Design and Implementation of an All-Digital Atomic Force Microscope Controller

by

Kerry Yackoboski

A thesis
presented to the University of Manitoba
in partial fulfilment of the
requirements for the degree of
Master of Science
in
Electrical Engineering

Winnipeg, Manitoba, 1992

ISBN 0-315-77872-5

DESIGN AND IMPLEMENTATION OF AN ALL-DIGITAL ATOMIC FORCE MICROSCOPE CONTROLLER

BY

KERRY YACKOBOSKI

A Thesis submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

I hereby declare that I am the sole author of this thesis.

I authorize the University of Manitoba to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the University of Manitoba to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Abstract

A digital controller for an Atomic Force Microscope is designed and implemented. The traditional analogue Proportional Integral control loop is replaced with a digital one. Prior to describing the experimental work in this area, a short review of AFM is given, with emphasis on the requirements for the controller.

The controller is built around a Motorola 56001 Digital Signal Processor interfaced to an 80386-based Personal Computer and to a custom Printed Circuit Board. The Digital Signal Processor rasters the scanning tip over the sample surface, maintaining a constant force between the tip and the surface as it does so. As the experimenter's means of controlling the Digital Signal Processor, the Personal Computer runs a mouse-based graphical user interface program. The custom Printed Circuit Board enables communication between the analogue Atomic Force Microscope and the digital Personal Computer and Digital Signal Processor.

The controller is used to acquire an image of a diffraction grating. This image is compared with similar data acquired by a commercially available AFM controller and recommendations are made regarding further development and improvement of the digital Atomic Force Microscope controller.

Acknowledgments

I remain indebted to Professor Doug Thomson for the guidance and assistance he so generously provided during my stay in his research group. His help and remarkable patience as my thesis topic strayed from the original plan is greatly appreciated, and I would especially like to thank him for giving me the opportunity to work on the project described in this thesis.

An equal portion of gratitude is also due Dr. Gord McGonigal not only for his work on the All-Digital AFM Controller but also for his advice and aid on a wide range of topics both technical and recreational. Gord has guided and participated in the development of the Controller from the beginning, making many of the decisions regarding the controller architecture. The design and layout of the printed circuit board is entirely his.

I also thank the students and professors within the Scanning Probe Microscopy Group and the VLSI Group for freely sharing both their advice on the problem-du-jour as well as their general camaraderie, especially over the many unforgettable meals we shared on campus. With respect to the All-Digital AFM Controller, particular thanks go to Bradley Brown for his sage words on hardware debugging and the loan of his logic analyzer as well as the "swinging buffer" suggestion. Other notable contributions came from Jeff Dickson with assistance on virtually every computer subject under the Sun and Al McKay for the benefit of his considerable technical knowledge of electronic design.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Preamble

The Atomic Force Microscope (AFM)[1]is a recent invention capable of imaging surfaces with atomic resolution[1,2]. This is achieved by mounting a very sharp tip on a small cantilever and bringing this tip within several angstroms of the surface of the sample to be examined. Interatomic forces between the tip and the sample deflect the cantilever and this deflection is monitored and recorded as the tip is scanned across the surface, yielding a map of the contours of the sample surface. Atomic Force Microscopy[2] has been shown to achieve atomic resolution[2,9,19,46]. Several variations on this principle have been used to measure many different forces between the tip and the surface — contact forces[1,2], Van der Waals forces[21,50], magnetic forces[8,10,32] , and electrostatic[11,23,31] forces.

[1]A glossary of acronyms used is included as Appendix A.

[2]The acronym AFM commonly refers to both the field (AF Microscopy) and the instrument (AF Microscope), as is the case with Scanning Tunneling Microscopy and the Scanning Tunneling Microscope (both known as STM). The relevant meaning is taken from the context.

1.2 On the Origin of the Thesis

The invention of the Scanning Tunneling Microscope (STM) and the success it has subsequently enjoyed has spawned numerous other related scanning probe microscopes based on a sharp tip scanned over a sample surface. The most popular of these new instruments is the Atomic Force Microscope (AFM) invented by G. Binnig with Ch. Gerber and C.F Quate[1,2].

Gerd Binnig, who would later share in the Nobel prize for his earlier contribution to the invention of STM, was laying on the floor of his house gazing at his ceiling[14] one day in 1985 when the inspiration for the AFM struck. The texture of the ceiling reminded him of some surfaces he had imaged with the STM and he wondered why a surface could not be imaged with a force instead of with current as in the STM. This would permit the atomic scale imaging of nonconducting materials, something that could not be done with STM. Subsequent calculations of the forces between atoms showed that it would be easy to make a cantilever with a weaker spring constant than the equivalent spring between atoms[1]. If a sharp tip is fixed to the end of this cantilever, this tip could then be dragged across a surface much like a stylus along the modulated groove of a record album, following the contours of the surface. The AFM image could be created by monitoring the force on the tip created by the proximity to the surface, and if the force was small enough the surface of the sample will not be modified by the tip as it presses down. If this force were to be kept constant by a feedback mechanism as the tip is scanned across the surface, the tip would follow the contours of the surface as shown in Figure 1.1. This, of course, is exactly how the AFM is used. The inventors were able to

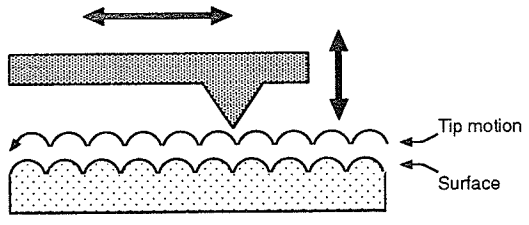

Figure 1.1: An AFM tip tracking over a sample.

exploit the experience gained in designing the STM, particularly the mechanisms

for rastering the STM's tunneling tip, in their design of the AFM. In comparison with STM the principles behind AFM are somewhat simpler and probably have helped to move AFM into the mainstream research community more quickly than was the case with STM.

In 1987 the Scanning Probe Microscopy research group at the University of Manitoba began work with a single STM and a NanoScope^{TM} I controller[74]; in 1990 an STM for use in ultra-high vacuum and an AFM head were acquired. As the group expanded it became apparent that the single acquisition system used to control the three microscopes in the laboratory was not enough and that the efforts of the research group would be better served if another controller were acquired. This new controller would be able to mimic the performance of the older NanoScope^{TM} I controller, but it should also be adaptable to new experiments. In May 1991 work began and some criteria for the new controller were set:

- The new controller should at a minimum simulate the analogue NanoScope^{TM} I used in the STM and AFM Laboratory.
- The system should be easily adapted to new experiments and to take advantage of unforeseen new techniques and new equipment. For instance, although the controller is intended for AFM it should be readily adapted to STM with minimal hardware changes or software rewrites.
- The new controller should cost less than Digital Instruments' new NanoScope^{TM} III[3].
- The feedback should be controlled digitally, for reasons discussed in section 1.3.4.
- Design and development time should be minimized. This was originally intended to be a part-time endeavour for two workers.
- The system should provide an image in real time with a minimum of hardware.

[3]Approximately $85,000.

- The controller should be able to raster the tip at between 5 and 30 lines per second. Slower speeds are often used in AFM, while the higher rates are used in STM.
- The controller should have a feedback bandwidth of at least 50 kHz for efficiency, stability, and convenience[80]. The high bandwidth is required for reasonable scan speeds.
- The proportional and integral feedback constants should be easily set to prevent feedback loop oscillation or underdamping.

This thesis describes the design and implementation of this system and is intended to serve as a reference for future use and development of the microscope controller.

1.3 Instrument Overview

The components of a system for atomic force microscopy can be roughly categorized into six subsystems, namely:

1. A flexible cantilever spring with a sharp tip used to probe the sample surface. The tip should possess a low mass and a high resonant frequency and have a small radius of curvature.
2. A method to detect the deflection of the cantilever due to forces between the tip and the sample surface.
3. An electromechanical scanning system to move a sample in a raster pattern with respect to the tip.
4. A feedback circuit to monitor and control the deflected cantilever and therefore the force between the cantilever and the sample surface.
5. A display system to convert the measured data into a meaningful and easily interpreted image.

6. Means by which to store image data for future reference or image processing.

This thesis concerns itself with items 3 to 6 on this list; however, for completeness' sake the first three systems will also be described.

In the recent past, most STM and AFM systems incorporated a computer system that functioned primarily to log data and display images. The scanning probe was rastered by triangle wave generators and external analogue electronics were used to implement a feedback circuit to control the distance between the tip and the sample surface. As systems became more sophisticated, triangle wave generators were replaced with digital-to-analogue converters, allowing computers to control the rastering of the scanning probe, and most systems integrated some form of digital image processing to enhance images. However, most acquisition systems still accomplished the tip feedback with analogue electronics.

This thesis discusses a new microscopy system in which all rastering, feedback, image data collection and image display is done digitally using two microprocessors and several digital-to-analogue converters (DAC's) and analogue-to-digital converters (ADC's). This chapter outlines the basic components of an AFM system and states the requirements of an AFM controller. Subsequent chapters describe more fully how solutions to these requirements are approached in our system, outlining the design and implementation of the hardware and software used. The programming details are reserved for the Appendices.

1.3.1 Scanning Mechanism

The STM and the AFM are able to achieve their extraordinarily fine movements of scanning probes through the use of piezoelectric ceramics. These devices are commonly made of lead zirconium titanate, known as PZT, widely used as an electromechanical transducer to obtain minute movements and for micropositioning in such applications as fluid control valves and optical interferometers[58].

Early STM's and AFM's used a wide range of piezoelectric scanning actuators such as a tripod of three PZT posts connected in three directions orthogonal to

each other[65]. This provided motion in three dimensions, with 2 bars providing X and Y raster scanning in the plane of the surface and the third post providing the Z motion perpendicular to the surface being scanned. However, this apparatus was mechanically complex since the three pieces had to be glued or clamped together and suffered from mechanical resonances at too low a frequency, large cross talk between piezo rods, and small scan range[56,41]

For this reason the tube scanner[67] was developed. This device is a hollow cylinder of piezoelectric material with a single electrode on the inner wall and four symmetrical longitudinal electrodes on the outer wall. These tubes are machined from a solid piece of piezoelectric material, metallized to form the electrodes, and polarized in the radial direction. The piezo tube is sectored into five separate electrodes to give orthogonal X, Y, and Z motion. The piezo has a solid electrode on the inside of the tube with the outside electrode divided into quadrants along the long axis of the tube. Vertical, or Z, motion is created by applying a voltage to the inside of the tube relative to the outer electrodes. Similarly, horizontal X and Y motion is controlled by two electrodes each. By applying a voltage to one of the outside electrodes and the negative of that signal to the opposite outside electrode bends the tube in one scanning direction, while applying similar signals to the other pair of outside electrodes bends the tubes in an orthogonal direction. While the first systems derived signals to raster the tip from simple function generators, in current systems the signals are generated digitally by the same computer used to acquire images.

The scanning tube used, provided by Park Scientific Instruments[75], has a nominal sensitivity of 240 Å/V in the horizontal X and Y directions and a sensitivity of 82 Å/V in the vertical Z direction.

1.3.2 Scanning Probes

A typical AFM scanning probe is usually a small sharp tip affixed to a flexible beam or cantilever that acts as a spring to keep the tip pressed against the surface as it follows the surface of the sample. This spring must be extremely flexible in order to track the surface at anything approaching an atomic level. If the spring

is very pliable it will yield maximal deflection for a given force, in order that these deflections may be accurately monitored and recorded more easily. However, this need for a soft spring is balanced by the need for a stiff spring with a high resonant frequency, in order to minimize the effects of ambient vibrations. The resonant frequency of the entire controller should be kept as high as possible to allow data acquisition at a reasonably quick rate; like any other control system, if it is operated at or beyond its resonant frequency the system is certain to oscillate out of control. The resonant frequency, f_o, of the spring system is given by $f_o = (1/2\pi)(k/m_o)^{1/2}$, where k is the spring constant and m_o the effective mass loading the spring. It is apparent that resonant frequency may be kept high by keeping m_o low as k is lowered to soften the spring; this will keep the ratio k/m_o low and therefore the resonant frequency high. For this reason AFM tips are usually small enough to make them somewhat difficult to handle, particularly when aligning them on a sample and adjusting the cantilever detector.

The first AFM used a tip made of a small diamond stylus glued to the tip of a 1 mm long gold foil cantilever. In order to minimize mass while maintaining maximum flexibility in the cantilever, current custom is to use a probe microfabricated from silicon nitride or similar materials using standard microlithographic methods[33]. These probes may be made in batch microfabrication processes and are now commercially available with the pyramidal tip and cantilever as an integral monolithic unit[24]. Figure 1.2 is a photomicrograph of one of these probes hovering several microns above a sample surface. These integral probes are made with a width and length on the order of 100 μm and a thickness on the order of one μm, yielding a spring constant of 0.01-1 N/m and resonant frequencies of 10-100 kHz[24,33]. Figure 1.3 shows such a probe consisting of a flexible beam and an integrated pyramidal tip. The tip itself is shown in close-up in Figure 1.3(a) while Figures 1.3(b) and 1.3(c) show the tip and the cantilever on an increasingly larger scale. Figure 1.3(c) shows that the flexible cantilever is not a single rectangular beam but is instead "V"-shaped to yield greater lateral stiffness. Several pyramidal tips are seen on each probe, although one would suffice. Figure 1.3(b) shows that the back of each pyramidal tip is actually hollow. Advantages provided by this

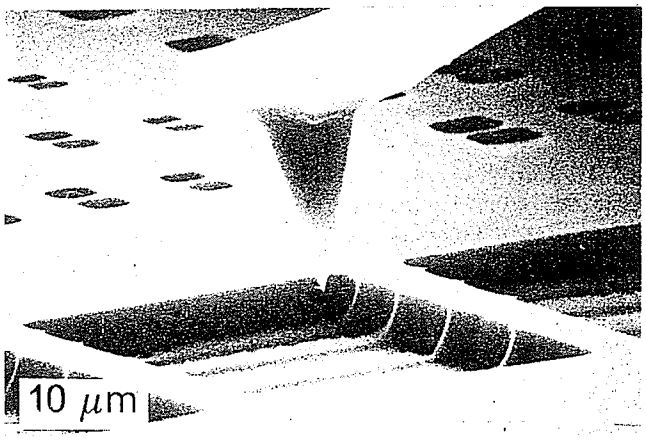

Figure 1.2: Scanning electron microscope micrograph of an AFM probe hovering over a sample, from Wolter *et. al.* [24].

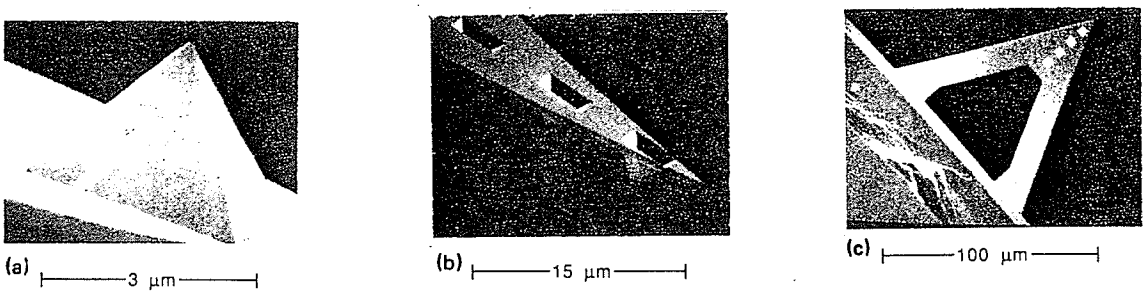

Figure 1.3: Scanning electron microscope micrographs of silicon nitride AFM probes. In (a) the tip comes to a sharp point with a radius of less than 300 Å. In (b) each cantilever is shown to have several tips, although only one is needed. In (c) four tips can be seen at the end of the "V"-shaped cantilever. Figures from Albrecht *et. al.* [33].

process include:

- Mass production techniques used in microfabrication brings cost per unit down to a reasonable level (approximately $10 per chip holding 3 probes).
- Mass production yields reliable and uniform tips.
- Making a tip, cantilever, and handling chip from the same material results in no strain due to thermal mismatch that often occurs when different parts are bonded together.
- The material commonly used, Si_3N_4, is unusually resistant to mechanical fatigue. A probe can be flexed at an angle 90° to its relaxed position without breaking.
- The manufacturing etching processes leave the back of the cantilever flat and smooth, useful for several methods of optical detection of the cantilever deflection.
- The tip may be made conductive by coating it with a metal and if the native oxide is avoided it may be used as a tunneling tip to the sample surface.

High resolution is achieved by the AFM when very small forces are placed by the tip on the surface. Typically between 10^{-7} to 10^{-11} N is applied[14,15,30], making the area of contact between tip and sample very small, the force probably being transmitted through a small irregularity a few atoms in size on the end of the tip. If AFM is to succeed as a non-destructive imaging tool scanning forces should be kept below 10^{-11} N for biological samples[12] and below 10^{-9} N for harder surfaces[29].

1.3.3 Probe Deflection Monitoring Schemes

The ideal cantilever monitoring mechanism would have infinitely small resolution, large dynamic range, good stability, and would apply no force to the cantilever. All that is required of the mechanism is that it generate an electrical signal that varies greatly with cantilever deflection. Several different methods have been used to

successfully detect deflection of the cantilever[21,26]; however here we will outline only the three most common schemes, widely proven to monitor the cantilever with subangstrom sensitivity: electron tunneling, optical beam deflection, and optical interferometry.

