

3-2018

Implementing an SPM Controller with LabVIEW

Jianghua Bai
Portland State University

John L. Freeouf
Portland State University

Andres H. La Rosa
Portland State University, andres@pdx.edu

Follow this and additional works at: https://pdxscholar.library.pdx.edu/phy_fac



Part of the [Physics Commons](#)

Let us know how access to this document benefits you.

Citation Details

Bai, J., Freeouf J., La Rosa, A. (2018). Implementing an SPM Controller with LabVIEW. *Journal of Measurement Science and Instrumentation*. Volume 3.

This Article is brought to you for free and open access. It has been accepted for inclusion in Physics Faculty Publications and Presentations by an authorized administrator of PDXScholar. Please contact us if we can make this document more accessible: pdxscholar@pdx.edu.

Implementing an SPM Controller with LabVIEW

Jianghua Bai, John L. Freeouf, Andres La Rosa

Portland State University, the Physics Department, USA

Abstract: The purpose of this article is to reduce the barrier of developing house made SPMs. Here, we cover all the details of programming an SPM controller with LabVIEW. The main controller has three major sequential portions. They are system initialization portion, scan control and image display portion and system shutdown portion. The most complicated and essential part of the main controller is the scan control and image display portion, which is achieved with various parallel tasks. These tasks are scan area and image size adjusting module, Y-axis scan control module, X-axis scan and image transferring module, parameters readjusting module, emergency shutdown module and etc. A NI7831R FPGA board is used to output the control signals and utilize the Z-axis real-time feedback controls. The system emergency shutdown is also carried out by the FPGA module. Receiving the shutdown command from the main controller, the FPGA board will move the probe to its XYZ zero position, turn off all the high voltage control signals and also eliminate the possible oscillations in the system. Finally, how to operate the controller is also briefly introduced. Messy wires fly back and forth is the main drawback of LabVIEW programming. Especially when the program is complicated, this problem becomes more serious. We use a real example to show how to achieve complex functionalities with structural programming and parallel multi-task programming. The actual code showed in this paper is clear, intuitive and simple. Following the examples showed in this paper, readers are able to develop simple LabVIEW programs to achieve complex functionalities.

Keywords: Scanning Probe Microscope (SPM), LabVIEW, FPGA, Multi-task Programming, Real-time Control

1. Introduction

Scanning Probe Microscopes (SPMs) are important tools in modern nanoscale research. House-made SPMs are popular because they are versatile, flexible, and able to perform non-standard measurements. Although SPMs can be used in quite different fields, the major technologies to build an SPM are similar. Usually, an SPM is made with a sensor which measures a certain kind of information about the sample, a piezo tube or piezo stage to manipulate the XY scans and control the distance between the tip and the sample (Z motion), a sensor signal processing unit which picks up the sensor signal and feeds it to the control system, a control system and data processing system and other support electronics. See Fig 1.

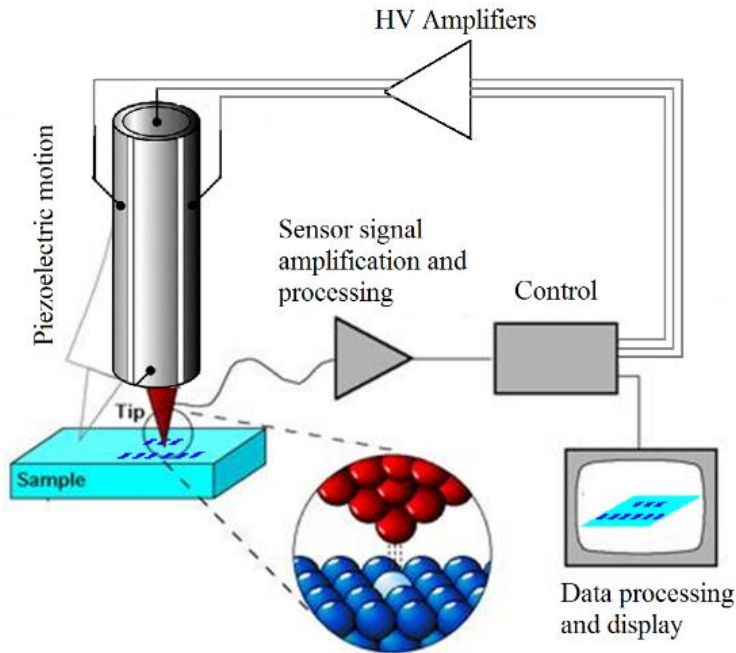


Fig 1, an illustration showing the working principle of an SPM

To build a house-made SPM successfully is not an easy job. Besides of building precise and stable hardware, one still needs a software system to control the whole scan process and implement the Z direction real-time controls. In order to facilitate the process of building a house made SPM, in this article, we use a tested example to show the design and implementation of an SPM controller with LabVIEW.

2. The Structure of the Program

As shown in Fig2, the main control program has three major sequential portions. At first, the program initializes the system, which includes initializing the FPGA, setting the default parameters to the controller and locating the probe to its zero position. Another task in the initialization step is to move the scanning head down to the sample and let the probe engage the sample. The second portion of the program is to scan the sample and display the images. This part is the most sophisticated and important part of the whole program. Its details will be covered in section 4. The last portion of the program is to turn off the system safely. After the scanning, the scanning head of the SPM should be raised from the sample to protect the probe. High voltage control signals were applied to the scanning stage during the imaging process. After the scanning is done, before turning off the system, one should move the probe to its zero XYZ position. Text to Speech (TTS) technology is also incorporated in this example. With a speaker connected to the controller, voice notices about the SPM state and possible actions the operator may take are sent through the speaker. This greatly facilitates the operation of the SPM.

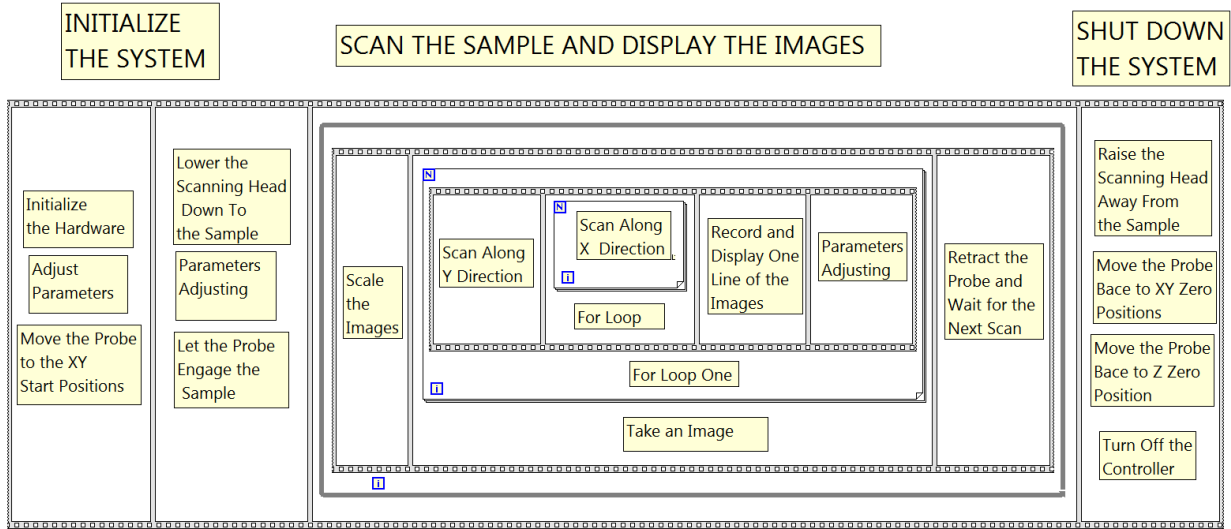


Fig2, the structure of the controller with pseudo LabVIEW code.

3. System Initialization (portion I)

This controller uses a NI7831R FPGA board to output the control signals and read the sensor data. In the first step of the initialization, proportional (P), integral (I), derivative (D) and the set point of the Z motion controller can be set up. These parameters can be set to the old experienced values from former operations of the SPM. During the experiment, according to signal levels and sample properties, these values can be readjusted further. Low and High are used to set the boundaries of the Z motion. There are two reasons to set up boundaries. One is that the piezo has a limit in its output range. The other is that one wants to limit the output of the PID controller to eliminate possible oscillations. These two parameters can be readjusted again, according to the roughness of the sample. LPTM is used to control the update rate of the voltage, which will control the speed of the Z motion eventually. PidTM is used to set the cycle time of the Z motion feedback controller. All these FPGA modules related parameters will be covered in detail in section 4.

In order to make the main program concise and simple, we code the subroutines as parallel tasks. These tasks are triggered by occurrences, which are supported by LabVIEW. We also use a notifier to transfer data from the scanning task to the main program. Occurrence and notifier examples can be found in Ref [1]. One can also code the parallel tasks as LabVIEW subroutines with a layered model. See Ref [2] [3]. The main difference is that all subroutines will be saved into different LabVIEW VIs. Here, in order to show everything in one VI file, we code all the functional modules as parallel tasks. The functionalities of all the tasks will be explained in following sections. In order to eliminate messy cross screen wires, local variables are used extensively to pass parameters and handle across different modules.

