface. The sinusoidal encoder signals from the LIDA scale are transformed into square wave signals with a Heidenhain EX563. They travel 5 meters through a shielded cable to a G10 terminal block, whereupon they travel half of a meter to the SCB-68 and thereby into the FPGA I/O channels. This encoder signal is sensitive to noise generated by the motor driver and a large amount of noise can be eliminated by selecting a special low noise motor driver with DC power that is not modulated by chopping. The Compumotor micro-step series LN motor driver was chosen due to availability and familiarity. Any remaining random noise spikes can be eliminated with a Schmitt trigger, a comparator, and/or a line driver. Enough of the noise was removed through power supply and motor driver replacement such that implementation of additional measures was not necessary. These additional measures were, however, investigated as a safeguard against system changes that could introduce additional noise.

The trigger is sent from the SCB-68 to two HP3458A multi-meters that are initialized to accept triggers with the LabWindows main-control program. The first multi-meter measures the Hall probe voltage from the vertical field and the second multi-meter measures the Hall probe voltage from the horizontal field.

SCB pin	FPGA Channel	Encoder signal
36	DIO1	X axis A
37	DIO2	X axis B
38	DIO3	X axis Z
39	DIO4	Y axis A
40	DIO5	Y axis B
41	DIO6	Y axis Z
42	DIO7	Z axis A
43	DIO8	Z axis B
44	DIO9	Z axis Z
45	DIO10	Output trigger
46	DIO11	FPGA speed

FIG 6. SCB board connections and pin-out. The SCB board should be connected to the PXI-7831R port 0.

4 Software Design

The LabView environment consists of “block diagrams” shown below in Fig 7 and Fig 8 and “control panels” shown in section 5, Fig. 9 and Fig 10. The desired process is written onto the block diagram in the form of a graphical program and is controlled by a user with the control panel. The LabView FPGA environment is different from regular LabView in that one writes a LabView control panel and a block diagram for the FPGA chip on a PC and this LabView code is translated into Xylinx code, compiled, and downloaded to the FPGA chip where it runs independently while a “host” LabView program communicates with the chip. Example icons from the host block diagram are shown in Fig. 8. One reads from and writes to the process that runs on the chip from either the FPGA control panel or the host control panel. One must, however, switch the “execution target” when running the FPGA control panel or compiling the block diagram.

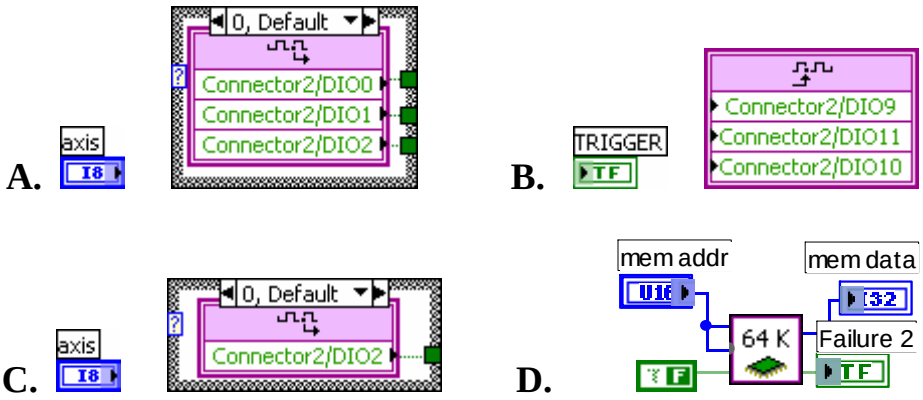

FIG 7. LabView FPGA block diagram icons. One while loop contains the encoder inputs [A] and position counters, mask array, and trigger generation [B]. Another while loop contains an index counter [C]. Still another while loop contains a process that writes the positions to a buffer that is read by the host program [D]. These icons do not represent the entire process, they are merely examples of icons unique to LabView FPGA.