Tunneling Monitor

Binnig's original AFM[1] used a tunneling tip held stationary above the back of the cantilever to track deflection of the tip, as in Figure 1.4. Since the tunneling current between two conducting surfaces typically changes by a factor of 10 for

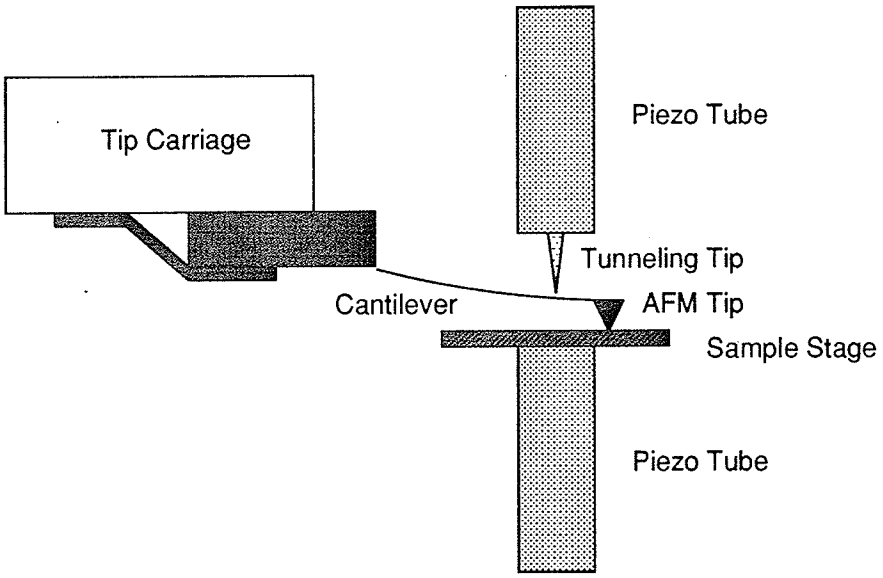

Figure 1.4: A tunneling tip used to monitor the position of an AFM tip.

every angstrom of separation between the two surfaces, employing tunneling current to measure distance has the advantage of being very sensitive to variations in distance. Feedback is employed to keep the tunneling current between the AFM tip and the STM tip constant by lowering or raising the entire apparatus, therefore maintaining a constant deflection of the cantilever. However, this detection scheme has the same disadvantage as STM; performance depends on the quality of the tunneling current and necessarily includes the $1/f$ noise inherent in the current[28].

Experience indicates that tunneling tips rarely, if ever, improve with time; rather, they become increasingly noisy as they are used. This method also relies on two tips (the AFM probe tracking the surface and the STM probe tracking the AFM probe) and is sensitive to the local tunneling properties of these tips, requiring the AFM probe to be made of or coated with some conducting material. The complexity of this scheme as well as its limitations mean that most AFM's now built rely on simpler optical methods to detect cantilever deflection.

Optical Beam Deflection Monitor

Optical detection methods generally take advantage of the sensitivity of light to its optical path length or path direction and fall into either interferometry[18] or beam deflection[46] categories, although other promising methods have been successfully attempted[26]. Both of these methods are capable of resolutions of 0.1Å. Optical detection is preferred over tunneling detection because it is reliable, simple, insensitive to the roughness of the cantilever, less sensitive to thermal drifts compared to tunneling methods, and does not incorporate $1/f$ noise. Noise due to photons from the laser bouncing off of the cantilever is negligible and is of the order of 10^{-19} N for a 1 mW laser beam[18].

The simpler of the two most common optical detection methods is the optical lever method illustrated in Figure 1.5. The light from a laser is focused onto the end of the cantilever probe and the reflected beam is measured by a photodiode that has been split into two segments. This photodiode acts as a position-sensitive photodetector (PSPD) by measuring the difference in light intensity absorbed by its two segments. When the cantilever is deflected, the light reflected on to the PSPD moves by the same angle, resulting in a change in the difference in light intensity absorbed by the diodes. Feedback is used to keep the reflected laser beam at a given point on the PSPD and consequently keep the cantilever at a constant deflection.

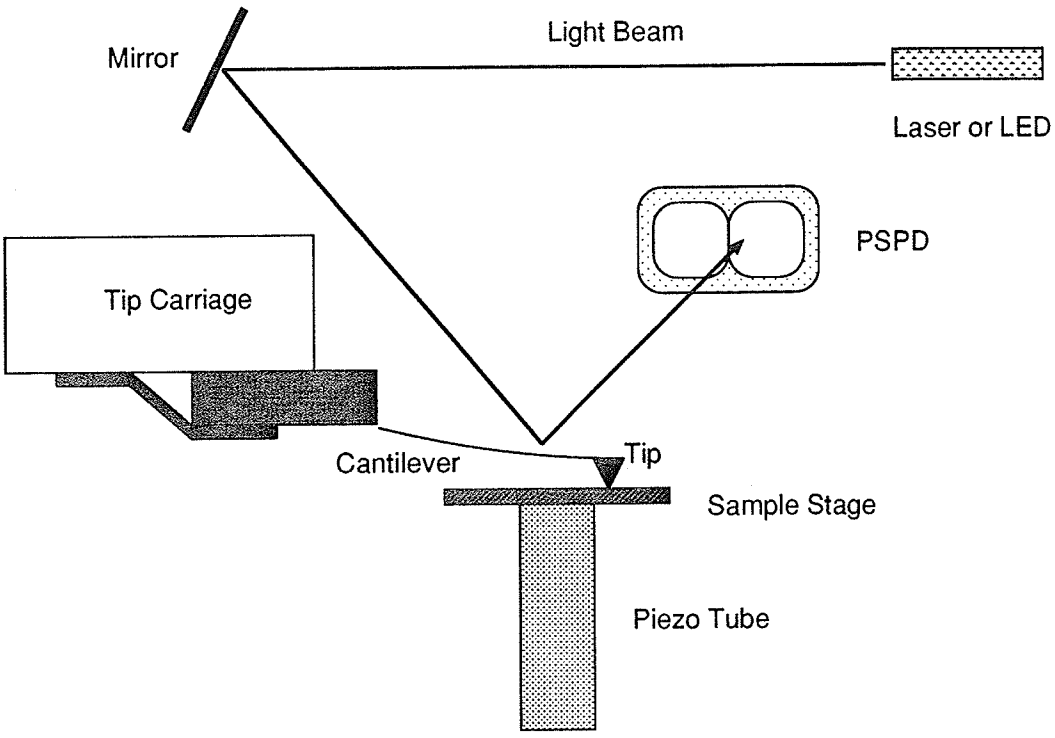

Figure 1.5: Optical beam deflection used to detect cantilever motion.

Interferometer Deflection Monitor

Although beam deflection is quite simple and reliable, more elegant interferometer arrangements have the advantage of not requiring mirror surfaces on the cantilever. This permits a wider range of cantilevers to be used and allows wire-like cantilevers to be used[8]. In this arrangement, shown in Figure 1.6, light from a laser or light emitting diode is launched into one end of a single-mode optical fibre connected to a 2x2 directional coupler, with one of the ends exiting the coupler held several microns above the cantilever. Approximately 4% of the light incident at this end is reflected back by the glass-air interface at this end. The light exiting the fibre enters the cavity between the fibre and the cantilever, striking the cantilever, with a fraction of this light reflected back into the fibre and interfering with the light originally reflected from the glass-air interface[69]. The interface at the fibre end is the reference reflector in the interferometer and the internally reflected light serves

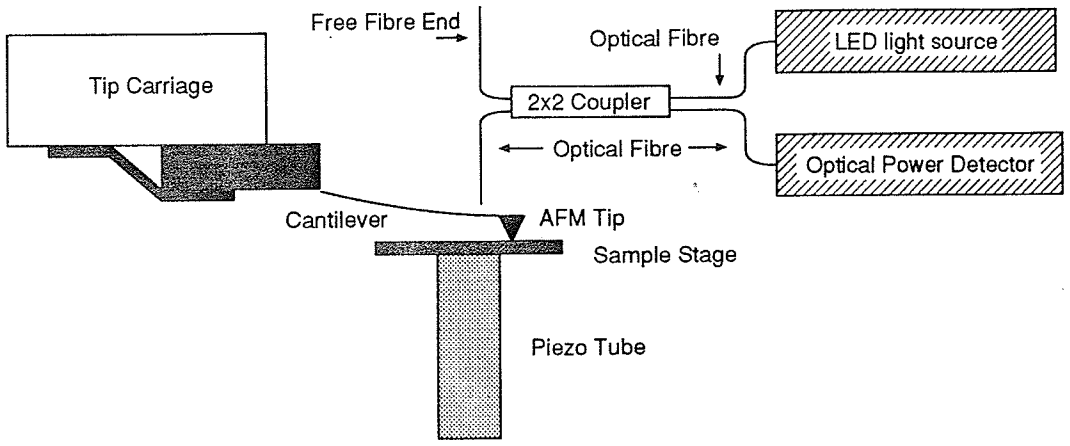

Figure 1.6: Optical fibre interferometer used to monitor the position of an AFM tip.

as the reference beam to the light scattered back by the cantilever. The total optical power propagated back through the fibre is determined by the phase difference between the two reflected beams[8], which is determined in part by the spacing between the fibre and the cantilever. The reflected light travels back towards the directional coupler, where it is split and half the power is sent to a photodetector and half back to the laser source. A feedback mechanism attempts to keep the optical power measured at the photodetector at a constant level.

One of the drawbacks of this system is that the laser is sensitive to the optical feedback generated by the interferometer, producing noise resulting in erroneous distance measurement. To avoid this problem, similar systems based on this principle are under development[69].

1.3.4 Feedback Mechanism

In AFM feedback is employed to keep the force of the tip on the sample constant. The force is measured by one of the means described in section 1.3.3 and the signal is compared to a reference chosen by the experimenter. The voltage controlling the Z piezo is then adjusted to raise or lower the sample with the respect to the tip, and the new height of the Z piezo is recorded and forms one picture element

in the AFM image. The simplest and most common feedback systems have been analogue circuits implementing the familiar proportional integral (PI) controller, known to combine zero steady state error with fast response time[37,85]. Integral control sums the errors in the control loop and compensates the Z position based on this error, while proportional control provides non-cumulative error correction in the loop by providing error correction in direct proportion to the difference between the measured force and a preset value. The derivative controller is shunned because of its inherent noisy characteristics. Although numerous articles have discussed moving different aspects of an AFM system into the digital realm, little has been said about carrying out the feedback digitally. In most systems the feedback governing the height of the tip is implemented using simple analogue circuitry even as computer systems become more complicated often control the X-Y motion of the tip[36,42,48,51,52]. Even as workers employ such advanced implementations as multi-microprocessor systems encompassing digital signal processors[40], various frame grabbers and frame processors[42,45], or even two entire computers[43], they retain the analogue feedback circuit which limits microscope flexibility while the researchers increase the overall complexity of their systems[34,43]. Some implementations have included a flavour of digital feedback but have done it in a limited manner such as employing a digital up/down counter to carry out digital integration with no proportional feedback[39,47,54]. It is desirable to take that final step and make the control of the tip height a digital phenomenon. Although Digital Instruments' NanoScope^{TM} III uses a Motorola DSP to carry out the feedback digitally, neither they nor other manufacturers[91,92] are about to share their proprietary information. Piner *et. al.* discuss their digital feedback system[35], but in general few workers have successfully replaced the analogue feedback circuit with a digital one[40,38], despite obvious advantages of such a system. It would give the experimenter:

- greater flexibility in experiments since new modifications would not have to be made to the equipment for each new experiment. New routines could be written and only added in when they are tested and known to work, eliminating down-time of the system.

- greater flexibility in feedback mechanisms, allowing the experimenter to break away from the traditional PI feedback controller if desired and install a new feedback algorithm through software.
- the ability to raster the tip in any direction through software. This would be very useful in lining up samples with respect to the scan pattern in order to acquire the best image. Instead of physically rotating the sample on the stage, the user could rotate the nominal scan orientation.
- the ability to hold the tip at a given height above the sample as some other variable is changed. This would be very useful in conducting voltage spectroscopy experiments, for example. In order to carry this out with an analogue feedback mechanism, new circuitry must be added to break the feedback loop and hold the tip at a constant height.

1.3.5 Image Display System

The earliest STM's produced their results in line scan form on an oscilloscope, with Y position and Z height combined to form vertical motion while the X signal provides horizontal motion across the oscilloscope screen. The result is a trace showing the profile of the tip motion, varying around a horizontal screen as it moves across the screen. A storage oscilloscope could be used to display an entire image as it is acquired. Figure 1.7 shows a photograph of typical output of this sort; this image is of highly-oriented pyrolitic graphite. These line displays are not very pretty and are sometimes not very understandable, but they do provide the experimenter with immediate feedback on the operation of the instrument. Since the data is drawn on the oscilloscope screen as soon as it is acquired by the controller electronics, the user can use it as a diagnostic tool to decide whether the AFM is behaving as desired, or whether the current image is worth acquiring and saving with the PC.

Soon after the first STM's were built, advanced display systems were used to display the data on computer monitors in false colours with simulated 3-dimensional perspective, or in a top-down greyscale display that approximates looking directly

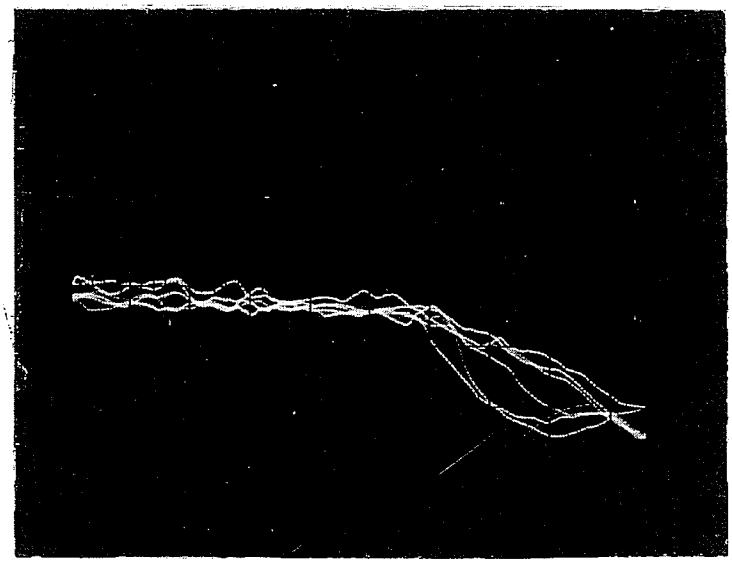

Figure 1.7: Line scan type of AFM image display.

down into the contours of the image. In the top-down display the pixel location represents the X and Y position of the probe while the intensity of the pixel shows the Z height; lighter shades are interpreted as being closer to the viewer and the darker shades deeper into the computer monitor. An example of the top-down greyscale display is shown in Figure 1.8. Although these more sophisticated displays make an entire image easier to analyze, they also have the disadvantage of not making it particularly easy to diagnose the behaviour of the microscope, since any trends within the image are not apparent until at least several lines have been acquired and drawn on the monitor. For instance, if the probe should start oscillating the line display will indicate this immediately, but with the top-down greyscale display several lines may have to be collected and drawn before the trend is apparent. For this reason the microscope should be able to provide an image in both of these formats.

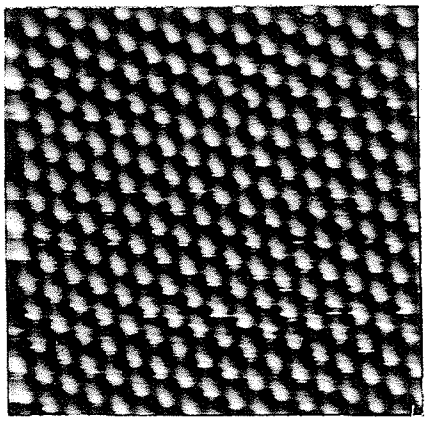

Figure 1.8: Top-down greyscale type of AFM image display.

Chapter 2

AFM System Overview

2.1 Introduction

To date scanning probe microscopy research conducted within the STM and AFM research group has been performed using commercially available STM's and AFM's, all interfaced to a NanoScope^{TM} I microscope controller manufactured by Digital Instruments of Santa Barbara, California. In addition to the AFM considered in this report and described in section 2.2, the Scanning Probe Microscopy Laboratory uses two other microscopes; the first microscope to be used here is an air STM supplied by Digital Instruments, and more recently an STM for operation in ultra-high vacuum has been acquired from Park Scientific Instruments of Mountain View, California. The NanoScope^{TM} I is a microscope controller consisting of an analogue electronic control unit and a storage oscilloscope. The storage oscilloscope is used to display images in real time as the data is being collected, but since the NanoScope^{TM} I is of relatively older design it makes no provision for permanent digital storage of data or of sophisticated image processing or display. Therefore an analogue-to-digital converter is used to interface the the NanoScope^{TM} I to an IBM-type Personal Computer running an in-house software program to conduct the data acquisition and display. For digital image processing the data is passed to another in-house program that runs on Sun workstations.

As described in section 1.2, the design and construction of a second controller

was undertaken. The design of the controller involved users of the system as well as the workers responsible for the design of the software and hardware. The user interface to the controller and the function of the system were specified in collaboration with the users in the hopes of developing a system that would be most useful to the intended users.