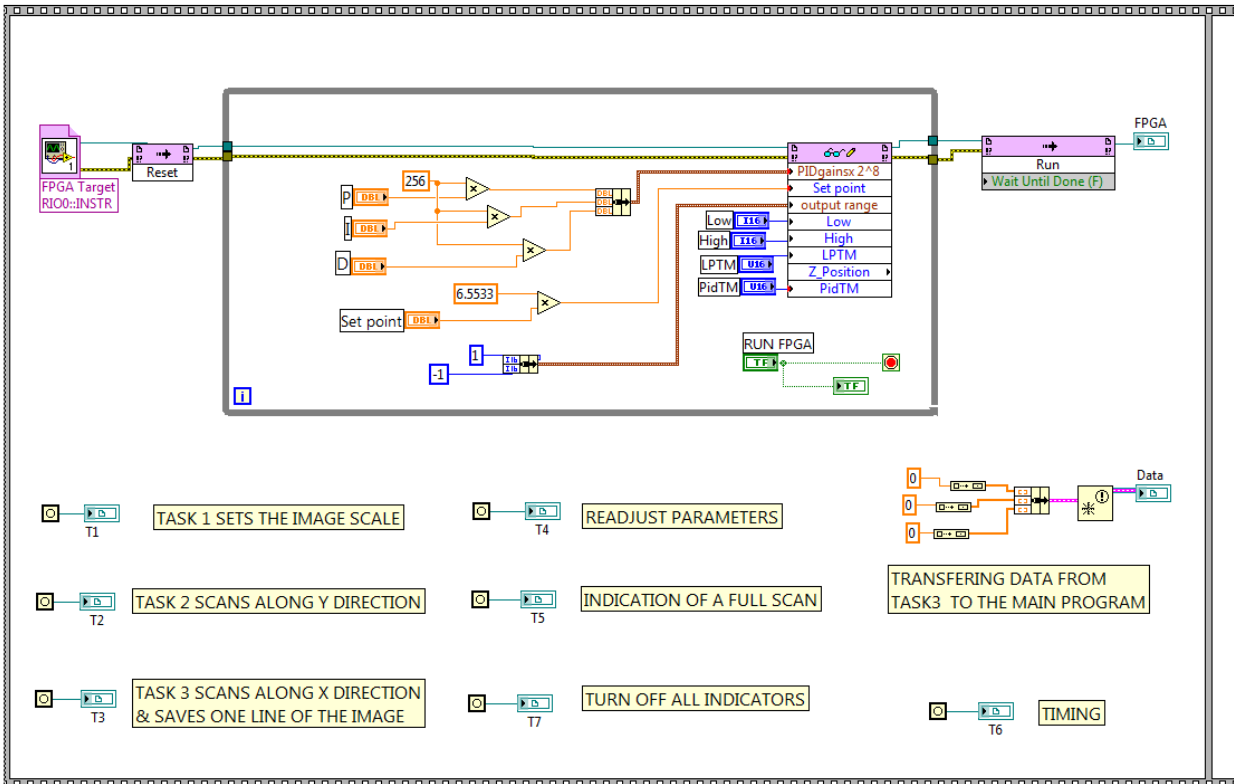


Fig3, The first step of the initialization, initializing the FPGA and assigning the parallel task handles.

As shown in Fig4 and Fig5, the second step of the initialization is to set and reset all the indicators of the controller. At the same time, the relative Z position will also be displayed. One extra function is that the operator can control the Z position manually. This is useful for nanoindentation and testing of the system. By click the Cnst button, one can input the Z_Cnst values to control the Z position manually. Emergency stop capability is also activated in this step. If the operator wants to turn off the SMP, he can click the EMERGENCYSTOP button, then the controller will reset all control signals to their default values and prompt the operator to turn off the system safely. Text To Speech (TTS) technology is also implemented at this step. The controller will tell the operator the current state of the microscope through a voice message. A lot of local variables are used in this program! How to use local variable efficiently and effectively can be found in Ref [4].

All the graph indicators used in this example have some kind of properties such as range and contrasts. These parameters can be set as static at the beginning of the program, or set as dynamically changed. Fig4 shows the default case. Fig5 shows how these parameters are dynamically updated according to real-time measured values.

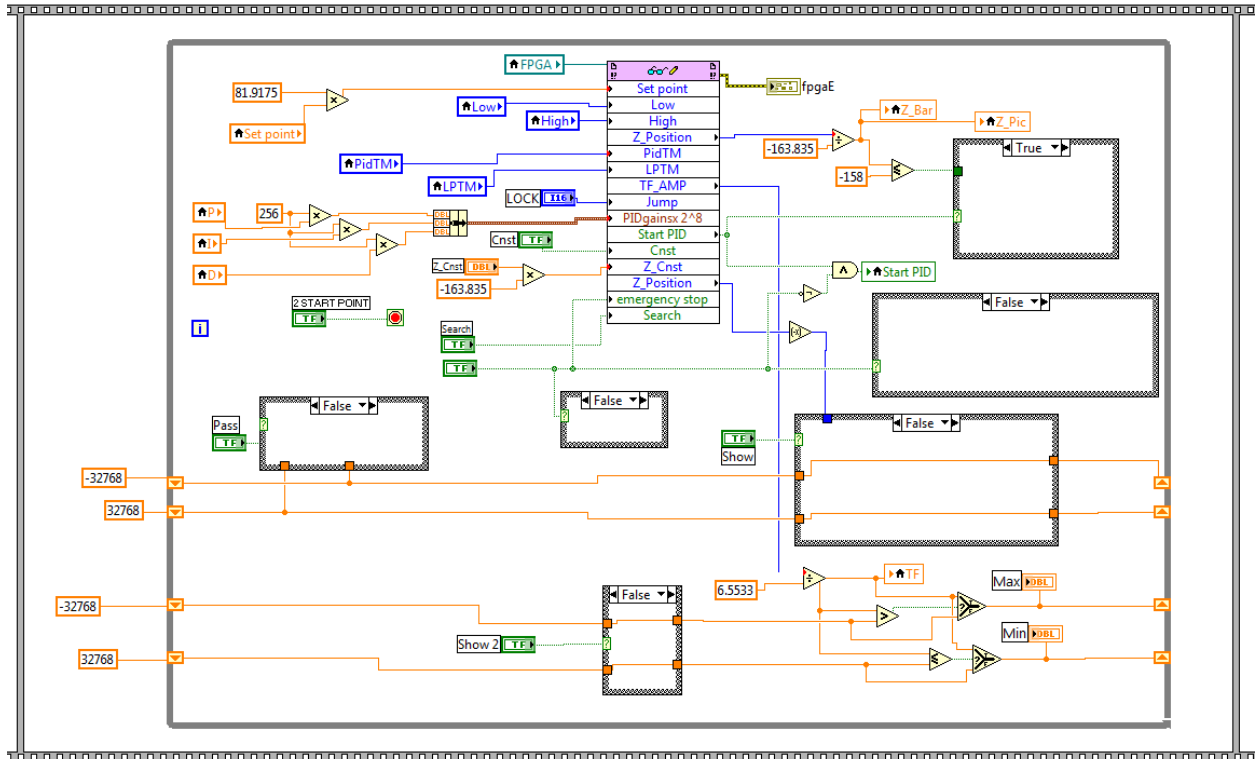


Fig4, The second step of the initialization, Setting the indicators and testing the system (case #1).

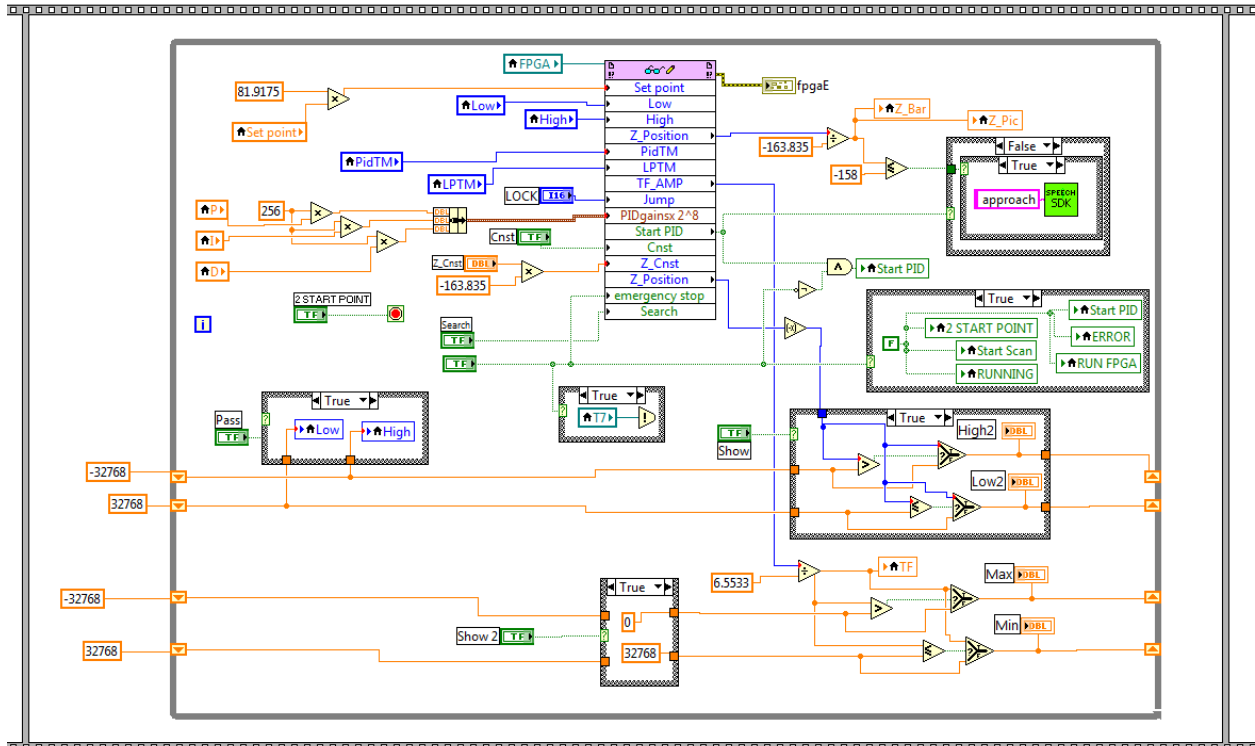


Fig5, The second step of the initialization, Setting the indicators and testing the system (case # 2).