At start-up, the LabView host program, “trigger host.vi”, sits in a waiting state, continuously looking for setup information from a text file on a public, networked drive shown in Fig. 8 as “V:\MET\MagServ\PXIStuff\status.lvm”. This setup information text file is initially blank because the LabView host program clears the file during initialization or following the successful conclusion of trigger generation. The LabWindows main control program running on a remote PC writes startup information onto this text file, status.lvm, thereby letting the LabView/PXI host know that the main control program has initialized devices to accept triggers and will initiate probe motion. The PXI host program will then send a list of initialization information to the FPGA followed by a reset command that prepares the FPGA to begin reading the input encoder signal shown in Fig. 7-A. When the FPGA detects the number of index pulses (Fig. 7-C) specified in the initialization file, the generation of triggers (Fig. 7-B) begins. Following the detection of a specified number of subsequent index pulses, the generation of triggers stops and the host program clears the initialization file, “status.lvm”, and resumes its waiting state activities, continuously checking for initialization information sent by the main process control PC running LabWindows.

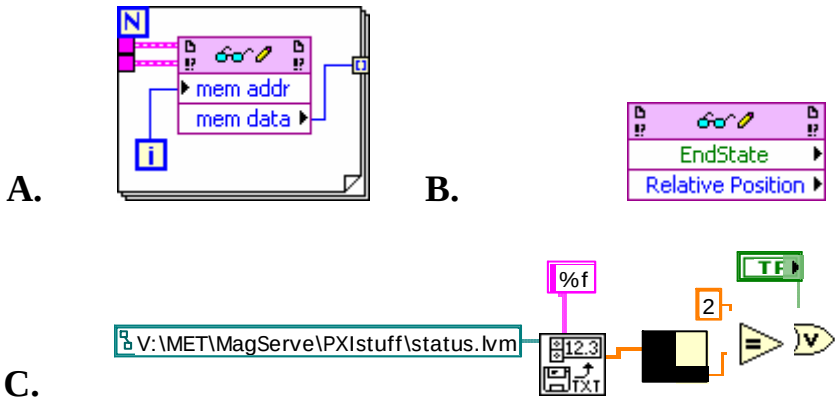

FIG 8. LabView host block diagram icons. The host program reads the buffer of position values "mem data" from the FPGA chip buffer location "mem addr" shown in A. Since the FPGA is faster than the LabView on the PC, a buffer is used to compensate for the data acquisition at different speeds. The host program also provides information about the "Relative Position" to the user and waits for an "EndState" boolean value to become true in order to return to the waiting state shown in C.

In parallel to this triggering, edge counting process is a loop that writes the position at which every trigger is generated to a location in a memory block (Fig. 7-D). The host program reads the position information from this memory block (Fig. 8-A) and shapes the information into an array that will be written to a file. The FPGA writes to the memory block more rapidly than the PC can read from it and that is why a large memory block is necessary. While the FPGA is writing to one portion of the memory block, the PC is reading from another and when the FPGA is done writing to the block, the PC can continue to read. This memory buffer ensures that no data is lost in the transfer from FPGA to PC.

5 Software Operation

One can control the triggering system from either the PXI PC LabView interface or one can write commands to an initialization file that puts the triggering system in either a waiting state or a trigger generation state. The system will initialize itself following a PC re-start so that the LabWindows can control the system by writing to the text file on the public drive. More information for trouble-shooting, however, can be gained by looking at the LabView control panels on the PXI PC.

Following a PC re-start, the host control program runs automatically. This, in turn, starts the FPGA program running on the FPGA chip. The host program then looks at the file shared with the LabWindows main process control program for setup information and a cue to start counting index pulses. This shared file exists on a drive that both the main process control program and the LabView programs can access.

Fig. 9 shows the "trigger FPGA.vi" control panel. On the right are "mem 1", "mem 2", and "mem 3". These variables let the host program know which section of the

memory block has already been filled with position information. The variables “mem data” and “mem addr” allow the host program to point to locations within the memory block and read out the position information contained therein. “Start Position” gives the “Absolute Position” at which the “Reset” cue was given. “Relative Position begins incrementing when the “EndState” cue becomes false and stops incrementing when it becomes true. Following a “Reset”, “EndState” always becomes true. “Reset Toggle” is a used by the host program to re-initialize the FPGA before every triggering run. “counts/index” is set by the host program and should always be less than 3 counts different from the “Index Save” read-back. “EndState” becomes true when “Index Count” is greater than “# indices”. “Output Count” times “2ond multiple” times “1st multiple” should always agree with “Relative Position”. The X, Y, or Z axis can be selected with “axis” 1-3.