2.2 AFM Head

The microscope used is a model SFM-BD2 manufactured by Park Scientific Instruments of Mountain View, California[75]. The instrument is of the reflected optical beam type, as described in section 1.3.3. The microscope, shown in Figure 2.1, consists of a heavy cylindrical base, a detachable head, and a metal lid to go over the head and is about 20 cm high when assembled for use. The heavy base contains a stepper motor, the piezoelectric scanning tube, and the sample stage, while the detachable head houses a diode laser, a photodetector, and a mount for the scanning probe. The sample stage is actually a removable standard scanning electron microscopy sample stub that is mounted on a special stage mount protruding from the flat top of the base. The sample is taped or glued to the stage and the stage is clamped in place with an Allen screw. This stage mount is fixed to the free end of a single tube piezoelectric scanner to control the three-dimensional motion of the sample with respect to the tip. Although we frequently speak in terms of "moving the tip" over the sample, with this microscope the reality is that the tip is more or less fixed in place as the sample rasters back and forth.

The microscope head rests on three micrometer screws protruding through the flat level top of the base. These screws have magnetized ball bearings fixed to their ends; the smooth bearings ensure that the head will move up and down smoothly as the screws advance and retract, and they are magnetized to prevent the head from being dislodged once in place. The micrometer screws have 80 turns per inch and are arranged in a triangular pattern with the front-most screws accessible to the operator and the single rear screw driven by a stepper motor in the microscope base. The user may lower or raise the cantilever to the surface in a rough manner

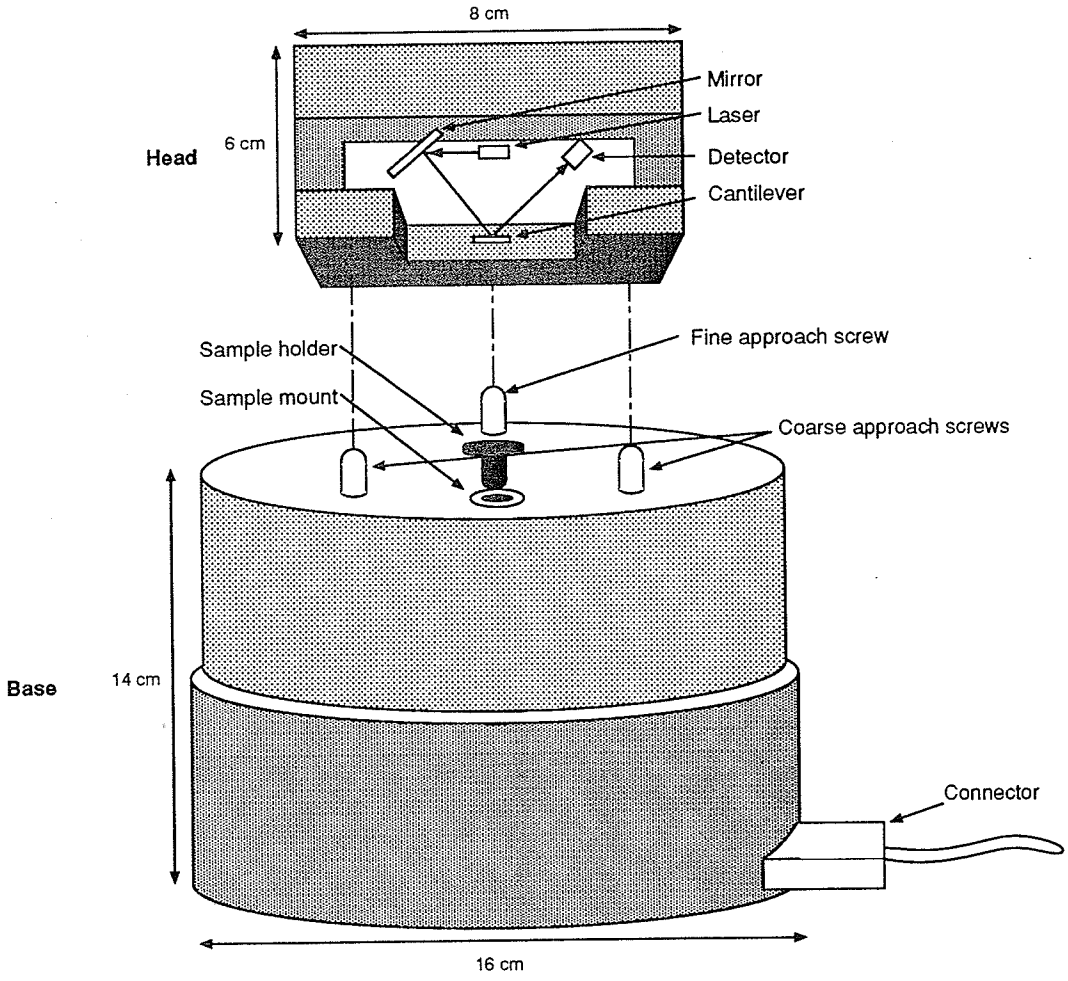

Figure 2.1: Simple diagram of the AFM head and base, adapted from [75]. The thumbscrews used to manually lower or raise the head have been omitted for clarity.

by turning the front two screws. Finer adjustments and the final approach of the tip to the sample is carried out by the rear screw driven by the stepper motor.

As the rear screw advances, it pushes the rear of the microscope head up, and this motion is translated into a downwards motion at the tip due to the levering action of the head. The sample is located just to one side of a line between the two manually accessible screws and this line acts as a fulcrum for the lever. When the microscope head is in place the tip is just slightly ahead of this fulcrum line, while the stepper motor screw is much further away on the other side of the fulcrum line. This places the lever at a mechanical disadvantage of about 0.1, so that the front of the microscope is lowered 1 micron for every 10 microns that the rear of the microscope is raised. As a result the fine advance screw turned by the stepper motor is able to lower the tip to the surface in steps of only a few angstroms. Problems may arise if the thread of the screw is rough or if the head has an unstable joint, leading to large movements in a single step.

Once the chip containing the cantilevers has been fastened within its holder on the microscope head and the sample stub has been fixed in its mount, the head is placed on the three screws. The magnetized ball bearings on the ends of the screws fit into grooves and indentations on the head, ensuring that the head seats itself securely on the bearings. The head itself contains the cantilever, the laser, and the position-sensitive photodetector (PSPD). The chip with the cantilever is held in place by the cantilever mount, and light from the laser reflects off a mirror and strikes the back of the cantilever. Light reflected from the cantilever is gathered by the PSPD, and the electrical signal out of the PSPD is sent to a preamplifier inside the AFM head.

The PSPD is actually split into two photodetectors positioned very close to each other, called A and B. The preamplifier board within the AFM head measures the output from both photodetectors and sends four signals to the control unit:

1. A + B
2. A - B
3. 10 * (A - B)
4. 100 * (A - B)

The (A + B) signal is used to set up the instrument as the tip is lowered to the sample surface, while the (A - B) signals are interpreted as a deflection of the cantilever from its nominal unbent position.

When the head has been seated on the stage the mirror may be adjusted to aim the laser light onto the back of the cantilever. After the laser beam has been aimed to shine on the back of the cantilever, the PSPD is adjusted to give a difference signal of zero. The tip is then brought closer to the surface by means of the micrometer screws. The operator may lower the 2 front screws thereby moving the tip downwards in a coarse manner. Once the tip is within a few microns of the surface the stepper motor is used to drive the tip closer to the surface. The stepper motor is controlled externally[1] by a circuit that advances the rear screw until a preset power level is measured by the PSPD. Once the tip is touching the sample, the amount of cantilever deflection is controlled by the set point of the control system.

2.3 The All-Digital AFM Controller

Four primary goals in the design of the system were that it

- enables the control of all data acquisition, image display, and data logging from a single interface.
- remain simple in order to keep it more reliable.
- operate in a manner that is appreciated by the operator. That is, the experimenter should be provided with a tool to do the job at hand in the most

[1]See Appendix B.2 for details on the control of the stepper motor.

desirable way. The user should always have control of the operating parameters.

- remain of open-ended design so that it can be easily modified in the future to accommodate new experiments, new equipment, and new concepts in scanning probe control. One way of ensuring that the unit remain maintainable is to design it in modules that may be individually upgraded or modified.

The design should also operate at high speeds for it to be useful; hard-wired systems are generally quite fast but are quite inflexible, requiring hardware changes in order to make even small modifications to the system. To maintain maximum flexibility, most of our instrument is controlled through software. The instrument achieves its high speed without sacrificing flexibility through the use of two microprocessors, specifically a Digital Signal Processor interfaced to a personal computer.

The experimental controller developed is a complete controller built around an Intel 80386-based personal computer (PC), a Motorola 56001 Digital Signal Processor (DSP), a printed circuit board designed and developed especially for this application, and the Park Scientific Instruments AFM head described in section 2.2. This controller manages the rastering of the tip, the measurement of the cantilever deflection as indicated by the signal from the PSPD, appropriate feedback for the Z piezo, the image collection, and the image display. Strictly speaking, the controller could be built with a single microprocessor since the simple calculations necessary to raster the tip and to update the Z piezo feedback can be done by an 80386 computer clocked at 33MHz. However, to be useful the controller must also have the 80386 microprocessor available to conduct the real-time display of the acquired data, properly shaded on a video monitor, as well as act upon user requests to vary the experiment by, say, adjusting a scan size or a current set point, or even by saving an image without interrupting the data acquisition. This is a lot to request of a single microprocessor, and for that reason two different ones are employed: a digital signal processor interfaced at one end to the AFM and interfaced at the other end to a personal computer. A block diagram showing the relationship of the hardware components is shown in Figure 2.2.

As Figure 2.2 shows, the deflection of the cantilever is sensed by the PSPD and the analogue difference signal is sent to the ADC where it is digitized and sent to the DSP. The DSP computes a compensated value for the new tip position based on the current tip position and the predefined control loop. This newly compensated value is then sent by the DSP to the Z piezo DAC, where it is converted to an analogue voltage and sent to the Z piezo. While the DSP is doing this it is also calculating the next X and Y tip positions and sending them to the appropriate DAC's in order to raster the tip. The rate at which the DSP does this is determined indirectly by the choice of a tip scan frequency by the experimenter, as explained in Appendix B.

The use of the DSP to directly control the AFM frees the CPU in the PC to be used as an interface for the user, collecting image data from the DSP and providing a real-time display of the data in both line scan and top-down grey-scale formats. The PC also handles all user requests to change parameters governing the operation of the AFM or to perform tasks such as saving data to a disk drive.

In effect, the DSP controls the AFM and the user controls the DSP through the PC. This is similar to the approach taken by commercial makers of STM's and AFM's, although they are using dedicated hardware that costs the end user upwards of $85,000.

2.4 AFM Control

The standard control mechanism used in scanned probe microscopies has been the common proportional integral (PI) controller implementing the equation

$$V_{\text{out}} = K_{\text{P}} * V_{\text{in}} + K_{\text{I}} \int V_{\text{in}}(t)dt. \tag{2.1}$$

The proportional feedback provides quick response to fast inputs, while the integral feedback provides zero steady state error when the gain K_I is high. The feedback mechanism used in the digital AFM controller is a digital version of the PI controller and is described in section 4.3. The equation takes the form

$$\text{Z} = \text{Proportional}_{\text{const}} * \text{error} + \text{Integral}_{\text{const}} * \text{error}_{\text{cumulative}} \tag{2.2}$$

The time constant of the feedback loop depends on the sampling speed and on the number of operations that the DSP must execute in order to arrive at the compensated output value. The time constant also depends upon the gain as in an analogue system, represented in the digital system by a digital multiplication of a constant times the error. Obviously a higher sampling rate allows the integral feedback to reach zero steady state error more rapidly[38].

The feedback mechanism should have a high bandwidth in order to better track the surface and to allow higher scanning rates. If the feedback is not fast enough to respond properly as the tip is scanned, the output signal applied to the Z piezo will not track the surface[41]. In addition, acquisition of images should be done in as short a period as possible to minimize the effects of $1/f$ noise, thermal drift, and piezoelectric creep.

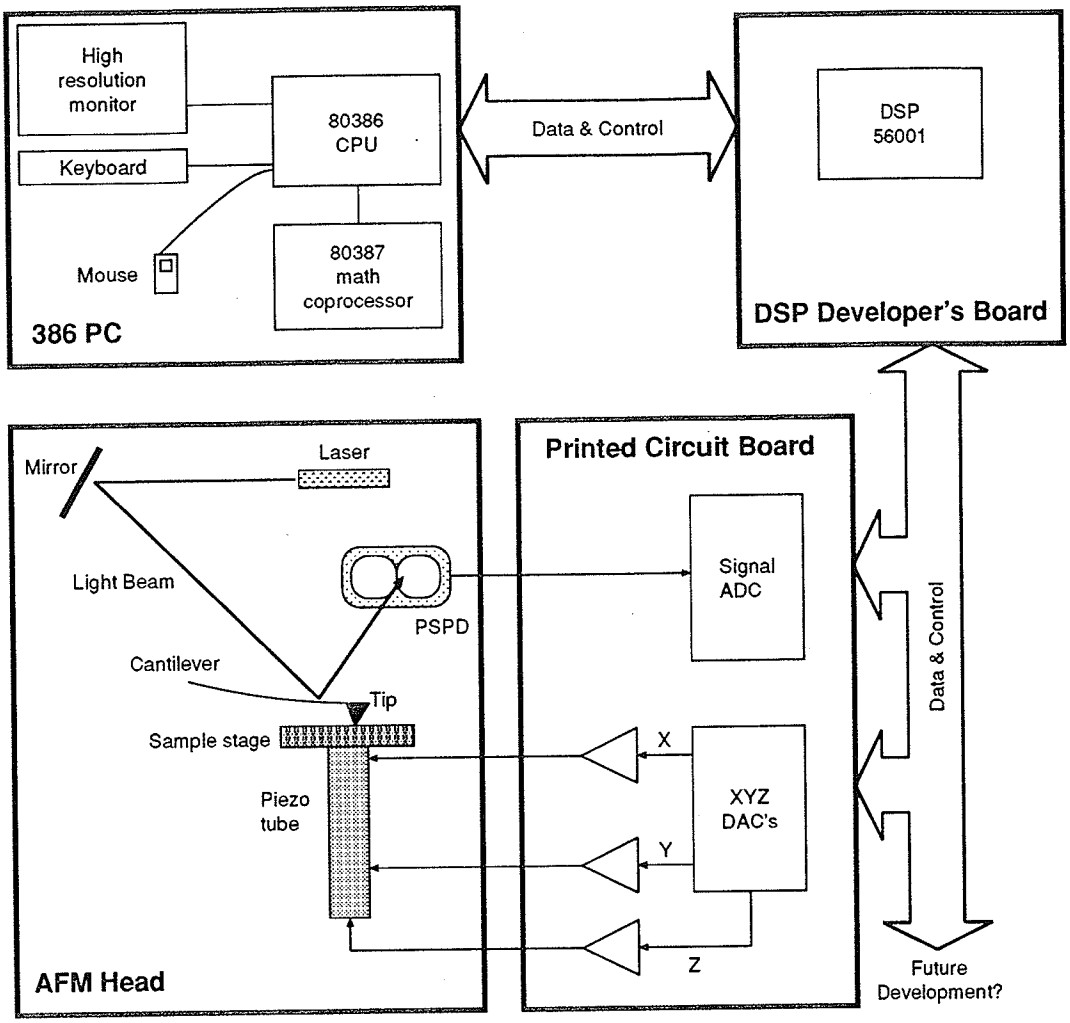

Figure 2.2: Block diagram of microscope controller hardware.

Chapter 3

Personal Computer Interface

3.1 Introduction

As described in section 2.3, the controller uses a PC as a user interface to the microscope via the DSP. The PC displays and stores the microscope image and also provides the user with a means of controlling the microscope by changing any of its operating parameters on the DSP, *e.g.* scan size. This chapter first gives an overview of the hardware and software used and describes the way in which the PC is used. Particular attention is then given to the data transfer between the PC and the DSP, to the user input functions, and to the data display functions.

The PC is really just a front end for the DSP and is blithely oblivious of the DSP's interaction with the AFM via the PCB. The PC accepts from the DSP image data to display on a video monitor, and the PC interrupts the DSP when the user wants to change some aspect of the microscope's operation. A mouse is used to accept input from the user, hiding the complexity of the PC–DSP interface from the experimenter. As the PC monitors the mouse and responds to user input, it also acquires and displays the image data as it is being transferred from the DSP.

3.2 Instrumentation

The host microcomputer must be fast and reasonably powerful, yet expandable and easily programmed. An IBM-style personal computer based on an Intel 80386 microprocessor with an Intel 80387 math co-processor chip installed for quicker and more accurate arithmetic was purchased[89]. The computer operates at 33 MHz and has 640 kilobytes of standard random access memory along with 1 megabyte of extended memory and 896 kilobytes of expanded memory. The operating system used, MicroSoft's MS-DOS version 4.00, does not leave the extended and expanded memory readily available to the programmer although the controller program does make limited use of the extended memory as described in section 3.3. For image display as well as user control of the program an analogue colour monitor is combined with an enhanced Video Graphics Array video card[90] capable of 800 pixels across and 600 pixels down with a maximum of 256 colours selected from a palette of 262,144. An 87 megabyte hard drive is available for data storage, and a standard serial mouse is available to handle user input. The personal computer is compatible with the Industry Standard Architecture (ISA).