The third step of the initialization is to move the probe to its XY start point. Two for loops are used to drive the stage to the XY start position gradually. After the probe moves to the start position, again a voice message will tell the operator the status of the SPM. Then, the operator is able to adjust the parameters again, before a scan.

One can see the Y controls are scaled up 163.835 times. Here, we use a 16-bit DAC, whose input goes from -32768 to 32767 digitally. Accordingly, the output side of the DAC goes from -10 V to $+10$ V. We use a high voltage (HV) amplifier, which has a gain of 20, to drive the piezo. After this scaling, the displayed control voltage will match the actual control voltage applied on the piezo. Now suppose the control input is 100V. The conversion equation is shown below.

$$100V(\text{input}) \frac{163.835}{32767} 10(\text{outputRange})20(\text{HV amp Gain}) = 100V(\text{output})$$

If one uses different DACs or HV amplifiers, this scale factor should be changed according to the specific numbers to keep the displayed values and actual output values to be the same.

The Z control is scaled up by -163.835 times. The reason is that if a positive voltage is applied on the piezo, the piezo extends along the Z direction, while the probe moves downward.

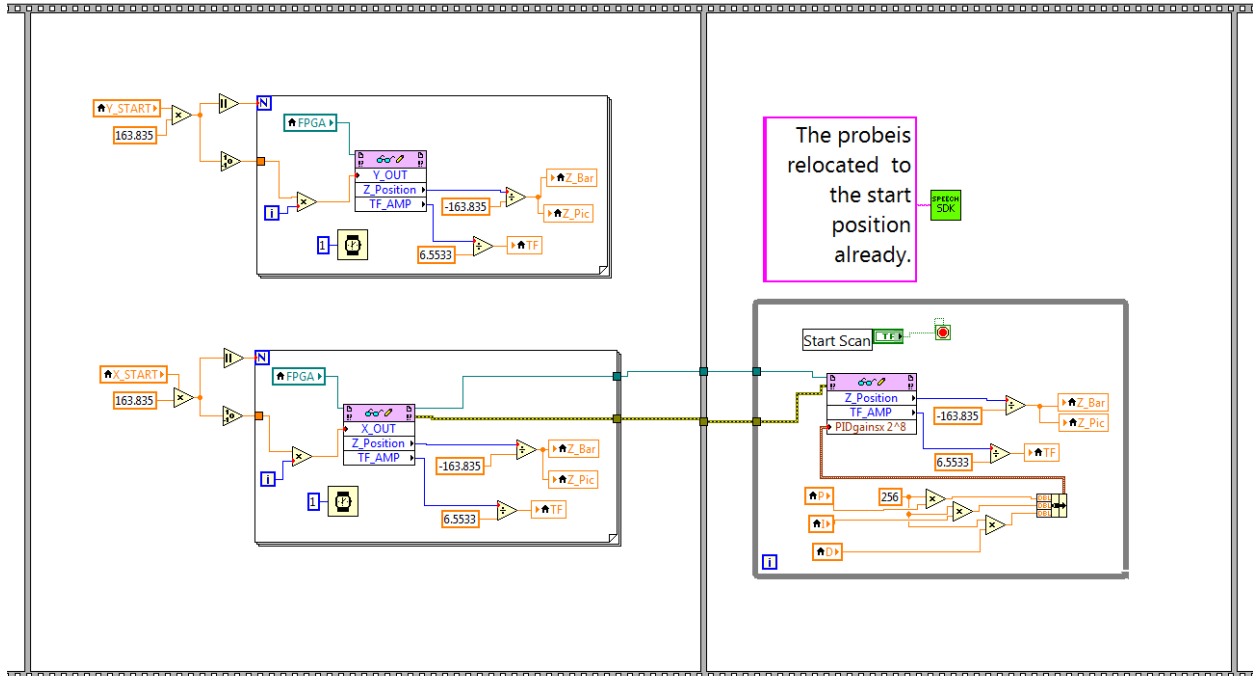


Fig6, The second step of the initialization, moving the probe to the start position.

4. Scan Control and the Image Display (portion II)

Fig7 shows the main structure of the scan control and image display part of the controller. This is the most complicated and essential part of the whole SPM controller. But we code this part with parallel tasks triggered by appropriate timing and relations. Eventually, the whole portion looks very simple. Another benefit of this kind of programming is that the modification and test of the system become very easy. Since house made SPMs are mainly used for training, teaching and research purposes, a simple software structure makes the teaching and modification of the program easier.

As shown in Fig7, the first step of the process is to rescale the image. This is important, because the operator may want to scan different areas, such 256x256 or 512x512 or 1024x1024, etc. This rescale process is carried out by the task T1. The second step is to scan along the Y direction, which is carried out by the task T2. The third step is to scan the X direction and save one line of the image, which is carried out by the task T3. The fourth step is to receive data from the task T3 and save one line of the image in the main program. Feedback nodes and arrays are used to build the images line by line. The feedback node examples can be found in Ref [5]. The fifth step is carried out by the task T4, in which the operator can readjust all the parameters again during the scan process. This is very important because, during the scan process, the operator can adjust the parameters to obtain the best images. After the execution of the for loop, a full image is obtained. Then the final step is carried out by the task T5, in which the operator is prompt to choose to turn off the system or to have a new scan.

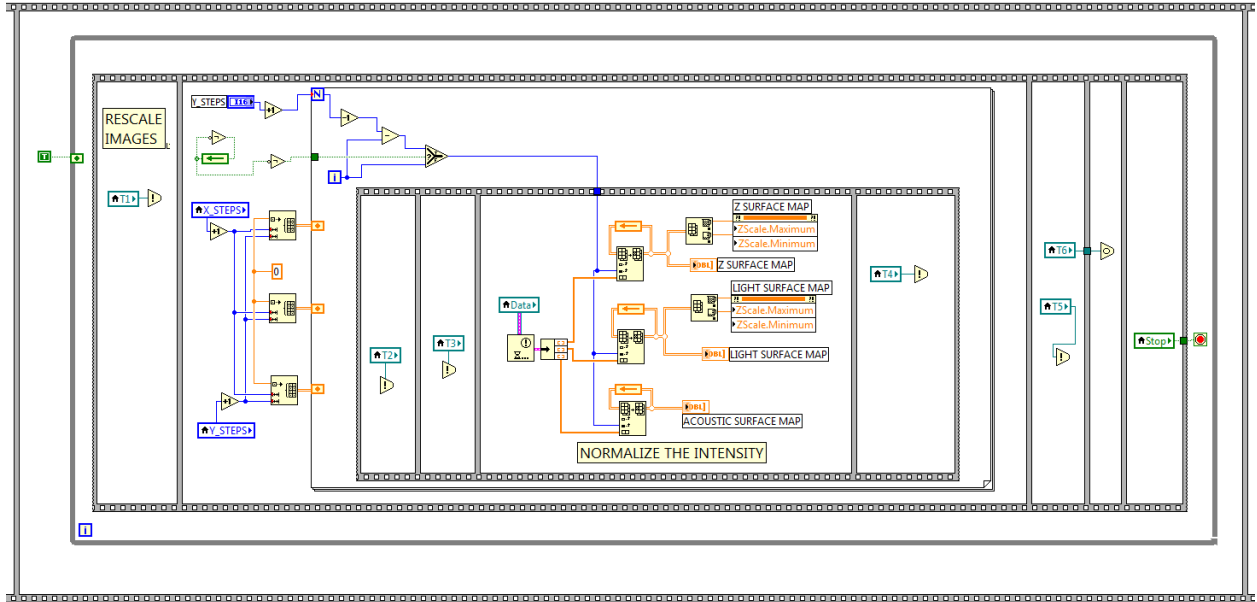


Fig7, Scan the sample and display the image.

A. The Parallel Task T1.

As shown in Fig8, the parallel task T1 is implemented with a while loop. As shown in Fig 2, in the main program all parallel tasks' handles should be initialized, one needs to make sure the main program is running before all the parallel tasks. In order to achieve this, a 20 seconds delay is added in the frame before the while loop. Property nodes are used to adjust the image scales and sizes. LabVIEW property node examples can be found in Ref [6]. One wants to readjust the sizes of the images and also wants to change the contrasts of the images, so this task is very important to obtain clearer images.