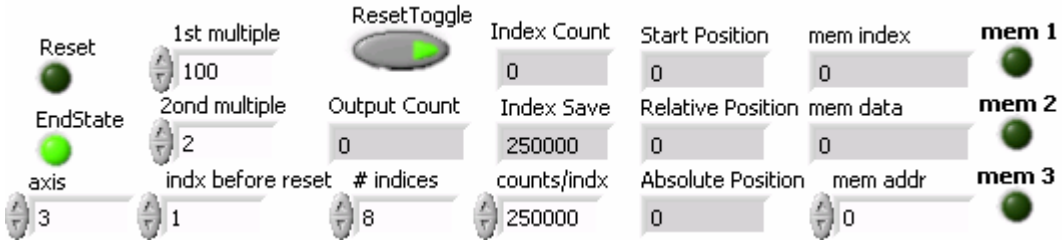

FIG 9. LabView FPGA control panel. This panel shows all of the values that can be changed or read out from the FPGA chip. One can control the FPGA vi from this panel when the host program is not running. Running this panel starts and stops the process on the chip.

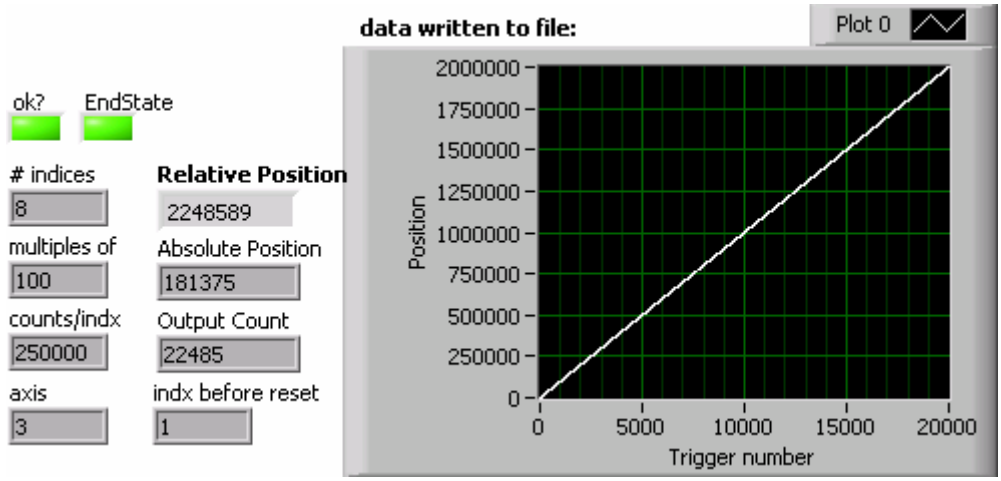

FIG 10. LabView host control panel. Running this panel starts the process on the chip, looks for the setup information in the file shared with the LabWindows main process control program, and writes that information to the FPGA chip. A straight line in the “data written to file” window at the end of a run indicates that no triggers were repeated.

Fig. 9 shows the control panel for the “trigger host.vi”. It shows the control values that have been read from the “status.lvm” file written by the LabWindows control PC as well as the position, output, and status information from the FPGA. Running this panel starts the process on the chip, looks for the setup information in the file shared with the LabWindows main process control program, and writes that information to the FPGA chip. A straight line in the “data written to file” window at the end of a run indicates that no triggers were repeated.

The triggering system operator should be aware of a few potential difficulties. If the carriage stops before an expected number of index pulses has been detected, the system must be restarted. If the number of edges detected in between index pulses does not match the expected counts/index value set in the initialization file, the triggered positions file will be truncated or repeated.

6 Possible developments

Since the z scale cannot be mounted directly adjacent to the measurement probe, the position at the probe will not exactly match the position on the linear scale if the carriage twists laterally as it moves in z. This discrepancy can be eliminated by adding an additional linear scale in z that is mounted on the opposite side of the carriage so that positions from the two z linear scales, the distance between the two scales, and the distance from one scale to the probe give the real position in z at the location of the probe. This information about the angular motion of the carriage can also be gotten by attaching a mirror to the carriage and using an autocollimator to measure the angular movements of the front face of the carriage. The autocollimator technique requires post processing of the data, whereas the FPGA method can adjust the position data on the fly.

7 Conclusion

A National Instruments PXI-1002 crate with PXI-7831R reconfigurable I/O board was used to program an FPGA chip to create triggers from quadrature encoder signals from 3 different linear scales. The system can be controlled remotely by writing commands to a text file stored in a public drive.

Acknowledgements

Thanks to Joseph Xu of Argonne National Lab for the suggestion of the PXI-FPGA system for triggering and for the demonstration of memory buffer read-back of position data from the FPGA.