An IBM-style PC was selected as the host computer primarily for three reasons:

- Firstly, interfacing external devices to the PC is comparatively simple and inexpensive since it is built to conform to the non-proprietary ISA standard. Many third party manufacturers sell low-cost cards to interface to the PC via the ISA bus, including Motorola with their DSP Application Development System for the PC. The PC also has more expansion slots available than does a normal Apple Macintosh^{TM}.
- Secondly, the PC itself is relatively low-priced in comparison to the Apple Macintosh^{TM} or the Sun Microcomputers SPARCstation^{TM}, both viable options for this project.
- Thirdly, previous experience with PC's and compilers for the PC within the Scanning Probe Microscopy Laboratory would speed development of the controller. In addition, the benefits of adding yet another type of computer to the Laboratory data acquisition environment must be weighed against such

disadvantages as burdening future Lab workers with having to learn an entirely new system with different problems to be overcome.

The application program was written especially for this controller and was designed to meet a few recognized general objectives in application software design[78]. These design objectives are all closely related, as described here.

- **Modularity** The functions of the application should be broken up into logically grouped smaller functions for maximum ease of understanding, maximum modifiability and maintainability, and ease of testing and debugging. As each component of the program is added it may be tested for correctness. In the future new functions could be added by writing new modules, and existing modules can be improved upon without rewriting the entire program or introducing new problems. For instance, it is anticipated that this controller will be used to implement various voltage spectroscopy experiments; dropping in new routines to implement this should not require rewriting large portions of the program.
- **Ease of Understanding** Use of a high level language programming language increases readability, shortens development time and simplifies future maintenance. Although assembly language programming yields faster performance, future maintenance work would be quite tedious and complicated and this in itself would cancel out any advantage that might be gained in the way of faster speed of the executable program.
- **Efficiency** The program must be fast and should execute within the 640 kilobyte limit set by MS-DOS. The PC program will be somewhat large and will have many tasks to do, some of them at the same time. For instance, the data from the DSP will be streaming in at a rate of several rows per second while the PC is drawing them to the screen and perhaps acting on a user request to change the scan speed. In order to cope the PC must handle all tasks in an economical manner. However, the desire to produce code that executes as fast as possible must still be weighed against all of the other factors itemized here such as ease of understanding.

- **Testability** The program will be slightly complicated and will depend on timing between the PC and the DSP; therefore it is probably unreasonable to expect that every routine will work perfectly the first time it is attempted. To complicate matters, the executable program and its associated data segments will be too large to easily run within any available debuggers on the PC. In order to efficiently debug the program it must be written and tested in sections, adding each debugged section to the aggregate program.
- **Maintainability** The program will undoubtedly need modifications in the future, either to remove undesired features, to change features, or to add new capabilities or to conduct new experiments. For instance, one very simple way to simplify "tweaking" of the program in the future is to hard-code in as few constants as possible; that way if, *e.g.*, the user interface screen has to be changed the programmer has to change values in one place only and they will take effect throughout the program, as opposed to forcing the programmer to change a value everywhere it occurs in a program.

3.3 Programming on the PC

The host program on the PC was written in Turbo Pascal Version 6.0 by Borland International[88]. This compiler for MS-DOS has been used in the Scanning Probe Microscopy Laboratory for several years now and the Lab has come to rely on it; an integrated editor and debugger are provided, but the compiler output is also compatible with the stand-alone Turbo Debugger sold by Borland. The compiler package includes several libraries that simplify programming on the PC, specifically in the areas of simple graphics and assembly language programming, both of which are essential in this application.

One particularly useful feature of the compiler is its overlay unit; overlays are parts of a program that share a common memory area, with only the parts of a program that are required at a given time loaded into memory. Overlaid parts can write over each other during program execution, storing an overlaid section in expanded memory and moving it into executable memory when it is needed.

This significantly reduces the program's total memory requirements and allows us to write a program much larger than we would have been able to previously, with negligible delays added by the time required to move a needed routine from expanded to executable memory. There is one caveat for the programmer; the overlaid functions must be grouped optimally in the overlay file to create minimal overhead in both time and memory when switching an overlaid function into execution memory. In the worst case, two overlaid units containing frequently-called routines can take turns writing over each other in main memory, meaning that the overlay manager is almost continually recalling the unit that was just overwritten. It is up to the programmer to group the overlaid functions so that procedures that are frequently called together or that call each other are compiled and linked in the same overlaid unit.

Due to the segmented nature of the PC's memory, the Turbo Pascal compiler does not directly support data structures greater than 64 kilobytes in size; however, this limitation is easily worked around through the use of data pointers. This technique is necessary for the controller because a single AFM image needs 131,072 bytes of memory[1]. The data structure used to store the image data is compatible with that used in the older acquisition systems in the Laboratory: data type *RowPtr* is a pointer to a single row of image data stored as a one-dimensional integer array of 256 elements, and data type *Image* is an array of 256 of these *RowPtr*'s. Two of these *Image* data types are declared so that the controller may hold two complete images in memory at once: the two image variables are called *IM1 and IM2*. A pointer to an *Image* array, of data type *ImagePtr*, is called *IMG*. Variable *IMG* can be used to access the different images just by redirecting it to the "other" image. This structure lends itself to a "swinging buffer" arrangement, where the pointer IMG starts out pointing at IMG1 and writes the incoming image data to IMG1 row by row. When IMG1 is full, the program checks a flag to see whether IMG2 can be written to; if it can, the IMG pointer is moved from IMG1 and points to IMG2, filling IMG2 row by row. If the flag indicates that IMG2 is locked, the program starts to write over IMG1 again. The primary advantage of this scheme becomes apparent when the problem of saving an image to the disk is considered.

[1] (256 points)^2 * (2 bytes per point) = 131,072 bytes

The PC waits until an entire image is collected and starts to save it, but as the data is written to disk the DSP is still sending new rows of data and overwriting the image as it is saved. This could be avoided by stopping the scan until the data is saved, but the swinging buffer method is preferable since it does not slow down the experiment.

3.4 Data Transfer

The PC deals solely with digital quantities; all values it works with and shares with the DSP are digital ones, leaving the DSP the problem of conversing with the AFM in analogue voltages. All data transfers between the two microprocessors are initiated by interrupts; that is, when one processor wants to send data or information to the other it does so by first interrupting the task that the other processor is conducting. The DSP sends only two types of information to the PC:

1. the row of image data that has most recently been collected by the DSP, and
2. the position of that row within the entire image.

On the other hand, the PC must inform the DSP of the values of virtually all of the variables involved in scanning and feedback, such as feedback constants, scan sizes, and scan speeds. The procedures used for each of these data transfers (DSP $\Rightarrow$ PC and PC $\Rightarrow$ DSP) are described in this section.

The PC uses the registers listed in table 3.1 to communicate with the DSP. Although the purpose of each register is summarized here, the details regarding the use of each register appear as required in the following sections.

- **PC Interrupt Mask Register**: holds the mask for hardware interrupts of the PC. When a bit in this byte is set to 0 the corresponding IRQ is enabled; that is, IRQ 7 is unmasked when bit 7 of the Interrupt Mask Register is 0.
- **PC In Service Register**: used to signal that an End of Interrupt (EOI) has occurred. The final step in an interrupt handling routine is to send an EOI signal to the In Service Register; if this is not done, the interrupt controller

Port	PC Address
PC Interrupt Mask Register	21_{16}
PC In Service Register	20_{16}
DSP Interrupt Control Register	300_{16}
DSP Command Vector Register	301_{16}
DSP Interrupt Status Register	302_{16}
Transmit / Receive High Byte	305_{16}
Transmit / Receive Middle Byte	306_{16}
Transmit / Receive Low Byte	307_{16}

Table 3.1: Addresses of the PC's control and data ports.

sees that an interrupt has been requested and, not realizing that it is seeing the interrupt that was just handled, immediately calls the routine that just finished, in a never-ending loop.

- **DSP Interrupt Control Register**: controls the DSP's Host Interface interrupts, most importantly whether or not the DSP is allowed to interrupt the PC when it wants to transfer a row of image data.
- **DSP Command Vector Register**: is used by the PC to force the DSP to execute a vectored interrupt. The PC specifies which interrupt vector the DSP is to execute by writing the starting address of the interrupt on the DSP in the Command Vector Register.
- **DSP Interrupt Status Register**: the Interrupt Status Register is used to verify that the DSP generated an interrupt. If the Interrupt Status Register does not have the proper bits set high, the PC may assume that the interrupt was triggered by noise and resets itself to await the next interrupt from the DSP.
- **Transmit / Receive Byte Registers**: these registers are used to transfer data between the two microprocessors. The PC views the Receive Byte Registers as 3 8-bit read-only registers. The three bytes receive data from

the high, middle, and low bytes of the DSP's 24-bit wide transmit register. Similarly the Transmit Byte Registers are 3 8-bit write-only registers that write to the DSP's wide receive register.

3.4.1 DSP to PC

The acquisition of image data is driven by the DSP and therefore the DSP decides when to send data to the PC; this is done on a row by row basis, *i.e.* when the DSP has acquired an entire row of data it initiates the transfer by interrupting the PC program. When it starts up, the PC program enables IRQ 7 hardware interrupts by the DSP by turning off bit 7 in the PC Interrupt Mask Register, and the PC interrupt handler is directed to to call a special procedure to handle the IRQ 7. This interrupt redirection is done by entering the address of the interrupt handling procedure in the interrupt vector reserved for IRQ 7. The DSP starts the interrupt by setting its interrupt pin high; this pin is connected to pin 32 of the PC's ISA bus and the PC's hardware interrupt controller[2] interprets the high signal on this pin as a hardware Interrupt ReQuest 7 (IRQ 7). The PIC therefore calls the designated interrupt procedure, known as *Get_Row_of_Data*, to handle the IRQ 7. If the PC is in the middle of a task with an interrupt level of higher priority than an IRQ 7, it finishes the current job before handling the data transfer. Otherwise the PC stops what it is doing (it will return to the interrupted task later) and goes to procedure *Get_Row_of_Data*.

Procedure *Get_Row_of_Data* is straightforward; it does its job as quickly as possible, since another call to this procedure is going to come very soon and this call must be finished before the next call can start; we do not want another interrupt interrupting this one! In addition, the program has other tasks that it wants to execute before reading in another row of data, such as drawing the row it is currently reading to the display. Another constraint on this procedure is that it must not call any of the PC's Basic Input and Output Services (BIOS) or call routines or functions that make BIOS calls. This is due to the non-reentrant nature of MS-DOS; once a DOS function has been called, it can not be called again until the first

[2]An Intel 8259A Programmable Interrupt Controller chip[77] on the PC motherboard.

call has ended. For example, imagine what could happen if the *Get_Row_of_Data* procedure included, as one of its last steps, instructions to draw the new row of data on the PC monitor. Suppose the PC is busy redrawing one of the user option buttons on the display and it is interrupted by an IRQ 7. The *Get_Row_of_Data* procedure is immediately called and it reads in a row of data from the DSP. Just before *Get_Row_of_Data* finishes it starts to draw the new row of data on the display, but since video writes are not reentrant, it can't — a new screen write (for the row of data) can not be initiated while a previous screen write (redrawing the user option buttons) remains unfinished. Any attempt to reenter a non-reentrant procedure is guaranteed to produce unpredictable behaviour on the part of the computer; usually the computer "locks up" and has to be restarted. To avoid this, the restriction against indirect BIOS calls in the *Get_Row_of_Data* procedure must include Turbo Pascal's input, output, and dynamic memory allocation procedures, including video writes, because they all make use of BIOS calls. This complicates the PC controller program, since the natural desire is to draw the new row of data as soon as it is acquired, *whenever* it is acquired - but the interrupt procedure is forbidden from drawing the row.

The *Get_Row_of_Data* procedure upon entry disables further interrupts before doing anything else — this prevents undesired interruption of this procedure by interrupts of lower or equal priority, including further attempts by the DSP to interrupt the PC. The PC then checks to ensure that this interrupt is a genuine one and not one generated by spurious noise triggering the PIC. An unfortunate aspect of the design of the 8259A PIC chip is that if it receives an interrupt trigger that can not be decoded and assigned an interrupt level[3], it assigns any undecodable interrupt triggers to interrupt level 7. This means that noise on the interrupt line from the DSP to the PC could be erroneously interpreted by the PIC as many IRQ 7's in sequence, and the interrupt handling procedure would be called willy-nilly when the DSP in fact had no data for the PC to read and wasn't expecting the PC to try to read an image row. This results in much confusion on the part of both microprocessors, usually leading to the halting of the controller program. As the controller was being built this indeed occurred and a check was inserted in the

[3]For instance, if the interrupt trigger signal did not last long enough for the PIC to decode.

interrupt handling procedure to prevent the problem.

As described in Chapter 4, the DSP initiated this interrupt when it handed the PC the number of the row it was about to transmit. The PC now reads this number, and waits for a few cycles for the DSP to get the image data on the data bus. This delay is implemented by looping and doing several NOP's; if the PC attempts to read the first element of the row of data before the DSP has it on the bus, the PC usually ends up reading a 0 and dropping the first element that the DSP does send. The PC reads the first point sent from the Low Byte of the Receive Register, and then loops to read 256 data points in order. The data is read from the Low and Middle bytes of the Receive Register and these two bytes are assembled to form a single 16-bit integer. This is done by shifting the most significant (Middle) byte left 8 times and adding the least significant (Low) byte. As the data points are read in they are stored in a row-long array.

The PC then raises a boolean flag called *Row_To_Draw* to signify that there is a new row to be drawn. This flag is a global variable, and it is checked at several places throughout this program, most notably within the main loop that checks for mouse input from the user and waits for the DSP to interrupt. When this flag is HIGH (TRUE), it signals that a new row has arrived and has not yet been drawn, and therefore the flag check will call a procedure to draw the newest row of data, described in section 3.6.2.

Finally the procedure sends a code to the PIC signalling the end of the interrupt, allowing the DSP or any other hardware device to initiate a new interrupt. The current interrupt is finished and flow of the program resumes wherever it left off. Since the program is peppered with checks for the *Row_To_Draw* flag signifying that a new line has arrived, it won't be long before the row is drawn.

3.4.2 PC to DSP

The PC sends the DSP the values of all of the operating parameters it will require:

- Current set point, defining the value that the feedback mechanism compares with the output of the PSPD.

- Surface tracking parameters (integral and proportional feedback constants).
- X and Y scan offsets specifying where to scan on the sample.
- X and Y scan size defining the size of the area to scan.
- Scan rate specifying the speed at which to raster the sample.

The protocol of the exchange of data from the PC to the DSP is more interesting on the DSP side and is explained in detail in section 4.4.2. The procedure used for sending data to the DSP is simple, since the PC may initiate an interrupt on the DSP by writing a 5-bit code called a Host Vector (HV) to the DSP's Command Vector Register (CVR). The PC must also set bit 7 of the CVR high to trigger the DSP to accept the interrupt request. These interrupts may be used to transfer data from one processor to another or simply to tell the DSP to execute any subroutine in its program. The DSP interprets the code written in the CVR as an interrupt vector indicating the address of the next instruction to execute. The DSP calculates the address it will jump to as

$$\text{Interrupt Address} = 2 * (\text{Host Vector}) \tag{3.1}$$

The DSP executes two instructions starting at the requested interrupt vector. The DSP programmer may use these two instructions to execute the entire interrupt, or one of the two instructions may be a jump to a longer subroutine that will handle the interrupt. The PC is oblivious to this and blithely assumes that the DSP will behave as expected. These command vectors are limited to a total of 32 by the DSP architecture, and the vectors used by the PC program and their functions are listed in Table 3.2. After sending the HV to the DSP the PC assumes that the host vector command was accepted by the DSP and that the DSP is acting as expected; for example, if the PC issues a command vector to send a new scan rate to the DSP, it would write the corresponding code in the CVR and then write the new scan rate in the Transmit Registers. It would be wise to insert a simple handshake and force the PC to wait until the DSP has begun the interrupt; this can be done by monitoring the CVR and waiting until bit 7 is set low by the DSP as it acknowledges the interrupt. The introduction of a simple handshake would decrease the possibility of the DSP missing an interrupt request from the PC

Purpose	Host Vector sent by PC
Trigger the stepper motor	$8C_{16}$
Get value for PSPD meter	$8D_{16}$
Clamp Z piezo	92_{16}
Change X scan offset	93_{16}
Change Y scan offset	94_{16}
Stop Z feedback	97_{16}
Change X scan size	98_{16}
Change Y scan size	99_{16}
Reserved to change bias voltage	$9A_{16}$
Change proportional feedback constant	$9B_{16}$
Change integral feedback constant	$9C_{16}$
Change tip scan rate	$9D_{16}$
Change feedback set point	$9E_{16}$

Table 3.2: Command vectors of DSP interrupts from the PC view.

because it is busy in a non-interruptible procedure. This method of communicating between the PC and the DSP is useful not only in moving data to the DSP but it is also used to issue other commands to the DSP, such as:

- Clamp or unclamp the Z piezo voltage within acceptable ranges.
- Step the stepper motor up or down to keep the sample within the range of the Z DAC.
- Toggle the feedback and rastering on or off, stopping the tip from scanning.
- Enable HTIE interrupts on the DSP.
- Disable HTIE interrupts on the DSP.