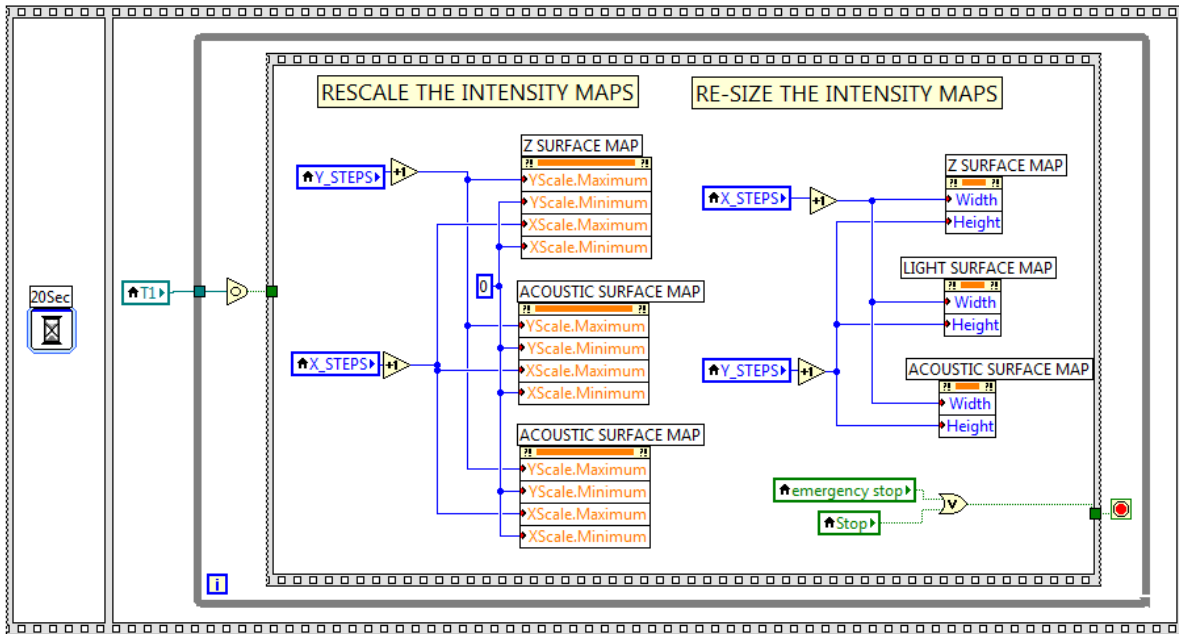


Fig8, the task T1, adjust the image sizes and scales.

B. The Parallel Task T2.

The parallel task T2 drives the probe scanning along the Y direction. Here a simple raster scan algorithm is applied. The output of the Y control is a ramp signal. A Boolean switch is used to control the signal to ramp up or ramp down. When the switch is TRUE the probe scans upward. When the switch is FALSE the probe scans downward.

As shown in Fig9, Y_STOP and Y_START control the Y direction scanning range. Y_FINAL is an indicator showing the real-time position of the probe along the Y direction. How these parameters will affect the operation of the SMP will be covered in section 8.

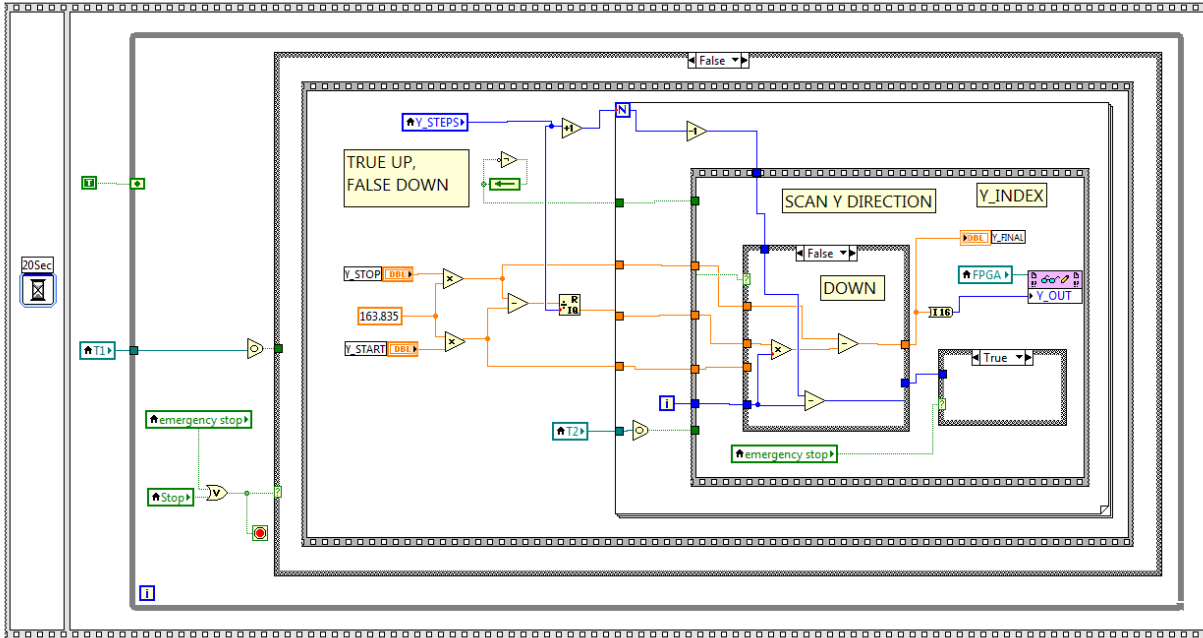


Fig9, the task T2, scan along the Y direction.

C. The Parallel Task T3.

The parallel task T3 drives the probe scanning along the X direction. Again, a simple raster scan algorithm is applied. A Boolean switch is used to control scan forward or backward. When the switch is TRUE the probe scans forward. When the switch is FALSE the probe scans backward. The scale equation of the control signals is the same as the one used in Section 4B. X_FINAL is the real-time indicator showing the location of the probe along the X direction.

The frame after the X-axis control signal output is a delay module. The Hold(mS) control will determine the actual delay time in milliseconds. This is an important factor to control the scan speed. Suppose one choose the delay to be 2ms and a 512x512 scan area. Then the scan rate is about 1 Hz. It would take about 9 minutes to finish a full scan.

$$\text{The Time to Scan One Line} = 512 \times 2 = 1024 \text{ ms} \approx 1 \text{ Hz}$$

$$\text{The Time to Scan the Full Image} = 1024 \times 512 \approx 524 \text{ Sec} \approx 9 \text{ Mnt}$$

The frame after the time delay is to display the real-time sensor signals and the topography. After one line scan, the one line image data will be saved into a notifier, which will trigger the main program to

read and save the data. After successfully saving the data, the main program will trigger the task T4. See Fig7.

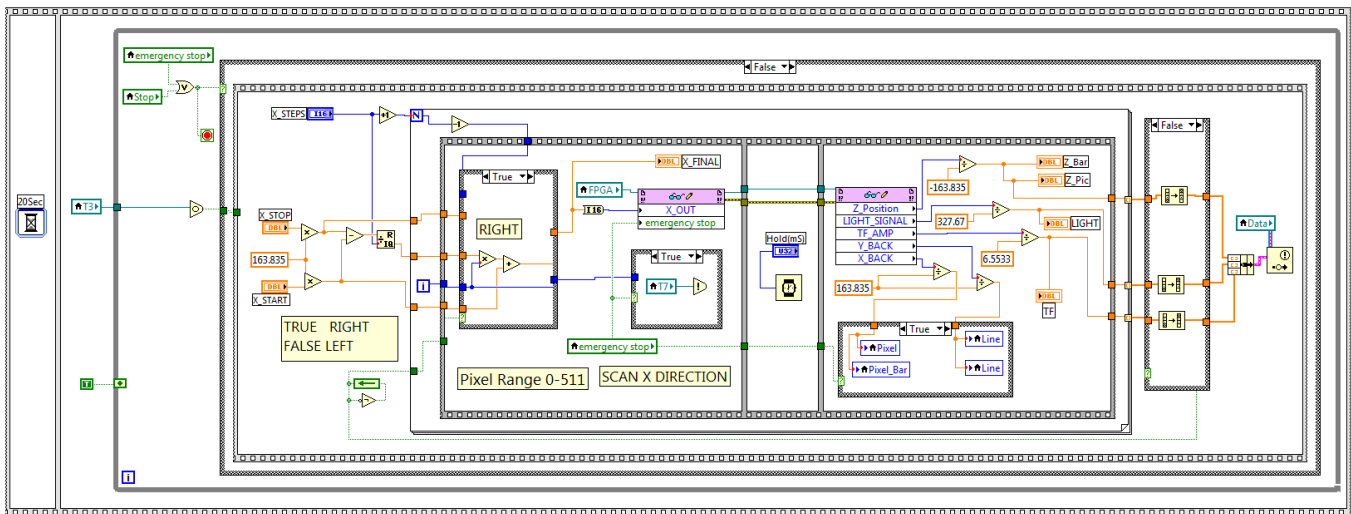


Fig10, the task T3, scan along the X direction and display the real-time signals.

D. The Parallel Task T4.

Usually, after one line of scan, the operator may want to adjust the parameters again to have a better response. As shown in Fig11, the task T4 allows the operators to readjust the parameters, such as P, I, D, Setpoint and feedback cycle time. One is also able to switch the scanning mode to manual control.

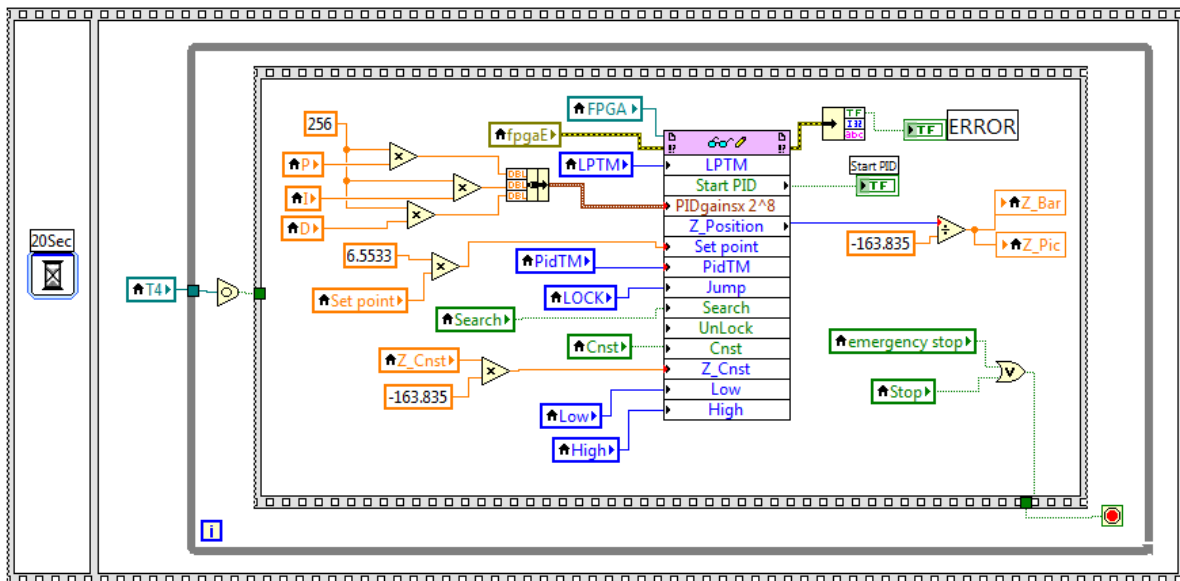


Fig11, the task T4, readjust controller parameters.

E. The Parallel Task T5.

The task T5 notices the operators that one scan has finished. After this, the operator may turn off the microscope or blow up a portion of the image to have a new scan. As shown in Fig12, the task T5 utilize the TTS technology and send the voice message to the speaker with a sequence. Another very import process carried out by the task T5 is to move the scanning head up. During the scanning process, the distance between the probe and the sample is several nanometers. After the scan is done, the probe should raise up from the sample, so the probe would not be damaged accidentally. The program achieves this by writing the Setpoint to the highest possible position, 32767. Driven by the feedback controller, the probe will raise up and disengage the sample.

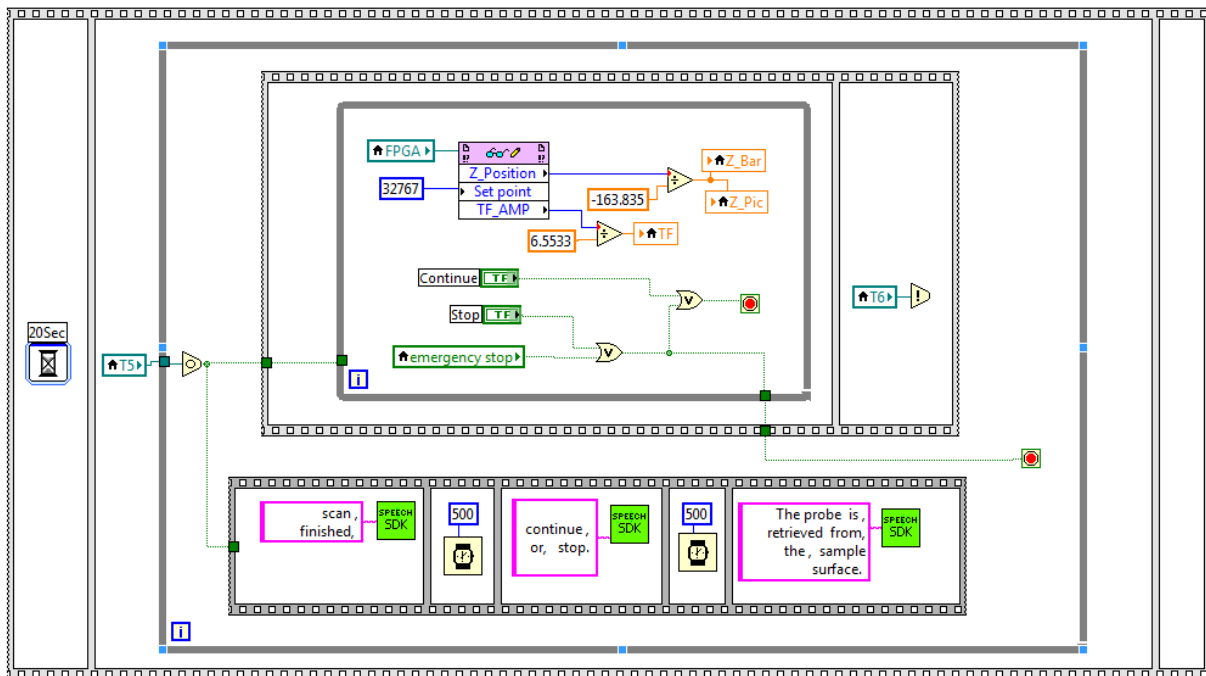


Fig12, the parallel task T5, which notices the operator the scan has finished.

5. System Shut Down (portion III)

After a full image scan, the probe raises up electrically by the piezo. This action will retrieve the probe from the sample by several micrometers. When a Stop command is issued in the task T5, the system shutdown procedure initiates. As shown in Fig13, the first step of the shutdown sequence is to mechanically move the scanning head up. This may take about 1 minute, eventually, the scanning head will raise up from the sample by about 10 millimeters. The second step of the sequence is to move the XY to their zero position. During these steps, the parallel tasks related to scan and imaging should be turned off. As shown in Section 4, all parallel tasks have turned off options. After the EmergencyStop or Stop buttons are clicked, if tasks T1, T2, T3, T4, and T5 trigger again, these tasks will be terminated. The third step of the sequence is to trigger the task T7 and make sure XYZ driving signals return to zero. The task T7 is used to turn off all the indicators and LEDs. In real machine operations, when the power is turned off, all indicators should be off. Running the task T7 to turn off all the indicators, let the virtual machine look like a real machine. See Fig14.

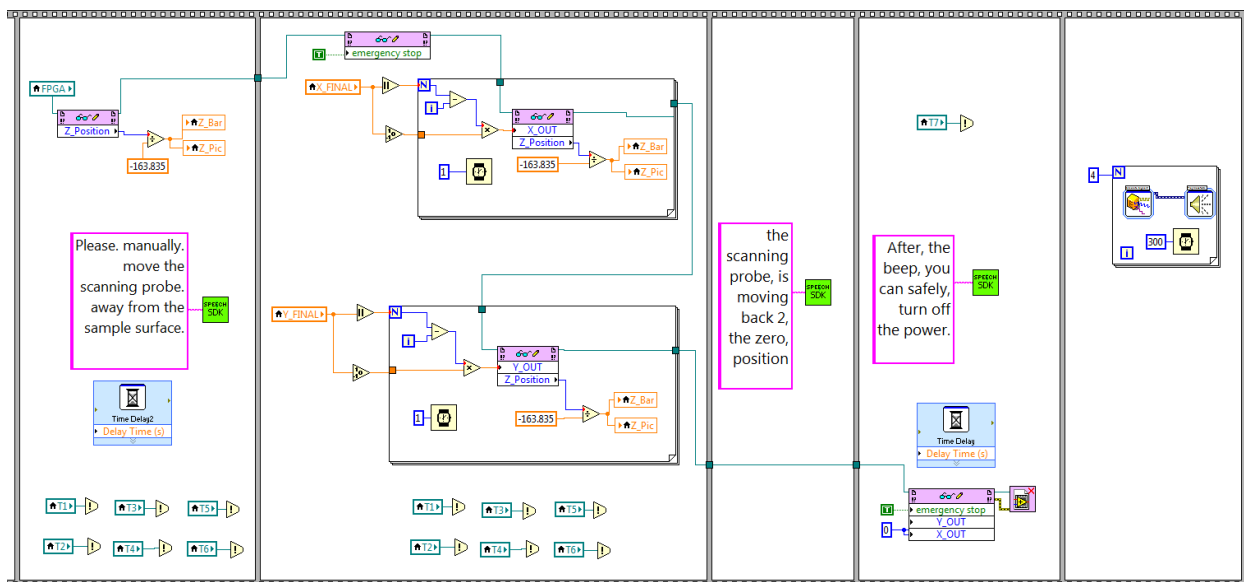


Fig13, system shutdown procedure.