3.5 Graphic User Interface

The user interface for the PC is written in the graphics mode of the PC, allowing for resolution of 800 horizontal pixels by 600 vertical pixels with up to 256 colours. Since the PC will be the only interface the user will have with the AFM and since during an experiment the AFM will have to be adjusted while scanning, it is important that the user have access to all of the controls of the AFM at all times, just as the user can grab any knob or flick any switch on an analogue box of electronics. In addition, experience has shown that experimenters do not want to refer to manuals or memorize available options in the program they are using. If all of the available options are displayed on the screen, experimenters will always know what options are available to them at any point in a session at the controller.

For these reasons a graphical user interface based on a mouse was written for the controller. A large portion of the screen is taken up by three rows of rectangular "buttons" that represent parameters that may be changed by the operator. These buttons replace the knobs and switches on the front panel of a tangible controller. By sliding the mouse over a button and clicking once with the right mouse key, the user activates that button and then may change the parameter represented by that button by clicking the left mouse key. The concept of first having to activate and

then change a variable was preferred over activating a button simply by moving the mouse over it to prevent unintentional selections by the user. Some functions do require text input, such as the naming of a file in which to save an image, and the standard computer keyboard is used for these functions once the function's button has been selected with the mouse. In addition, while the AFM is not scanning, several variables normally modified by mouse-only input can be changed by first selecting them with a button and then entering new values via the keyboard.

The mouse input depends on a resident program known as a "mouse driver" that is loaded into the PC's memory when it starts up and runs continually. This driver is sold by the manufacturer of the mouse and its purpose is to provide a link between the mouse and any programs that require its use. The driver interprets the information sent by the mouse to the PC, keeping track of the mouse position and whether any keys on the mouse have been depressed. Unfortunately, mouse drivers for the PC generally are not prepared to deal with any screen with resolutions greater than 640 by 400 pixels, and therefore some assembly language programming was needed to enable our mouse to work on our display. This is generally not difficult and is based on routines demonstrated in a programmer's guide[76].

The layout of the user screen is seen in Figure 3.1. Separate screen areas are dedicated to the display of:

- The top-down greyscale image. The image display area is 256 pixels wide by 256 pixels tall, although since the pixels have a 4:3 height to width ratio, the image looks not quite square but elongated slightly in the vertical dimension.
- The virtual oscilloscope display, above the image display area.
- The user option selection buttons.
- The meter displaying the instantaneous signal from the PSPD, on the lower right hand side of the screen.

A nontrivial problem is the placement of the functions on the screen; other manufacturers of turnkey STM and AFM systems have avoided the problem of cluttering and crowding the computer screen by using two monitors, one for user input and the other for data display[74,75,91]. The use of a single monitor places a constraint

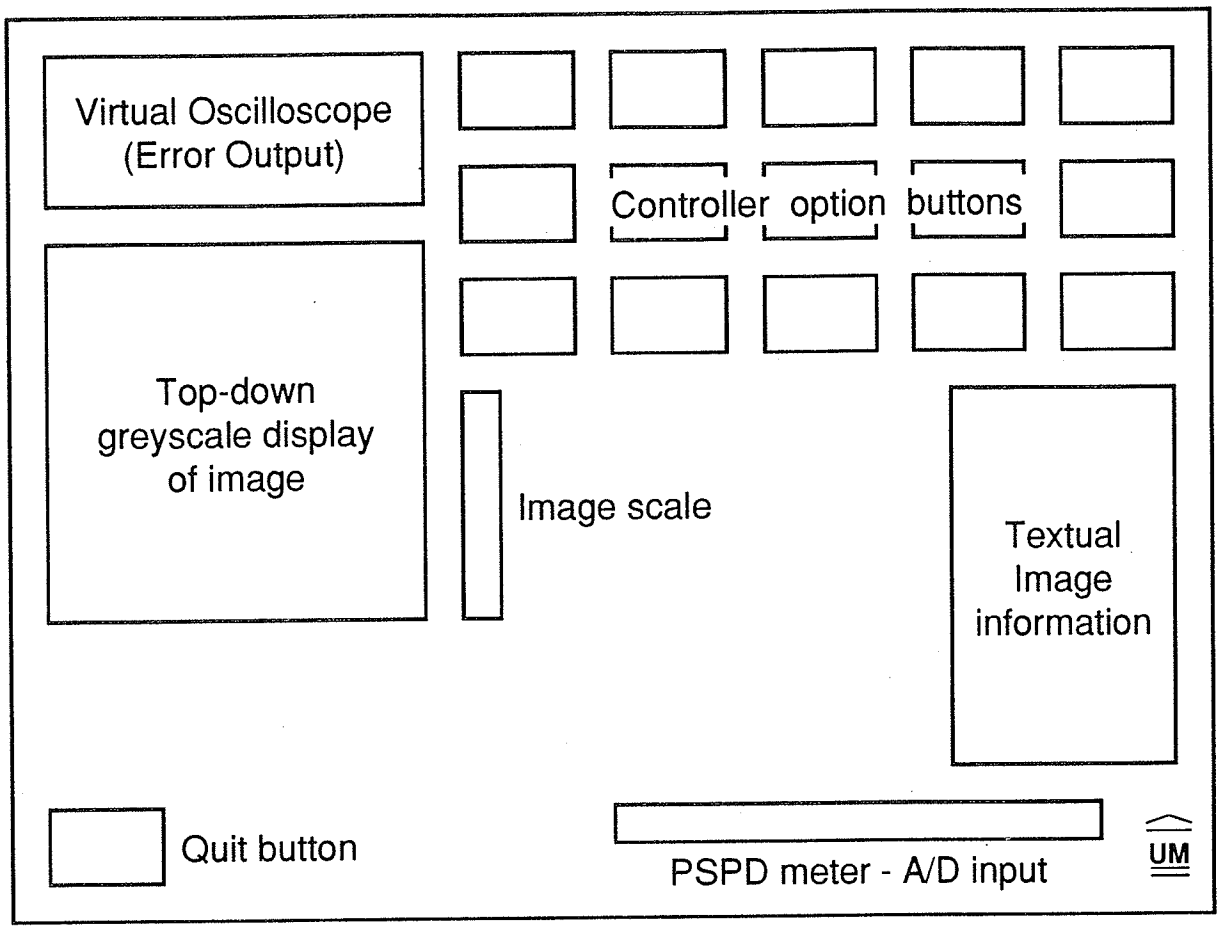

Figure 3.1: Schematic of the user's control screen.

on the design of the user interface that will become more troublesome as new functions are added. The 16 existing buttons take up as much room as is practical, yet they can not reasonably be made much smaller without becoming illegible. In order to fit any new features on a single screen, nested or pop-up menus might be used in the future, although that strategy does hide some of the available options from the user and would introduce additional delays in the speed with which a user may change parameters. Alternatively, an additional screen could be added to the system and the image could be displayed on a separate monitor, freeing up more space to be used for user control of the microscope.

3.6 Data Display

Both of the types of data displays described in section 1.3.5 are provided by the controller. The top-down grey-scale display uses the shade of the image to indicate the contour of the surface, with lighter pixels representing greater height over the darker pixels.

3.6.1 Line Display

The signal sent to compensate the Z piezo is plotted on a simulated virtual oscilloscope drawn in the top left corner of the monitor. The line drawn is a side elevation or cross section of the data and each row is plotted immediately after it is read in by the PC. Figure 3.2 shows a typical example of the display on the virtual

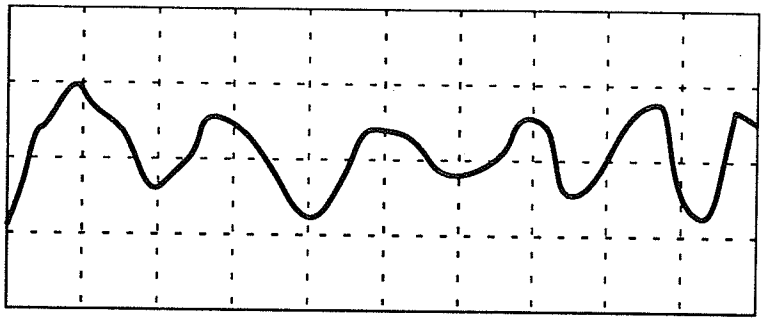

Figure 3.2: Typical virtual oscilloscope display.

oscilloscope. This format of the data is useful in operating the machine since it gives the experimenter immediate visual feedback about the status of the machine — *e.g.*, whether it is operating normally, or it is oscillating wildly, or behaving in an odd manner, and the user does not have to wait for several lines of the top-down display to be drawn before it presents a meaningful image. In the experimental arrangement previously used in the Laboratory, a real digital storage oscilloscope was usually dedicated to plotting only this signal in order to aid diagnosis and evaluation of the AFM's operation.

Since the image collected may be either relatively smooth (with little variation of the data) or rough (with a wide range of the data to be plotted) depending

on the sample, the line data must be scaled to fit the window before it is drawn. Just as the user of a real oscilloscope may select the voltage range to be drawn on an oscilloscope by turning a knob labelled "Volts per Division", the user of the AFM may select different sensitivities for this virtual oscilloscope. The options are not quantitatively labelled yet, but the options are 25 (scaled to the minimum and maximum value that may be transferred from the ADC) to 10, 5, 2, and 1. The 1 range is similar to selecting 1 Volt per Division on an oscilloscope (finer resolution), while the 25 option is analogous to selecting 25 Volts per Division (coarser resolution). These options modify the multiplicative factor used to scale the data to display it within the oscilloscope window.

The image data may in practice have a substantial DC offset that, if included in the data as it is drawn on the screen, might take up a large portion of the dynamic range available within the display window. That is, the user might be forced to choose an oscilloscope setting of lower resolution (such as the 25 setting) in order to get the display within the oscilloscope window, even though the signal of interest might be of smaller magnitude and therefore difficult to see at the low resolution setting. Therefore a type of AC coupling is introduced in the scaling procedure, leaving the line drawn centred vertically, with the data offset used determined by averaging the necessary offset over 16 rows of data. The decision to use 16 rows to average the offset is arbitrary and should be reviewed in the future; the intent is to keep the data within the range of the display and allow the use of a reasonable scale while avoiding calculating a new offset for each and every row. This is undesirable because it would remove *all* information about the DC offset of the data; some information is needed in order to monitor some aspects of the microscope's performance, *e.g.*, vertical drift of the Z piezo.

The outline and grid of the virtual oscilloscope is drawn once when the AFM enters data acquisition mode using a procedure called *InitScope* that calls routines from Turbo Pascal's graphics library. Subsequently each line representing the row of data is drawn by subroutine *UpdateScope*, making repeated calls to a standard Turbo Pascal library routine, *PutPixel*, that draws a single pixel. As each row is drawn the position of each pixel drawn is recorded in an array and as the next row is drawn in, the previous row is erased. Drawing the cross-section as a series of

single pixels has the advantage of erasing the previous line and drawing the new one in a fast and simple manner; however, since no line is being drawn between each point data that fluctuates wildly will look quite sparse on the screen. The alternative is to use graphic procedures such as Turbo Pascal's *Line* procedure to join each point drawn with lines to its neighbouring points, but this method would be slower and more complicated. The performance of this display should be evaluated and changed if necessary, although this would not be trivial if low-level graphics routines other than the ones provided by Turbo Pascal are required.

3.6.2 Greyscale Display

While the line format of displaying the data permits instantaneous user feedback regarding the temporal operation of the AFM, it does not provide meaningful spatial analysis of the entire sample image collected. The display method is preferred for this is the top-down grey-scale display as described in section 1.3.5.

The data representing the image is in the form of an ordered array of numbers representing the height of the tip at each point. A single image data point passed from the DSP to the PC is an integer between -32,768 and +32767, but in order to be displayed on the VGA monitor it must be assigned a pixel colour representing its height. This is known as *grey-level quantization* and is quite simple; the goal is to make full use of the shading range allowed by the display hardware while ensuring that all data elements are displayed by a valid colour. This is a straightforward task, since we need only know the value of each data element, the maximum value for an entire image, and the minimum value for each image. The maximum and minimum are used to normalize the data elements to take full advantage of the dynamic range of the display. For simplicity we have kept the range of the heights assigned to each display colour all the same magnitude, although it would not be difficult to introduce a nonuniform colour assignment scheme in an attempt to improve the contrast of an image[79].

The image is displayed in shades of grey, although colour could easily be programmed in through the use of false-colour look-up tables. Although the number of grey levels needed to accurately represent the data in this manner varies from

image to image, it is generally accepted that 16 levels is usually unacceptable while 64 levels suffice[79]. For this reason, and to keep things simple, we have employed 64 levels of grey in our image quantization scheme. In-house experiments varying this number have shown in a subjective manner that negligible improvement in image quality is perceived beyond 64 levels of grey.

As mentioned in section 3.4.1, a software flag is raised on the PC when a row of data is read in from from the DSP. This flag, a boolean variable called *Row_To_Draw*, is checked often by the host PC program and if it is high, a procedure called *GreyLine* displays the new row. Procedure *GreyLine* is passed the number of the row to be drawn and a pointer to the new row of data. The row number specifies how far it is to be drawn from the bottom of the image on the monitor. The row at the bottom is arbitrarily assigned number 1, while the row at the top is number 256. The row number is generated on the DSP and is passed to the PC to avert confusion on the part of the PC regarding which row it is currently drawing. The row number prevents situations where the row of data might be successfully transferred from the DSP but the PC might not have time to update the image on the screen, for example because the user is saving an image to the hard disk. In cases such as this the new image data would be stored in the PC's memory correctly but the image on the monitor would not show the new data. The row number is essential to ensure that the PC draws the new row of data where it belongs and not simply after the row of data most recently drawn.

Chapter 4

Digital Signal Processor Interface

4.1 Introduction

A Digital Signal Processor (DSP) is a microprocessor with an architecture optimized to handle digital data at high speed. For example, typically a DSP will execute a multiplication operation in a single clock cycle, as opposed to general purpose microprocessors such as the Intel 80386 that may take about 25 clock cycles to carry out the same operation. The data flow and program execution of a DSP are also usually pipelined so that the microprocessor does not have to idle while operands and instructions are fetched and decoded.

The DSP used in the AFM Controller is the Motorola 56001 DSP, a user-programmable fixed-point DSP that can perform 10.25 million instructions per second. Six memories, three peripheral interfaces, a clock generator, and seven buses are integrated on board the chip. The 56001's data paths are 24 bits wide, providing more precision than is needed for the AFM Controller. The DSP's address generation unit, arithmetic logic unit, program controller, memory, and peripherals all operate independently of each other and in parallel; this, in combination with the DSP's 3-cycle instruction pipeline, allows the chip to execute several operations in a single instruction cycle. The chip used is an integral part of Motorola's DSP56000ADS Application Development System. This system includes

1. a printed circuit DSP Developer's Board (DDB) with a DSP56001 microprocessor, external memory for the DSP's program and data, and the required control circuitry. The Developer's Board provides external connections to all of the DSP's interface ports as well as to all 96 pins on the DSP chip package.
2. a card to interface the DDB to the PC. This is a Host Bus Interface Board (HBIB) designed to sit in the backplane expansion slot of any ISA bus, such as on our PC. A flat ribbon cable attaches this card to the DDB. This card and interface are designed to allow the developer to download a DSP program from the PC and to monitor the registers and memory of the DSP. It does not lend itself to sustained high-speed data transfers between the two microprocessors, and therefore we use it only to download the DSP's program from the PC's hard drive at the start of a session. Other interface ports are used for data transfer. If the DSP program is developed to the point where it is going to remain stable for some period of time, it would be interesting to load the program into programmable read-only memory chips and install it on the DDB. This would eliminate having to download the program from the PC to the DSP every time it is to be used, although no real gains in performance should be expected.
3. software programs for the PC to
   - compile a program for the DSP,
   - download the executable program to the DSP, and
   - monitor the execution of the DSP.

As shown in Figure 2.2, the DSP communicates not only with the PC but also with the custom PCB that is the interface to the AFM. Devices on the PCB are mapped into unused memory addresses on the DSP, and the DSP accesses them just as if they were in DSP memory. The addresses of the devices accessed are listed in Table 4.1. Any number of devices may be addressed in this manner, the only fundamental limitation being the finite address space of the DSP itself. Of course, the PCB must be modified for each additional device, with the addition of a chip socket, routing to the chip, and address decoding for the device. This

Device	DSP Address
Z piezo DAC	y:a001$_{16}$
X piezo DAC	y:a002$_{16}$
Y piezo DAC	y:a003$_{16}$
V voltage DAC	y:a004$_{16}$
PSPD ADC	y:a005$_{16}$
ADC Conversion Trigger	y:a006$_{16}$
Stepper Motor Controller	x:ffe5$_{16}$

Table 4.1: Addresses of devices on the DSP.

inter-circuit board communication is done through the DDB's extension to the DSP chip's pins, since the DDB provides a 96-pin connector that we may plug into our custom PCB. The DDB also has a 16-pin connector that is used as our high speed interface between the DSP and the PC. This connector leads to the DSP's "Port B" interface, and its pins include data bits, address bits, read/write enables, host enable, host request, and host acknowledge pins. The other end of this connector is attached to a standard ISA prototyping card[93] sitting in a slot in the PC's backplane. The schematic for the circuitry on board this card is shown in Figure 4.1: the card came with the circuits needed for address decoding and data transfer, although two chips on the middle right side of the diagram were added to solve a problem discussed in section 4.4.1.

An additional problem that surfaced with the addition of this board was noise caused by a current loop through the ground from the PC through the connector used to monitor the DSP and the connector used to transfer data to and from the DSP. This ground loop was broken by removing the ground connection between the prototyping board and the DDB. Once the DSP program has been downloaded and started through the HBIB, the Port B interface connected to this prototyping board is the only port used for communication between the the DSP and the PC.

The program running on the DSP is written in a combination of C and assembly language and is compiled by the cross-compiler provided by Motorola as

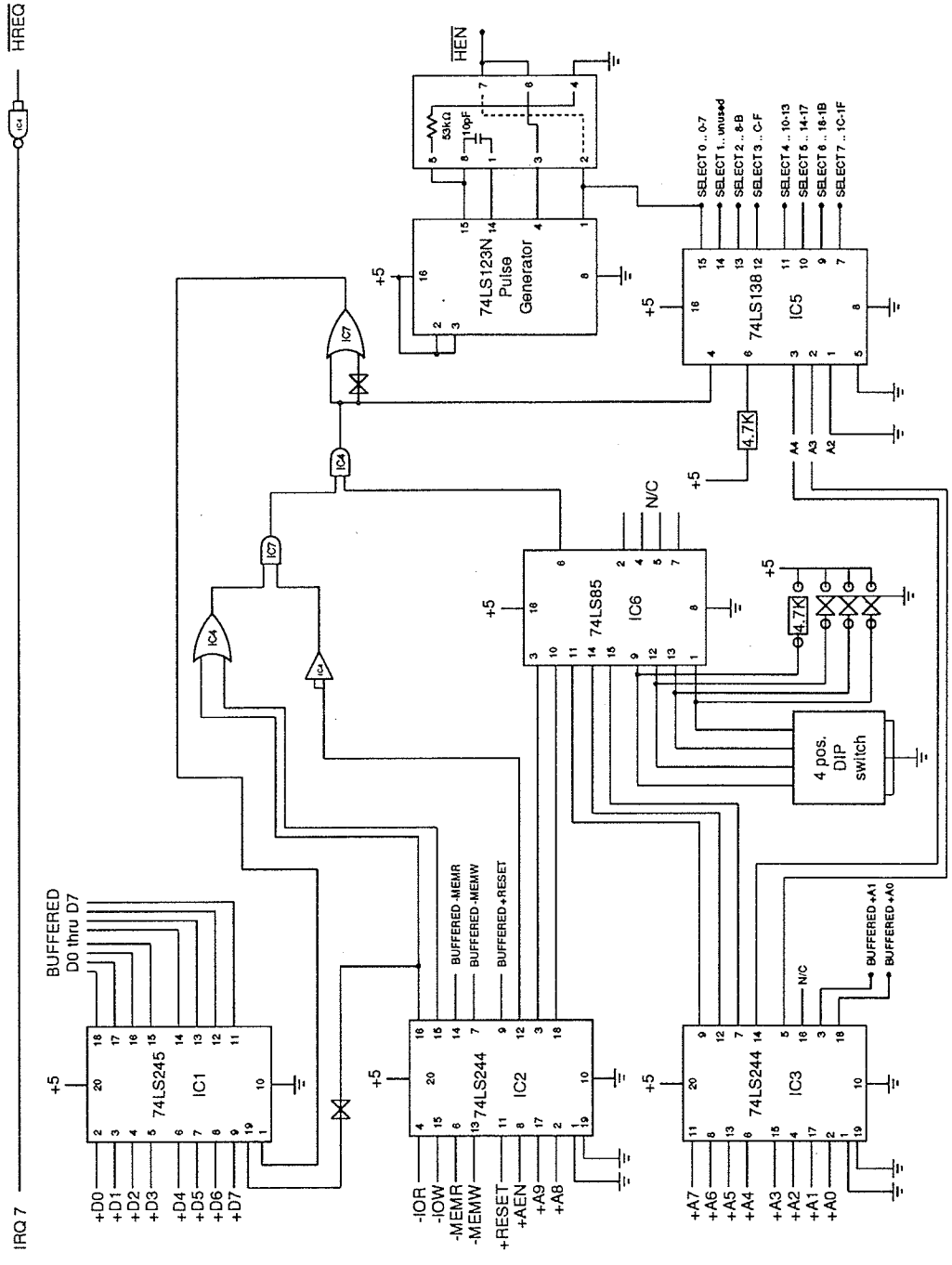

Figure 4.1: Schematic of ISA prototyping card used to interface the DSP Development Board to the PC. Adapted from [93].

part of the development package[4]. The use of C shortened the development time considerably, but assembly language still had to be used in all subroutines in order to specify the address of each subroutine in DSP memory. The location of the subroutines had to be known in advance of compilation in order to call them from interrupt vectors, and the assembly language was needed to designate the subroutine addresses.

The DSP's role in the AFM Controller is to

1. Raster the sample underneath the tip at a rate selected by the user. To do this the DSP must periodically update the values of DAC's for the X and Y piezos.
2. Maintain constant force between the cantilever probe and the sample. To do this the DSP must read the signal from the Z ADC, calculate the adjustment that must be made to the Z piezo according to the feedback formula, and write the compensated value for the Z piezo to the Z DAC.
3. Collect a row of evenly-spaced data points as the tip is rastering and send the row of data to the PC.
4. Accept any changes that the PC wants to make to the AFM's operating parameters, *e.g.*, scan speed.

The mechanics of these four tasks are explained in this chapter, while the details are reserved for Appendix B. The main body of the program on the DSP consists of an infinite loop that does nothing, while the loop is interrupted to carry out the functions of the program. Internal timer interrupts are employed to execute the feedback function at a constant rate and to raster the tip, while the PC can also generate interrupts in order to direct the DSP to either carry out a function or to transfer data from the PC to the DSP.

The DSP only needs to hold in memory one line of image data at any given time, since as soon as it collects one line it sends it to the PC. This data is stored in a one-dimensional array 256 elements long.

4.2 Sample Rastering

The DAC's used for the X and Y positioning of the sample are 14-bit converters, giving them an input range of 2^{14}, or 16,384. This 14-bit range is applied to the input of the high voltage amplifiers driving the X and Y piezos. These amplifiers have an output range of 400 V, yielding a sensitivity of

$$\frac{400\,\mathrm{V}}{2^{14}\,\mathrm{bits}} = 0.0244\,\frac{\mathrm{V}}{\mathrm{bit}}. \tag{4.1}$$

Since the scanning tube has a sensitivity of about 240 Å/V in the horizontal X and Y directions we see that since

$$0.0244\,\frac{\mathrm{V}}{\mathrm{bit}} * 240\frac{\mathrm{\AA}}{\mathrm{V}} = 5.86\,\frac{\mathrm{\AA}}{\mathrm{bit}}, \tag{4.2}$$

the smallest step the tip will be able to make is about 6 Å. This is adequate for most AFM, but will not approach atomic resolution. Note that if a 12-bit DAC were substituted for the 14-bit DAC used the smallest step available would be about 24 Å, while a 16-bit DAC would provide steps approaching 1 Å. While 1 Å steps are not needed for most AFM, 24 Å steps would limit the smallest detectable feature to about 24 nm, which is a little larger than we would like.

The total available scan area is therefore divided into a grid with the origin (0,0) at the top left hand corner of the possible scan range and the bottom right hand corner of the scan range labelled (16383, 16383) as in Figure 4.2.

For instance, to move the tip into a relaxed position in the centre of the scan range and unbent by applied voltages, it would be told to go to (8192, 8192), or exactly half it's range in both X and Y directions. The experimenter selects both the size of the area to be scanned and the position of the scan within the total possible scan area. The scan sizes available to the user are all integer multipliers of the number of data points in a row, in this case 256. This multiplier specifies how many times larger this scan is with respect to the smallest possible scan. The smallest possible scan would be with X and Y scan sizes of "1" and the tip would travel along the row one bit at a time, with each new point forming part of the image data. A scan size of "2" is twice as large, while a scan size of "63" is

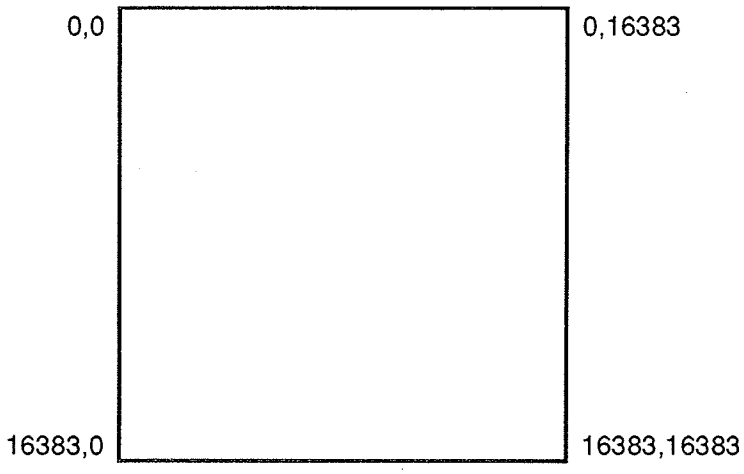

Figure 4.2: The raster co-ordinates.

the largest permissible size due to the range of the X and Y DAC's. Using only integer multiples of the unit scan size greatly simplifies the mathematics involved in calculating the size of the steps and ensures that all steps are of uniform size.

The PC is responsible for considering the requested scan speed and the requested scan size and deciding how large the DSP should make each raster step, as described in Appendix B. The code to raster the sample is called periodically by an internal timer interrupt, as described in Appendix B. Before the tip starts its first scan it is moved to the upper left corner of the scan area, and then the sample is rastered in the pattern shown in Figure 4.3; the X position is incremented along the row and 256 image data points are collected. The DSP actually compensates the tip height by reading the voltage from the PSPD and adjusting the Z piezo height four times for each one time the tip is moved; that is, the tip is moved once and then the feedback is updated four times before the tip is moved again. This is done because currently the feedback rate is tied to the raster speed, leading to very slow feedback response at low scan rates. By updating the feedback several times at each position the feedback mechanism is given time to work, instead of updating the tip height only once at each position, a situation where the integral feedback would not have a chance to take effect before the tip was moved again. The factor of four is arbitrary and is hard-coded into the DSP program; it may

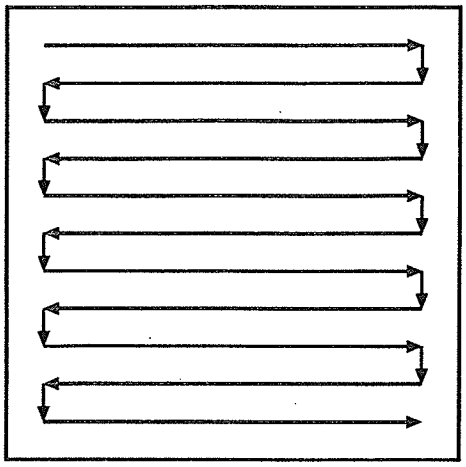

Figure 4.3: The raster pattern.

be changed if necessary. When the tip reaches the end of the scan line, the Y position is decremented once and then the X position begins decrementing, moving the tip back to the left. The tip is stepped twice in the Y direction for each row of data acquired; when the Y position has been incremented 512 times and the tip has reached the bottom right corner it simply reverses direction and retraces its route at the same speed. Image data is only collected when the tip is moving from left to right to minimize the effects of piezo hysteresis on the image. The position of the scan within the total possible scan area acts as an offset to the incrementing/decrementing procedure. The procedure to update the tip position is as simple as possible, and to illustrate we will consider the X raster signal. In a single raster line the DSP will collect 256 evenly-spaced image data points, but it will make many more than 256 steps; the exact ratio between steps and points in an image is determined by the PC as explained in Appendix B. This scheme is adopted to prevent excessively large steps in the tip motion by breaking the larger scan sizes into more and more small steps. An unduly large step applied to the tip could cause the tip to crash into the surface, and large voltage steps applied to the piezos could induce mechanical resonance and oscillation. For example, we might make a total of 768 steps across a row in the X direction, saving only each third point as part of the image as illustrated in Figure 4.4. This figure depicts the motion of the tip moving step by step from left to right, as the feedback mechanism

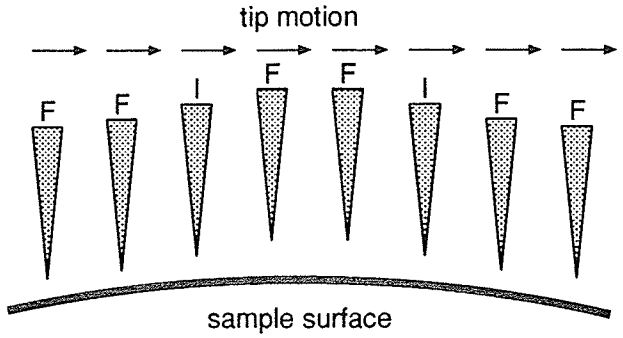

Figure 4.4: Feedback points versus image points. At points marked with an I, the tip height is compensated through feedback and the tip deflection is saved as an element of the image. At points marked with an F, the tip height is compensated but data is not saved as part of the image.

compensates the height at each step in order to maintain constant force on the tip. To differentiate between points where the tip height data is saved and where it is not saved, we name both of these points:

- An **image point** is any point stored to form an element of the image. A row of image points is collected by the DSP and saved for transmission to the PC.
- A **feedback point** is any point where the tip is moved horizontally and the tip height compensated through feedback, but the tip height is not saved to form part of the image. There are never more image points than there are feedback points.

Four variables are involved:

step_size: The magnitude of the number used to increment the X piezo position.

increment_factor: The sign of the value used to increment the X piezo. The tip changes direction when this value changes sign, keeping the code simple; when the tip reaches one end of the row, the sign of the increment_factor is negated and the tip reverses direction.

X_{offset}: The DC offset sent to the piezo to create the scan area X offset.

$X_{position}$: The tip position with respect to the X offset origin.

image_delta: The number of steps to take across one row.

X_{out}: The value that will be written to the X piezo DAC.

The new tip position is found as

$$X_{position} = X_{position} + \text{increment\_factor} * \text{step\_size} \tag{4.3}$$

$$X_{out} = X_{position} + X_{offset} \tag{4.4}$$

and the new value is then written to the X piezo DAC. The DSP then carries out the compensation at this new point and checks if it is an image point; if so, it is saved to the data array. If it is an image point, a check is also done to see if the tip is at the end of the scan line; if yes, the increment_factor is negated, the row number is incremented (or decremented if the tip is moving down), and, if the tip has finished a scan to the right, the DSP transfers the data line to the PC as described in section 4.4.1. The Y raster signal is treated in a simpler manner; it is incremented 512 times in one direction and then the direction is switched.

4.3 Digital Feedback

The feedback loop starts with the signal from the position-sensitive photodetector, serving as the input to an ADC. The DSP reads the value from the ADC and interprets it as the height of the tip above the sample, calculating a new value for the tip height in order to maintain a constant force on the sample. This new value is output to a DAC that controls the Z piezo height.

4.3.1 Conversion Devices

The signal from the PSPD forms the input to a 12-bit ADC configured to accept values between -5 V and +5V, resulting in a resolution of

$$\frac{5 - -5\,\text{V}}{2^{12}\,\text{bits}} = 2.44\,\frac{\text{mV}}{\text{bit}}. \tag{4.5}$$

The manufacturer gives the resolution of the PSPD/cantilever mechanism as 5 mV/Å[75], yielding an overall resolution of

$$\frac{2.44\,\frac{\mathrm{mV}}{\mathrm{bit}}}{5\frac{\mathrm{mV}}{\mathrm{\AA}}} = 0.488\,\frac{\mathrm{\AA}}{\mathrm{bit}}. \tag{4.6}$$

This should be more than adequate for the purposes of the AFM controller.

The height of the vertical Z piezo is governed by a 16-bit DAC on the PCB. The analogue output of the DAC forms the input to a high voltage amplifier with a range of -165 V to 65 V, yielding a precision of

$$\frac{65 - -165\,\mathrm{V}}{2^{16}\,\mathrm{bits}} = 0.00351\,\frac{\mathrm{V}}{\mathrm{bit}}. \tag{4.7}$$

Since the scanning tube has a sensitivity of about 82 Å/V in the vertical Z direction, the tip's vertical motion will be limited to steps of

$$0.00351\,\frac{\mathrm{V}}{\mathrm{bit}} * 82\frac{\mathrm{\AA}}{\mathrm{V}} = 0.29\,\frac{\mathrm{\AA}}{\mathrm{bit}}, \tag{4.8}$$

or about 0.3 Å each. This is sufficient for AFM, but is still a little too coarse for atomic-scale imaging. If a 14-bit DAC were substituted for the 16-bit DAC the smallest step available would be about 1 Å, while an 18-bit DAC would provide steps approaching 0.07 Å. Tip height resolution of 0.07 Å would be conducive to imaging at atomic resolution and should be a minimal requirement if the controller were to be modified for STM. On the other hand, 1 Å steps would result in fairly crude control and would yield indistinct images.