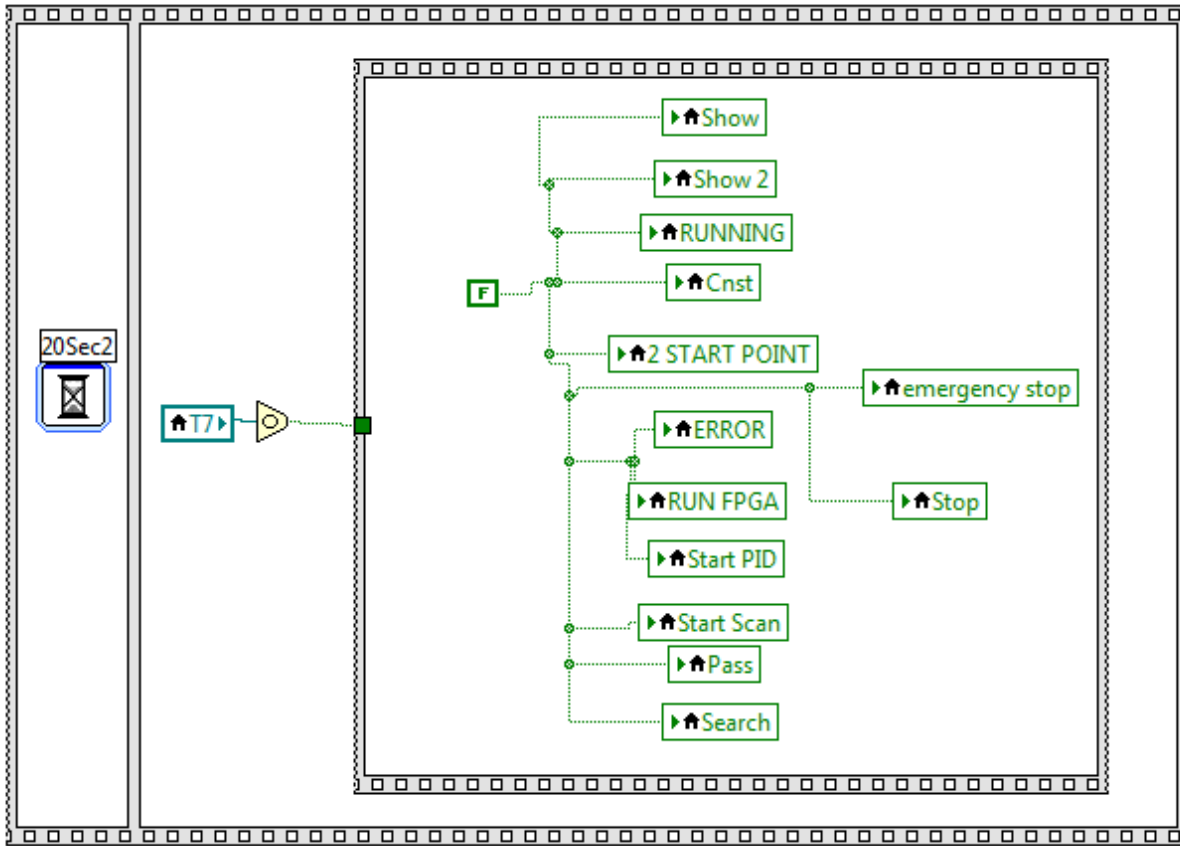


Fig14, the task T7, turns off all the indicators.

F. Speech SDK

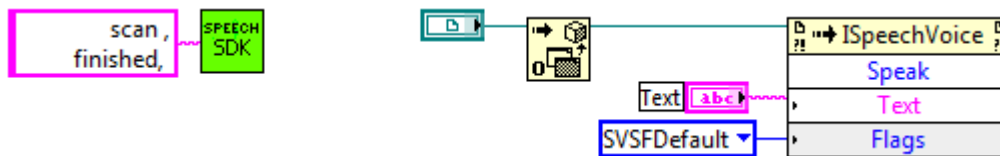


Fig15, the speech SDK

In order to facilitate the operations of the SPM, we utilize the TTS technology in this example. At various points, voice notices will be sent out by the speaker to prompt the operators to select the appropriate actions. Shows in Fig15 is the speech SDK, which is used to send out the voice messages. This SDK is quite simple. It uses an invoke node to call the TTS function embedded in the operating system and send out the text message in voice. Invoke note examples can be found in Ref [7].

6. The FPGA Module

Fig16 shows the front panel of the FPGA module. In this example, we use a NI7831R FPGA board to output the control signals and input the sensor signals. Most times, the users would not access the FPGA module directly. During normal operations, the main controller would talk to the FPGA by interrupt or FPGA nodes. Detailed LabVIEW FPGA program examples can be found in Ref [8]. Here, we only describe the functionality of the major controls and indicators of the FPGA module. The P, I, D, Setpoint, PidTM, Low and High are feedback control parameters. X_OUT and Y_OUT are real-time XY control signals. The Search button is used to engage the sample surface during the probe approach process. TF_AMP shows the real-time TF sensor signals. X_BACK and Y_BACK are used to show the probe's real-time XY position during an emergency stop incident.

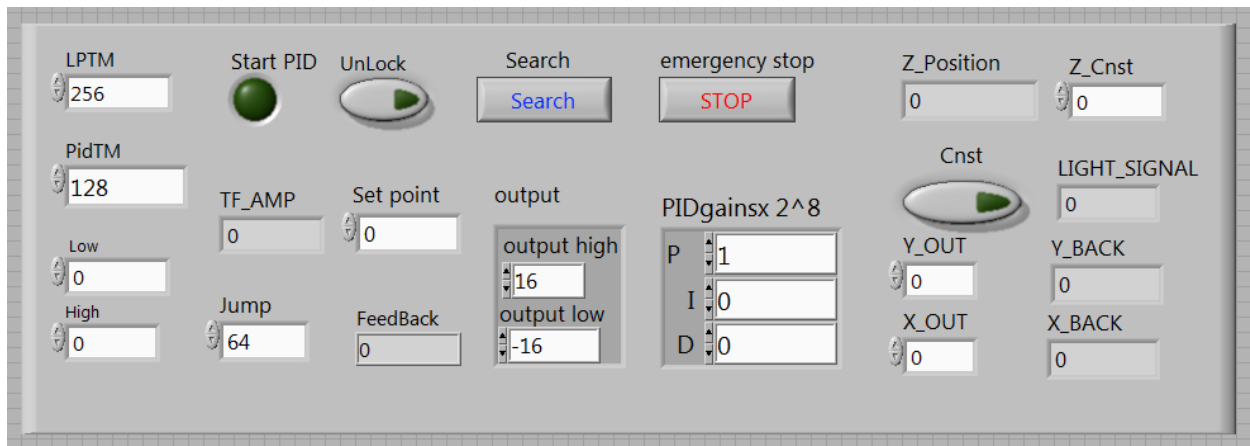


Fig16, The front panel of the FPGA module.

A. Z-Axis Raster Scan

Fig17 shows the Z-axis scan process. This part is relatively straightforward. The piezo is raised up at first. As the scanning head is lowered down mechanically, the piezo is moving downward and upward. Combining the coarse mechanical approach and the fine piezo sweep, the total effect of the probe movement would be the one as shown in Fig18. Along the Z direction, the piezo has a -30 micrometers to +30 micrometers range. This design is very important, especially when the probe is really close to the sample surface. The fine piezo sweep along the Z-axis may protect the probe from the sudden crash to the sample due to a coarse mechanical approach. During this whole process, the real-time TF signal is monitored by the controller as well. When the probe is close enough to the sample surface, the TF signal level will drop significantly. When the signal level reaches the Setpoint, the approach process stops, while the feedback control the Z motion will start.

Z-MOTION CONTROLLER

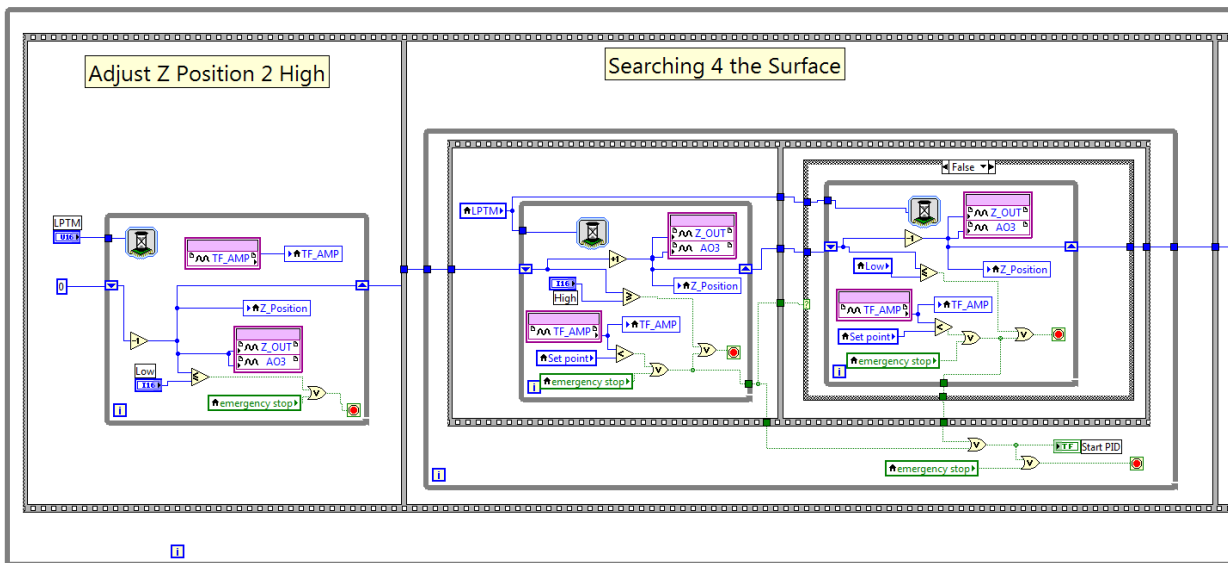


Fig17, Z direction ramp scan module.