4.3.2 Feedback Mechanism

The rate at which the Z piezo position is adjusted is determined as described in Appendix B. The code to update the feedback is called at regular intervals by the timer interrupt, and in fact is in the same subroutine as the code to raster the sample as described in section 4.2. The DSP reads a dummy value from the address of the conversion trigger on the Z height ADC; either reading or writing anything from this address starts the conversion. The signal from the PSPD is digitized by

the ADC and the DSP reads it into variable Z_{in}. The tracking error is found as the difference between the signal from the PSPD and the desired set point chosen by the experimenter,

$$\text{error} = \text{set point - } Z_{\text{in}}. \tag{4.9}$$

The tracking error is added to the cumulative tracking error; this will be used in the digital equivalent of an integral controller, while the proportional controller uses only the instantaneous error.

$$\text{error}_{\text{cumulative}} = \text{error}_{\text{cumulative}} + \text{error} \tag{4.10}$$

Each error is multiplied by the appropriate surface tracking parameter, already entered by the user, and the new Z piezo height is calculated

$$Z = \text{Proportional}_{\text{const}} * \text{error} + \text{Integral}_{\text{const}} * \text{error}_{\text{cumulative}} \tag{4.11}$$

and written to the Z piezo DAC by the DSP. The experimenter selects proportional and integral constants ranging between 0 and 1000, although on the DSP these are normalized to range between 0 and 10. The normalization is done on the DSP when the value of the variable is read in from the PC.

The entire feedback operation currently takes about 70 μs to execute, resulting in a bandwidth of about 14 kHz. This is somewhat slower than desired, although great efforts have not yet been made to speed up the feedback operation. The ADC requires a delay of only 6 μs between the "start conversion" trigger and the reading of the digital outputs in order to guarantee stable values, so this is a fixed delay within our loop. In order to achieve a bandwidth of 50 kHz the entire loop would have to execute in 20 μs or less, and this delay would form a significant proportion of the execution time at that speed. To avoid forcing the DSP to sit and wait for 6 μs while the ADC stabilizes its outputs, the conversion trigger for the *next* feedback iteration could be placed at the end of the current loop, so that by the time the loop was entered again the ADC would have a new input value waiting for the DSP.

Section 5 describes how the assembly language code generated by the cross-compiler could be hand-optimized to gain some speed; with little effort a gain in speed of 10 % might be introduced.

4.4 Data Transfer

All data communications between the two microprocessors is initiated by interrupts, since the two CPU's are not running in a synchronous mode. The DSP interrupts the PC to transfer a completed row of data, and the PC interrupts the DSP in order to change one of the operating parameters or to direct the DSP to carry out any other operation. In all of these instances, while the DSP is interrupted the feedback loop is halted and the tip is not moved. This should not provide a problem with the AFM, as long as the interrupt does not continue for too long. If the AFM tip drifts slightly during the period that the feedback is not being updated, no damage is likely to occur to tip or sample. However, if this controller is eventually to be adapted to STM, long intervals without compensating the tip position could lead to the crash of the tip into the surface.

4.4.1 DSP to PC — Image Data

The only data transferred from the DSP to the PC is

1. The row number of the data about to be sent. The row number is an integer variable on the DSP.
2. The data representing the image. The row of data is stored as a one-dimensional array of 256 elements. One of the DSP's pointer registers, *R3*, is dedicated as a pointer to the current element of the data array as it is being transferred to the PC.

At the end of each feedback point is a check to see if the tip is at the right end of a scan line; if it is, the program branches to a section that transfers the new row of image data to the PC. The procedure used to send a row to the PC makes use of a default host command vector interrupt, one of the standard 56001 Port B interface techniques. If the Host Transmit Interrupt Enable (HTIE) bit (bit 1) of the DSP's Host Control Register (HCR) is set by the DSP, and if the PC has set bit 0 in the Interrupt Control Register (ICR), the PC just has to read from the

lowest byte of the receive byte registers of the data register in order to initiate a transmit data interrupt request in the DSP. In other words, if the conditions are right the PC reads from the DSP's transmit registers and the DSP jumps to an interrupt vector where it puts the data on the transmit registers before the PC's read cycle ends. The DSP does this by acknowledging the PC's interrupt and continuing program execution at the host transmit interrupt vector, specifically by executing two instructions at $p:0022_{16}$. We only need one instruction to move the data pointed to by the *R3* pointer to the transmit data registers and to increment the pointer after doing so (the other available instruction is filled with a NOP). This means that the pointer is now pointing at the next piece of data to be sent to the PC. The PC may read 256 times from the DSP and the DSP will send it 256 data elements, advancing the pointer after each one; after the final element the DSP should return the *R3* pointer to the start of the data array to prevent sending wild data the next time a row is transferred. This row transfer, then, is initiated by the PC when it reads the first element; the trick in this scheme is in how the PC is going to know when to start reading a row, since the rastering is done autonomously by the DSP.

The answer, of course, is obvious. Since only the DSP knows when it has acquired an entire row, the DSP must inform the PC that is ready to transmit data. The mechanism the DSP uses to trigger the PC to make that first read employs an interrupt of the PC by the DSP. The PC must have bit 0 in the DSP's Interrupt Control Register (ICR) set high, and then the PC will receive an interrupt request the first time that the DSP writes anything to the DSP's transmit registers. Bit 0 in the ICR is called the Receive Request Enable (RREQ) bit, and hence this interrupt can be called an RREQ interrupt. When the DSP writes anything to the transmit registers, bit 0 (the Receive Data Register Full or RXDF bit) in the Interrupt Status Register (ISR) goes high; if the RXDF bit and the RREQ bit are both high, the DSP asserts its external Host Request or $\overline{\text{HREQ}}$ pin. The $\overline{\text{HREQ}}$ pin is connected to the IRQ 7 pin on the PC, and when it is asserted by the DSP it triggers an interrupt on the PC that directs the PC to jump to the interrupt handling routine *Get_Row_of_Data*.

Naturally, the DSP does not write just *anything* to its transmit registers in order to trigger the interrupt on the PC; it writes the row number of the row it is about to send to the PC, as described in section 3.4.1. The PC reads the row number and lowers the RREQ bit to prevent further interrupts. The PC then reads 256 image data points from the DSP's transmit registers, initiating the HTIE interrupt each time. After the PC has read the entire row of data, it terminates the row transfer by disabling the HTIE interrupts of the PC. This entire procedure can be fairly confusing, and since it is crucial to the understanding of the heart of the AFM controller, Figure 4.5 diagrams the procedure.

When we first tried to apply this mechanism to interrupt the PC, we discovered that it didn't work; internal flags in the DSP's Host Status Register were not being updated correctly, and the interrupt didn't occur on the PC. A Motorola Design Memo[6] explained that this problem is avoided when the Host Enable signal $\overline{\text{HEN}}$ is asserted for less than 9 machine cycles; that is, 450 ns if the DSP is clocked at 20.5 MHz as ours is. The $\overline{\text{HEN}}$ signal originates on the PC and enables a data transfer on the host data bus; when $\overline{\text{HEN}}$ is asserted data may be transferred between the PC and the DSP. The $\overline{\text{HEN}}$ we were using lasted for over 700 ns, so it was used to trigger a pulse generator with a shorter output. The pulse generator was added to the prototyping board between the PC and the DSP as shown in Figure 4.1. Along with the pulse generator is a chip socket used to hold the resistor and capacitor used to set the pulse length, along with a movable jumper that may be used to disable the pulse generator and use the $\overline{\text{HEN}}$ generated by the PC. Through trial and error, a pulse length of about 500 ns was found to solve the problem. $\overline{\text{HEN}}$ lengths of much more than 500 ns prevented the PC from writing to the DSP, while lengths of much less than this prevented the PC from reading from the DSP. This pulse length should be considered a likely suspect if the PC is unable to write to or read from the DSP in the future.

An unused NAND gate on the prototyping board was used to invert the $\overline{\text{HEN}}$ pulse produced by the pulse generator. This signal was normally high and went low when active, while the PIC chip in the PC expects interrupt requests to be normally low and activated when the interrupt request signal goes high.

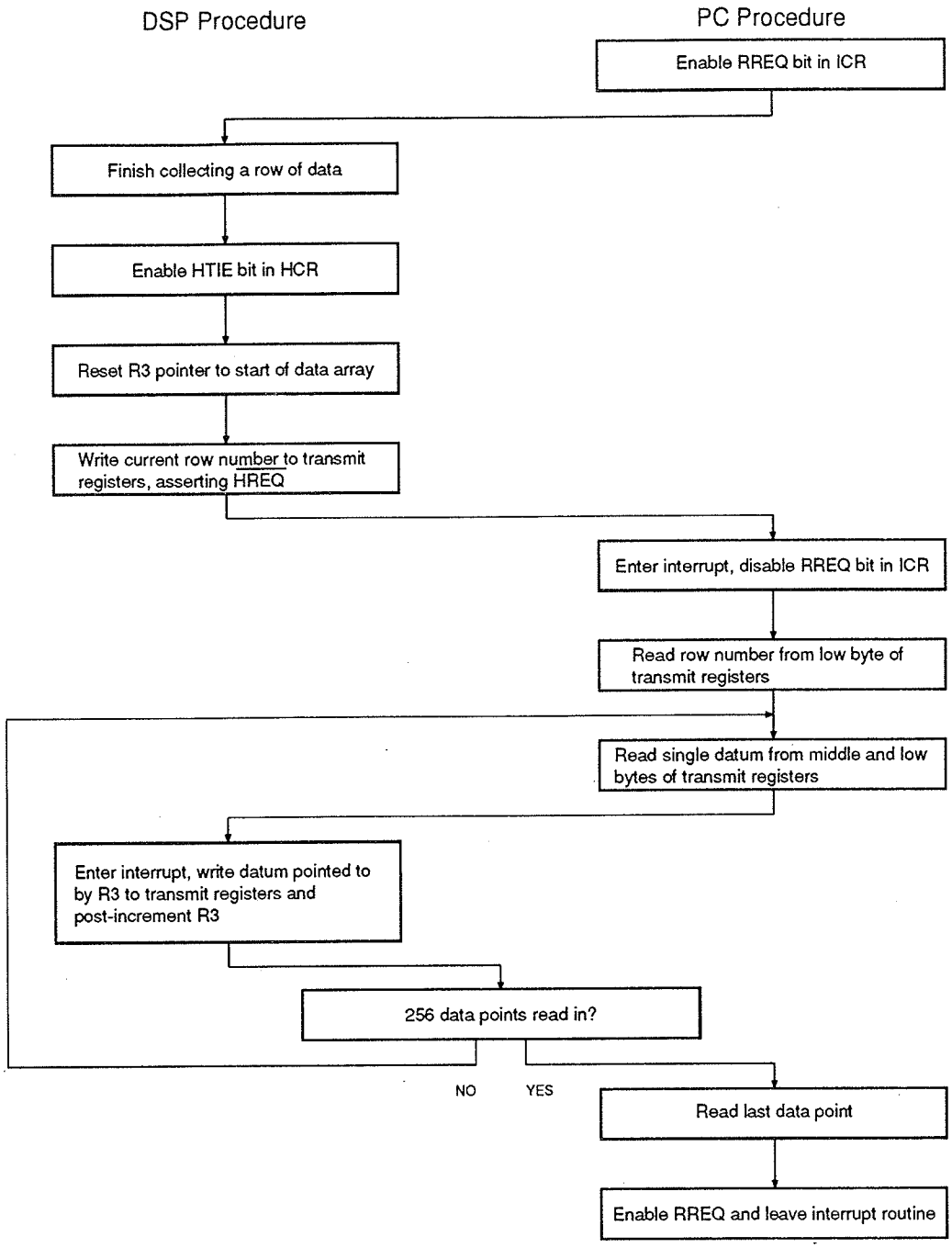

Figure 4.5: Transferring a row of data from the DSP to the PC. Execution flows from top to bottom of the diagram.

4.4.2 PC to DSP — AFM Control

The PC initiates an interrupt on the DSP for one of two reasons:

1. to download a new value for one of the DSP's operating parameters, *e.g.*, proportional feedback multiplicative constant.
2. to command the DSP to carry out a certain routine, *e.g.*, clamp the Z piezo voltage so that it does not go above a preset value.

The procedure followed by the two microprocessors is the same regardless of the PC's purpose in initiating the interrupt. The 56001's Host Command interrupt capability allows the PC to control the DSP by forcing it to do any of 13 user-defined subroutines or by causing it to execute any of the 19 default interrupt routines provided. If any of the 19 default interrupt routines are unused they may be converted into user-definable subroutines. The procedure to execute one of these subroutines is as follows:

1. The DSP must first have set the Host Command Interrupt Enable (HCIE) in the Host Control Register (HCR).
2. The PC writes to the Command Vector Register (CVR) with the desired Host Vector (HV). The HV is the memory address of the DSP's interrupt vector location divided by 2. That is, if an interrupt vector at p:0034_{16} is desired, HV = $001A_{16}$. The PC must also set bit 7, the Host Command (HC) bit in the CVR. This may be done at the same time that the HV is written, so that in the example above instead of writing $001A_{16}$ to the CVR the PC would write $009A_{16}$. A list of the interrupt vector addresses used to date and the function implemented at each one is shown in Table 4.2. Note that several valid addresses remain unused and are open for future expansion.
3. Once the HV has been set, the DSP begins processing the interrupt. The Host Command Pending Bit is set in the Host Status Register and may be read by the PC to verify that the Host Command has been started.
4. The DSP jumps to the interrupt vector address indicated by the HV and executes exactly 2 one-byte instructions. If the interrupt can be executed in

Purpose	DSP Address	Vector Written by PC
Enable timer interrupt	p:$000C_{16}$	86_{16}
Trigger stepper motor	p:0018_{16}	$8C_{16}$
Read value for PSPD meter on PC	p:$001A_{16}$	$8D_{16}$
Default timer interrupt vector	p:$001C_{16}$	$8E_{16}$
Reserved for Motorola	p:$001E_{16}$	$8F_{16}$
Default HTIE vector	p:0022_{16}	90_{16}
Toggle Z clamp	p:0024_{16}	92_{16}
Change X offset	p:0026_{16}	93_{16}
Change Y offset	p:0028_{16}	94_{16}
Enable HTIE interrupts	p:$002A_{16}$	95_{16}
Disable HTIE interrupts	p:$002C_{16}$	96_{16}
Toggle on/off	p:$002E_{16}$	97_{16}
Change X scan size	p:0030_{16}	98_{16}
Change Y scan size	p:0032_{16}	99_{16}
Change bias voltage	p:0034_{16}	$9A_{16}$
Change proportional feedback	p:0036_{16}	$9B_{16}$
Change integral feedback	p:0038_{16}	$9C_{16}$
Change scan rate	p:$003A_{16}$	$9D_{16}$
Change set point	p:$003C_{16}$	$9E_{16}$

Table 4.2: Addresses of interrupt vectors on the DSP.

two instructions this is a short interrupt and the program execution returns to where it left off almost immediately.

5. If the interrupt cannot be executed in two instructions, it may turned into a long interrupt by making one of the two instructions a jump to a subroutine.

Most of these interrupt routines used employ a long jump to another memory location where the actual work of the subroutine is carried out. Table 4.3 shows the address of each of these these subroutines in DSP memory. The actual address

Purpose	DSP Address
Start of program	$p:0040_{16}$
Enable HCIE and HTIE	$p:0110_{16}$
Read set point	$p:0114_{16}$
Disable HCIE and HTIE	$p:0120_{16}$
Enable timer interrupts	$p:0130_{16}$
Read and reset new scan rate	$p:0140_{16}$
Read proportional feedback constant	$p:0150_{16}$
Read integral feedback constant	$p:0160_{16}$
Change bias voltage	$p:0170_{16}$
Read X offset value	$p:0180_{16}$
Read Y offset value	$p:0190_{16}$
Read X step size	$p:01A0_{16}$
Read Y step size	$p:0200_{16}$
Toggle start/stop	$p:0220_{16}$
Send meter value to PC	$p:0300_{16}$
Step motor up/down	$p:0400_{16}$
Update feedback, X, Y position, data transfer	$p:0500_{16}$

Table 4.3: Addresses of program subroutines on the DSP.

of each routine is not important in and of itself, but since the jumps to these subroutines are written in assembly language, the programmer must know where

to jump before the program is compiled. Therefore the various subroutines are all compiled with the address of their first instruction specified by the programmer. This leaves the programmer with the onus of ensuring that one subroutine does not take up more memory than expected and run into the memory allocated for the next subroutine.

To illustrate the mechanism used by the PC to control the operation of the DSP, consider the changing of the integral feedback constant. The source code for both the DSP and the PC is included in Appendix ??, although the code is not necessary to follow the procedure. Assuming that the experimenter has already used the mouse or the keyboard to change the constant on the PC, the PC program flow is sent to a routine, *Convert_Int_Feedback*, in Turbo Pascal unit *ConvertUnit*. Procedure *Convert_Int_Feedback* accepts the new value for the constant as well as a flag indicating whether the AFM is now acquiring data. Upon entry *Convert_Int_Feedback* first checks this flag, and if the flag is high and the AFM is scanning, a global flag called *Row_to_Draw* is checked to see if a new row of data has arrived from the DSP and is waiting to be drawn on the monitor. If new data has arrived, procedure *GreyLine* is called to draw it on the screen, and program flow returns to *Convert_Int_Feedback*. The constant, a 2-byte integer between 0 and 1000, is broken down into single bytes and written to the middle and low bytes of the DSP's receive registers. The DSP doesn't read them yet, because it doesn't know that they've been put there. The PC then writes the correct Host Vector to the DSP's Command Vector Register and raises the Host Command bit in the CVR; for the integral feedback constant, this can be done in a single step by writing $9C_{16}$ to the CVR, as shown in Table 4.2. The PC actually has to write this twice to counteract the effects of the DSP's pipelined instruction queue, an arcane solution suggested informally by Motorola's support staff[7]. An exception occurs on the DSP and the DSP program jumps to the address indicated by the Host Vector; in this case, to p:0038_{16}. The DSP will execute exactly two instructions starting at this address. Our second instruction is a NOP and does nothing, but the first instruction is a jump to the subroutine that will read in the integral feedback constant. This subroutine is at memory location p:0160_{16}, as shown in Table 4.3. At memory address p:0160_{16} is code to read the new constant in from

the DSP's receive registers, normalize it to the range 0 to 1 by dividing it by 1000, and store the normalized value in the allocated memory location. This is done in six instructions, and program flow returns to execute the NOP at p:0039_{16} before continuing where it left off before the PC interrupted it. The AFM now has a new integral feedback constant.