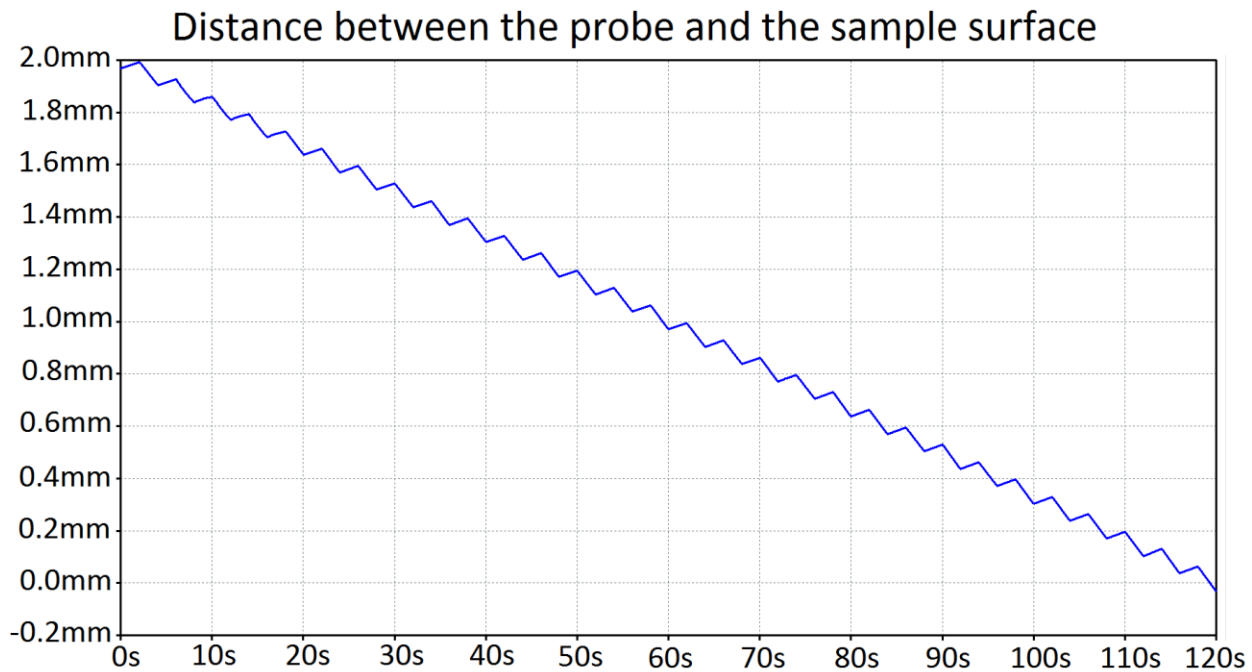


Fig 18, the probe approach curve.

B. Z-Axis Feedback Control

As shown in Fig17, during the Z-axis approaching process, the TF signal is monitored in real time. When the TF signal reaches the set point, the Z-axis raster scan process stops. Then the feedback control as shown in Fig19 will execute. We employ a PID feedback controller and manual controller inside of the while loop. LabVIEW PID controller examples can be found in Ref [9]. When the Cnst button is clicked, the manual controller is selected. The FPGA module will read the command from the main control through Z_Cnst, and output the Z control signal. In order to remove the possible oscillation, we set the output boundary of the PID controller through the Low and High controls. The frame after the feedback controller is used to drive the Z displacement to zero. When the operator clicks the Stop command, this frame will drive the probe to its zero position along the Z direction to bring the SPM to a safe stop.

Real-time control is an independent discipline. It includes a lot of topics, such as system modeling, system response, frequency analysis, controller design and etc. Control related stuff can be found in Ref [9][10]. Here, in this example, we use a simple PID controller. Generally, controller output range, feedback update rate, dead zone setup and possible integral windup are the most important issues a beginner should pay attention to. In this example, we use the Law and High controls to set up the controller output range. We use the PidTM to control the update rate of the feedback loop. The dead zone and integral saturation methods are implemented in the PID subVI. The control issues are out of the scope of this paper. Interested readers can check Ref [10].

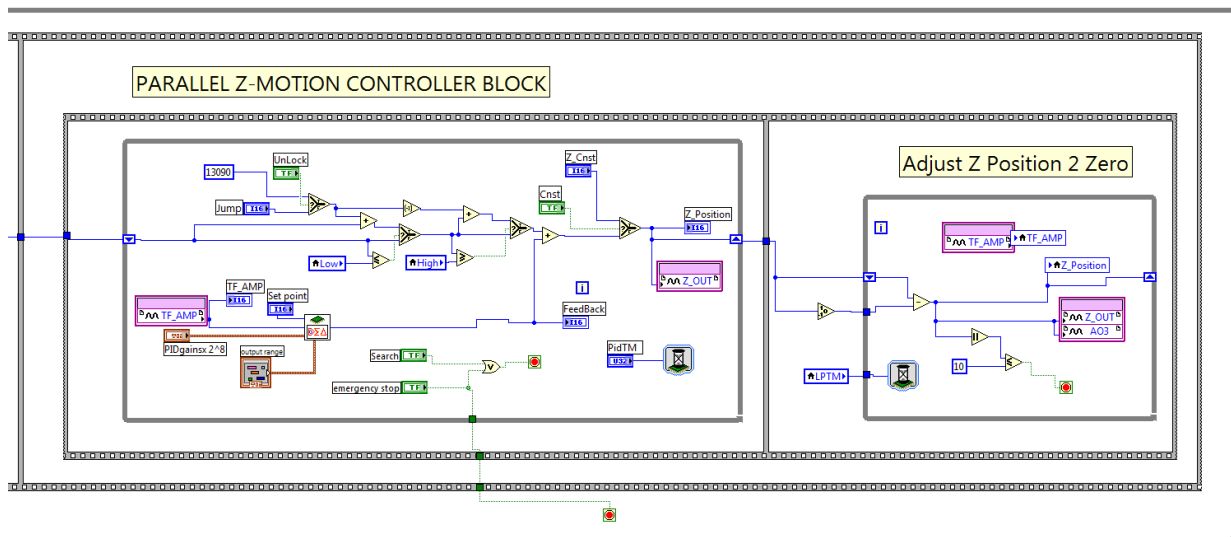


Fig19, Z direction feedback module.

C. X-Axis and Y-Axis Controls

In this example, the XY scans are controlled by the software. As shown in Fig20, the FPGA module here just reads the XY command from the main controller and output the XY control signals to the HV amplifiers. When the operator wants to stop the SPM, the two while loops after the XY control are used to drive the probe to its zero positions along the X and Y directions. The XY motion controller is running in parallel with the Z motion controller. The reason is that the Z motion controller updates much fast than XY motion controller. This way we can fully employ the parallel capabilities of the FPGA board.

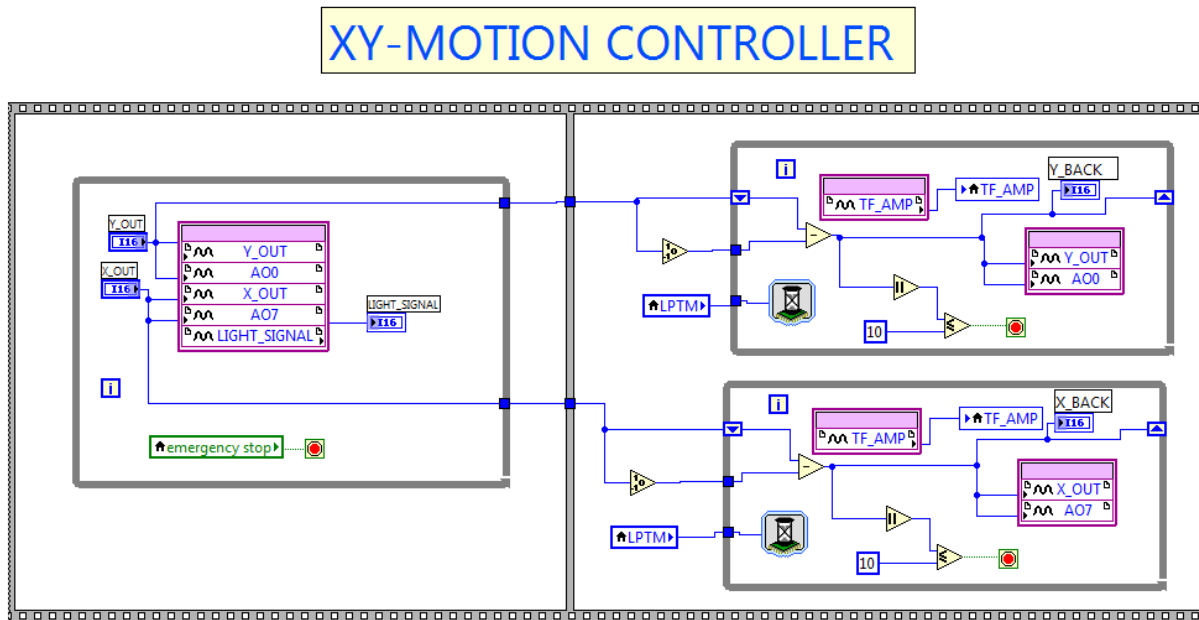


Fig20, XY control signal output module.

7. The Operation of the SPM Controller

The major part of the front panel of the SPM controller is shown in Fig21. Most of the controls are located around the upright corner. The indicators and displays are located on the left and the bottom.

A. Set Scan Related Parameters

Before running the SPM, the operator needs to set the X_STOP, X_START, Y_STOP and Y_START parameters. These parameters will control the scan area. Next, the operator needs to set the X_STEPS and Y_STEPS values. These will control the image size. Hold(mS) will set the scan rate, which is covered

in Section 4C. After the parameters are set properly, one can click the RUN FPGA and 2 START POINT buttons. Then the controller will drive the probe to its start position.

B. Find the Sample Surface

The next step is to control the Z position and find the sample surface. Initially, the probe is about 10 millimeters above the sample surface. In order to engage the sample, the operator should click the Search button. Then the scanning head will be lowered down mechanically and the piezo will also extend and contract periodically trying to find the surface. The relative piezo position will be shown by the Z_Bar vertical indicator. The distance between the probe and the sample surface will look like the one as shown in Fig18. During this process, one may want to adjust the P, I, D parameters and the Setpoint. In this example, a quartz crystal tuning fork (TF) is used to measure the interaction between the probe and the sample. If the amplitude of the TF is used to be the feedback signal, one can read the real-time TF signal from the TF window. Suppose the TF signal is around 5000, one can set the Setpoint to be about 4000. All the feedback parameters can be adjusted again in the future. This approaching process may take about one minute or two. After the sample surface is found, the Z_Bar indicator will stop moving up and down periodically. Instead, the Z_Bar indicator will stay around a fix position. Now all the feedback parameters and the Setpoint can be adjusted again to make the Z_Bar position stable. When all these are done, one is able to take a scan.

C. Take an Image

When the Start Scan button is hit, the SPM will start the scan process; and the images will be displayed line by line in the image window. The bead on the top of the image window will move left and right to show the probe's real-time X position. The bead on the left of the image window will move up and down to show the probe's real-time Y position. Again, during the scan process, one can readjust the P, I, D, Setpoint, PidTM and Hold(mS) parameters to obtain the best image quality.

D. Turn Off the SPM Controller

After a full image is obtained, one can click CONTINUE button to have a new scan or click the STOP button to shut down the SPM. At any time of the operation, one can click the EMERGENCY STOP button and shut down the SPM safely.

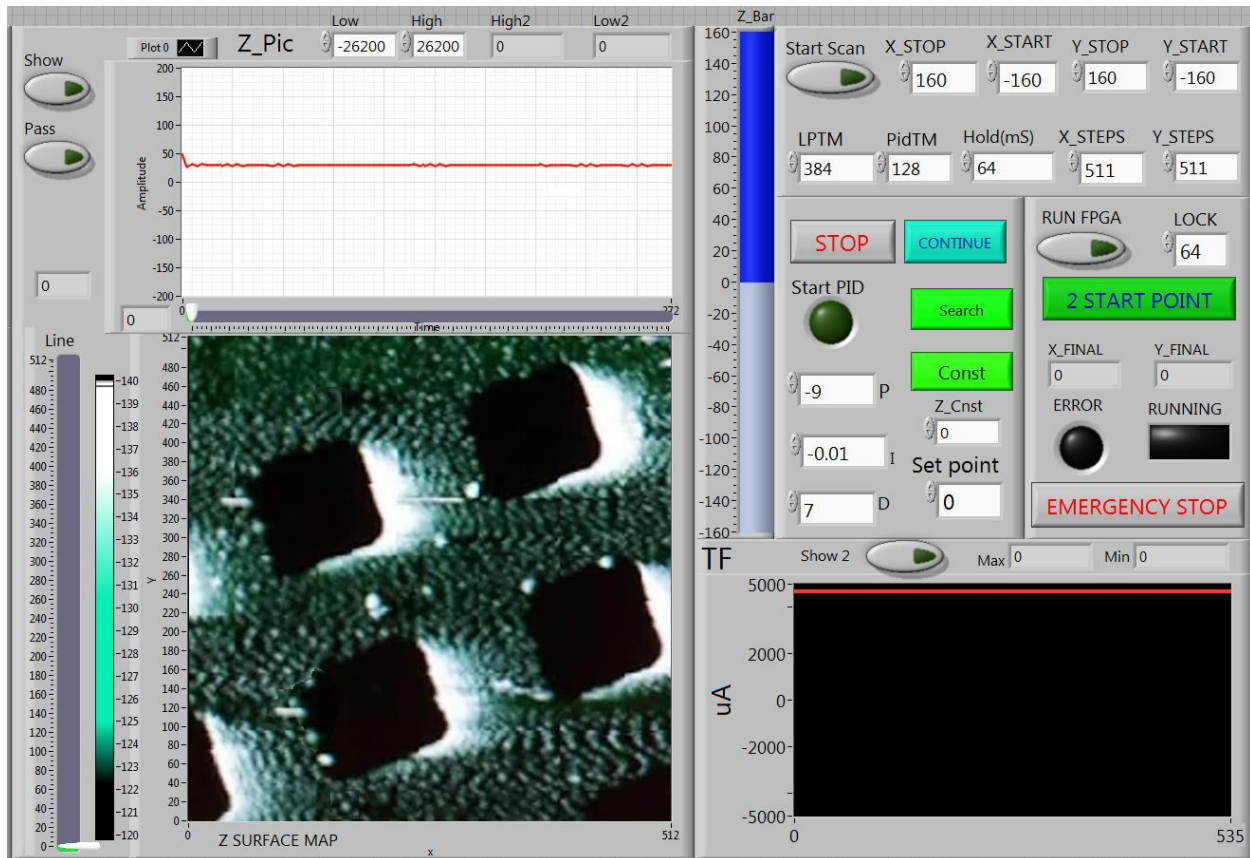


Fig21, the front panel of the SPM controller.

8. Conclusion

A scanning probe microscope (SPM) is a fairly complicated system, which has a lot of components, such as motors, piezo, signal process unit and etc. An SPM controller should control all the components properly and let them work together to finish the process.

To control an SPM effectively is not easy. Some beginners can barely program a LabVIEW controller with nested wires and messy icons. Eventually, the program becomes very difficult to read and maintain.

The purpose of this article is to reduce the barrier of developing house-made SPMs. Building a modular and elegant controller is a big part of this effort. The controller showed in this paper is intuitive and simple. A NI7831R FPGA board is used to output the control signals and utilize the Z-axis real-time feedback controls. Readers may use different hardware, such as microcontrollers to finish the real-time controls. Readers can follow the example showed in this article to build their own SPM controller with few modifications.

The structural method and parallel programming method applied in this article, are good examples showing how to achieve complex functionalities with simple programming. Local variables and feedback nodes are extensively used in this example to remove the nested wires across screens. Property nodes, parallel task blocks and LabVIEW occurrence and notifier are used to achieve modular programming. The main program is a sequence of three modules. All specific tasks are programmed as parallel modules. Finally, according to the process, all the modules are synchronized with LabVIEW occurrences and notifiers. In this example, each major module is accomplished in one screen area, without any cross screen wire. Each module can be modified separately, without influence the performance of other modules. Eventually, the whole program is very easy to read and maintain.

9. References

- [1] National Instruments, http://zone.ni.com/reference/en-XX/help/371361N-01/glang/synchronization_vis_funct/
- [2] Jianghua Bai, Jingwei Chen, John Freeouf, Andres La Rosa, A 4-layer method of developing integrated sensor systems with LabVIEW, *Journal of Measurement Science & Instrumentation*, 2013
- [3] Jianghua Bai, Andres La Rosa, "Essentials of Building Virtual Instruments with LabVIEW and Arduino for Lab Automation Applications", *International Journal of Science and Research (IJSR)*, <https://www.ijsr.net/archive/v6i5/v6i5.php>, Volume 6 Issue 5, May 2017, 640 - 644
- [4] National Instruments, http://zone.ni.com/reference/en-XX/help/371361H-01/glang/local_variable/
- [5] National Instruments, http://zone.ni.com/reference/en-XX/help/371361L-01/lvconcepts/block_diagram_feedback/
- [6] National Instruments, https://zone.ni.com/reference/en-XX/help/371361J-01/glang/property_node/
- [7] National Instruments, http://zone.ni.com/reference/en-XX/help/371361H-01/glang/invoke_node/
- [8] National Instruments, <http://www.ni.com/tutorial/14532/en/>
- [9] National Instruments White Papers, <http://www.ni.com/white-paper/6440/en/>
- [10] Gene Franklin, *Feedback Control of Dynamic systems*, Pearson Higher Education, Inc. 2010,