4.4.3 DSP to AFM

The interface between the digital world of the DSP and the PC and the analogue world of the piezoelectric tube scanner and the PSPD is a Printed Circuit Board (PCB) connected directly to the DDB's 96-pin connector extending the 56001's input and output pins. On this PCB resides a analogue-to-digital converter (ADC) to digitize the signal from the PSPD as well as several digital-to-analogue converters (DAC's) to convert the control signals for the piezos to analogue voltages. This board, shown schematically in Figure C.1 of Appendix C, was designed and layed out by AFM co-designer Dr. G.C. McGonigal. The PCB also has all of the additional circuitry needed to complete the interface, including high voltage op-amps, latches, a stepper motor controller, bus transceivers and multiplexors, and address decoders.

Since the devices communicating between the AFM and the DSP are mapped into the DSP's memory as shown in Table 4.1, the DSP can update any AFM parameter or read in any AFM parameter as if it were reading or writing a memory location. The address decoding is all done in hardware on the PCB.

The devices are laid out on the PCB with the digital devices on one half of the board and the analogue devices on the other. Separate ground planes are provided for the digital and the analog sides to keep electrical noise to a minimum; these ground planes are joined only at the power supplies to eliminate ground current loops.

Chapter 5

Results and Recommendations

Each component of the controller, both hardware and software, was tested as it was completed. Once the controller was assembled to a point that gave it a reasonable amount of functionality, it was tested on a dummy system made up of oscilloscopes, function generators, and resistor loads. Further adjustment and testing was done until the controller could be attached to the AFM head with some confidence. The test sample used was a gold diffraction grating intended for use in calibrating AFM systems. This grating has a sinusoidal pattern with peaks 1 μm apart.

The controller software was set to scan at the maximum size, but since the high voltage power supplies designed to drive the X and Y piezos were not yet hooked up, the maximum size scanned was about 3 to 4 μm instead of about $400\,\mathrm{V} * 240\frac{\mathrm{\AA}}{\mathrm{V}} = 9.6\,\mu\mathrm{m}$. The image acquired is seen in Figure 5.1.

Note that about three and one half periods of the diffraction grating are visible in the X direction; since the peaks are known to be spaced about 1 μm apart, we see that the AFM scanned about 3.5 μm, roughly as expected. We also note that the surface of the diffraction grating appears suspiciously smooth, especially in comparison with a similar image of the same grating acquired with the same AFM head but using the NanoScope^{TM} I analogue controller, as shown in

Figure 5.2. The lack of finer detail in Figure 5.1 indicates that the tip is not tracking the sample surface as accurately as it should be. This in turn implies

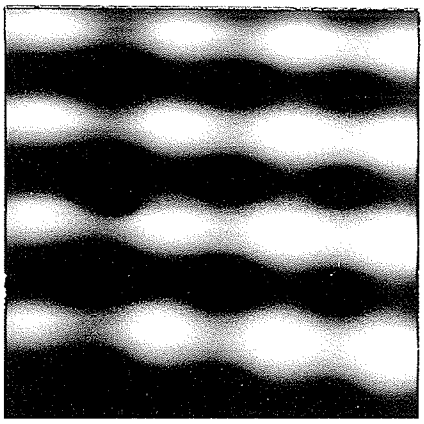

Figure 5.1: Gold diffraction grating imaged with the Digital AFM Controller.

their maximum for Figure 5.1, increasing the gain is not quite as simple as it might sound. Some redesign of the DSP's internal digital representation of the Z piezo height may be necessary in order to increase the feedback gain through software. Alternatively, another way of achieving the effect of higher gain would be to iterate the feedback mechanism more often for each horizontal step of the tip. This would require speeding up the entire feedback routine in order to maintain acceptably high bandwidth.

If the All-Digital AFM Controller is to become a valuable tool in the Scanning Probe Microscopy Lab, ongoing development should become a priority. While some projects could be undertaken by a fourth-year student as a Bachelor's Thesis, it would be advantageous to assign the care of the controller to a graduate student or other more permanent member of the research group.

There are some tasks that I did not have time to do, but that should be taken care of. Firstly, the DSP code should be hand-optimized for maximum execution speed. This is not as difficult as it might sound; the assembly language code is ex-

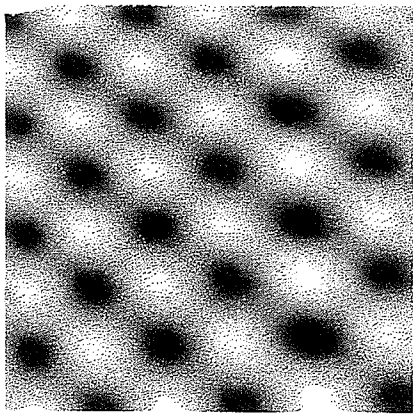

Figure 5.2: Gold diffraction grating imaged with the NanoScope^{TM} I AFM Controller.

amined and redundant data moves generated by the cross-compiler are eliminated. For instance, the cross compiler might create the program statements

```
move x1,+temp
move +temp,y0
```

which should be replaced by

```
move x1,+temp x1,y0
```

making use of the DSP's parallel move feature and saving a step in the computation.

Secondly, as explained in section 4.2, the rate at which feedback is updated is tied to the scan rate. This approach was taken because it was convenient with the DSP hardware, particularly with the SCI timer interrupt provided. Specifically, this scheme meant that at any time the DSP knew the exact position of the tip and had direct control over tip position and it would be able to hold it still if desired; this would be useful in some sorts of experiments. In retrospect it would have been wiser to trigger the updating of the feedback from a clock that is independent of the clock used to raster the tip, since doing so would allow the programmer to directly specify the feedback bandwidth and would, in most situations, provide the fastest feedback rate.

An easy solution to this problem is not apparent; the SCI timer interrupt should probably be used to trigger the updating of the feedback, since this timer is internal to the DSP and is best suited for the faster interrupt rates. The best solution might be to use an external clock, running at a rate set by the DSP, to somehow generate an interrupt on the DSP to move the tip. This would obviously require some hardware modifications and therefore might prove difficult.

Thirdly, the stepper motor has a "tip down" mode to automatically step the tip down towards the surface until the signal measured by the photodetector equals the set point value. There is no way for the user to stop this once it has started, except by changing the set point. The program should be changed to include some sort of override feature where the experimenter could halt the automated lowering of the tip.

Fourthly, a host of improvements could be made to the graphical user interface. A simple change that is essential is the addition of a routine to allow the user to make large changes in the AFM variables, such as the integral feedback constant, while the AFM is scanning. Currently while the AFM is scanning most variables may only be incremented or decremented in small steps; this could easily be adapted to, say, designate a key on the keyboard to be used along with the mouse to make larger increments. Another limiting problem with the user interface is that at higher scan speeds *e.g.,* above 25 Hz in the X directions, the PC can't keep up with the data passed by the DSP. Even though the DSP is still operating properly and presumably transferring the data correctly, the PC seems to read in only the occasional value from each row. Experiments should be carried out to find the cause of this problem and solve it. I recommend looking at the PC program first to see if shortening the PC's routines to draw the new data eliminates the problem; if so, the PC is the bottleneck and the routine should be sped up, either with a quicker algorithm or with faster hardware of some sort. A third annoying but harmless problem is the "mouse droppings" left over the screen by the mouse cursor. These show up as either

- entire sections, the shape and size of the cursor, drawn the wrong colour on the screen. This happens when an area covered by the mouse is supposed to change colour; instead of the specified colour another colour is drawn. Although the standard technique to avoid this is to hide the mouse cursor when the background is to change colour and redraw the mouse cursor when the colour change is done, this doesn't seem to work.
- two rows of oddly-coloured pixels appearing at the bottom of the mouse cursor, following the cursor as it moves around the screen.

Although these mouse droppings are unattractive they do not affect the operation of the controller. They also present a problem that could take much work to solve, for little gain. The easiest solution might be to try a newer mouse driver to see if it behaves as expected when the mouse is supposed to be hidden. Another approach that is not really a solution would be to wait for the next version of Turbo Pascal to see if they provide a framework for graphical user interfaces. Version 6.0 provided

such a framework for text-based interfaces only, and if a graphical interface system is provided it would be worth examining.

Extensions to the real-time operation of the Controller will in general be more challenging than modifications to the PC's graphics capabilities. Nevertheless, some of the modifications listed here would be of great use within the Lab and should be considered in descending order of relative importance.

1. One of the reasons for developing the All-Digital AFM Controller was to enable voltage spectroscopy types of experiments, where the tip is held stationary and tunneling current is measured as the voltage between tip and sample is varied. This example would be exceedingly easy to add to the Controller, although future variants might be somewhat less trivial.

2. An interesting experiment would be to forgo the traditional PI control loop and implement a different feedback rule[70]. A simple experiment would reveal the quantitative relation between tip height in nanometres and the signal from the PSPD in Volts. Another quick experiment would tell us the distance the Z piezo moves for each Volt applied to it. Armed with these two equations, the DSP could be programmed to compensate the tip height by reading the PSPD signal, calculating the distance the tip has to move up or down and applying the appropriate voltage to the Z piezo. This should compensate the tip almost perfectly within one or two iterations, eliminating the delay inherent in the PI controller. A low-pass filter would be required on the output of the Z piezo DAC for stability.

3. The DSP program could be rewritten to provide optimal or self-optimizing control of the AFM probe. In this scheme the DSP would continuously adjust the proportional and integral feedback constants based upon some measure of the tracking error. This is not a new idea and several texts are dedicated to the subject; Gruver and Sachs[86] provide an extensive bibliography on the topic. Although the implementation of this control would require a good deal of thought and effort, it could prove quite fruitful and should be kept in mind as a possible project for a suitable candidate.

4. A useful change to be made to the Controller is to adapt it to STM. This would require finer resolution of the control of the tip height, necessitating a more precise DAC for the Z piezo. Changing any devices on the PCB will require modifying its layout and will be a time-consuming task; for this reason it would be best to spend a few months using the AFM Controller to spot its flaws so that they may be avoided in the next iteration.

5. A constant problem in most nanoscale microscopy laboratories has been image distortion due to nonlinear behaviour of the equipment, affecting accuracy of any measurement at even relatively large scales. For instance, the piezoelectric tube scanner exhibits hysteresis and creep[56,57] resulting in distorted images. A second problem arises from interaction between the sample and the tip, *e.g.* stretching of features along one side of the image because the tip is still gaining speed just after reversing direction. Work is being done to address both of these problems[55]-[58] and when satisfactory solutions have been reached the controller should be able to incorporate them. This would give researchers more confidence in their results and would probably help to make AFM in general more useful to industry.

Of course, the DSP is not reserved solely to control the AFM, and when experiments are not being conducted the DSP is available for mathematical work. If the controller is being used to view images of data previously acquired, the PC could download the image data to the DSP for digital image processing: a few possibilities are listed here.

1. The most obvious use of the DSP is to perform a background plane subtraction of the image, removing the slope in the image and freeing up some of the dynamic range of the display system to better display the data. A more sophisticated solution would be to fit a polynomial plane to the image instead of a flat plane. Plane subtraction usually takes several seconds on the PC but one would think that the DSP could be expected to do this much quicker. The first obstacle in tackling this problem is that the data comprising an image takes more memory than the DSP has available for data. Therefore any attempts to use the DSP as a math co-processor must be able to carry

out the required operation by shifting sections of the image in and out of the DSP.

2. Often AFM images will have only a few isolated rows or points of noise; an easy way to minimize the effect of these noisy points is to average the value at the point with neighbouring points. The software currently used to present images does not allow the experimenter to average only the area immediately surrounding a single point; this would be very easy to implement on the PC. A similar routine could average a single noisy scan line with neighbouring lines.

3. Simulated 3-dimensional images with artificial light and shading are somewhat popular and are eye-catching. Several image processing texts include sections on how to create these images[94]-[96], and it does not seem to be very difficult; again, the DSP might be used for some of the math to speed up the artificial shading.

4. For certain images, such as large-scale scans of films, contour plots of the image data are desirable. Depending on how complicated the programmer wants to make it, this could be fairly easy to somewhat challenging.

Since these are generally low-priority projects, but potentially fun ones, a fourth-year student might want to make it part of a thesis. The project would give the student a chance to poke around in the system and figure out how the DSP works; if the student has to solve the memory problems discussed in the first point and work out the communication between the two microprocessors, this might become an entire thesis.

References

[1] G. Binnig, C.F. Quate and Ch. Gerber, "Atomic force microscope," *Phys. Rev. Lett.* **56**, 930 (1986).

[2] G. Binnig, Ch. Gerber, E. Stoll, T.R. Albrecht and C.F. Quate, "Atomic resolution with atomic force microscope," *Europhys. Lett.* **3**, 1281 (1987).

[3] Technical Staff, *DSP56000/DSP56001 Digital Signal Processor User's Manual,* Motorola Inc., Austin, Texas, 1990.

[4] Technical Staff, *DSP56000ADS Application Development System Reference Manual Version 2.00,* Motorola Inc., Austin, Texas, 1989.

[5] A. Aliphas and J.A. Feldman, "The versatility of digital signal processing chips," *IEEE Spectrum* June 1987, 40 (1987).

[6] Technical Staff, *XSP56001 C68S Silicon Design Memo,* Motorola Inc., Austin, Texas, 1991.

[7] Roman, Technical Staff member, Motorola Inc., phone number (512) 891-3230, personal communication of January 11, 1992.

[8] D. Rugar, H.J. Mamin, R. Erlandsen, J.E. Stern and B.D. Terris, "Force microscope using a fiber-optic displacement sensor," *Rev. Sci. Instrum.* **59**, 2337 (1988).

[9] O. Marti, B. Drake and P.K. Hansma, "Atomic force microscopy of liquid-covered surfaces: atomic resolution images," *Appl. Phys. Lett.* **51**, 484 (1987).

[10] H.J. Mamin, D. Rugar, J.E. Stern, B.D. Terris and S.E. Lambert, "Force microscopy of magnetization patterns in longitudinal recording media," *Appl. Phys. Lett.* **53**, 1563 (1988).

[11] J.E. Stern, B.D. Terris, H.J. Mamin and D. Rugar, "Position and imaging of localized charge on insulator surfaces using force microscope," *Appl. Phys. Lett.* **53**, 2717 (1988).

[12] B.N.J. Persson, "The atomic force microscope: can it be used to study biological molecules," *Chem. Phys. Lett.* **141**, 366 (1987).

[13] C.F. Quate, "Vacuum tunneling: A new technique for microscopy," *Physics Today* **39**, 26, August (1986).

[14] D. Rugar and P.K. Hansma, "Atomic force microscopy," *Physics Today* **39**, 23, October (1990).

[15] A.L. Weisenhorn, P.K. Hansma. T.R. Albrecht and C.F. Quate, "Forces in atomic force microscopy in air and water," *Appl. Phys. Lett.* **54**, 2651 (1989).

[16] P.K. Hansma, V.B. Elings, O. Marti and C.E. Bracker, "Scanning tunneling microscopy and atomic force microscopy: application to biology and technology," *Science* **242**, 209 (1988).

[17] H.K. Wickramasinghe, "Scanned-probe microscopies," *Scientific American* **54**, 98 (1989).

[18] Y. Martin, C.C. Williams and H.K. Wickramasinghe, "Atomic force microscope - force mapping and profiling on a sub 100-Å scale," *J. Appl. Phys.* **61**, 4723 (1987).

[19] T.R. Albrecht and C.F. Quate, "Atomic resolution imaging of a nonconductor by atomic force microscopy," *J. Appl. Phys.* **62**, 2599 (1987).

[20] D. Tománek, G. Overney, H. Miyazaki, S.D. Mahanti and H.J. Güntherodt, "Theory for the atomic force microscopy of deformable surfaces," *Phys. Rev. Lett.* **63**, 786 (1989).

[21] D. Sarid and V. Elings, "Review of scanning force microscopy," *J. Vac. Sci. Technol.* **B 9**, 431 (1991).

[22] G. Overney, W. Zhong and D. Tománek, "Theory of elastic tip-surface deformations in atomic force microscopy," *J. Vac. Sci. Technol.* **B 9**, 479 (1991).

[23] H.W. Hao, A.M. Baró and J.J. Sáenz, "Electrostatic and contact forces in force microscopy," *J. Vac. Sci. Technol.* **B 9**, 1323 (1991).

[24] O. Wolter, Th. Bayer and J. Greschner, "Micromachined silicon sensors for scanning force microscopy," *J. Vac. Sci. Technol.* **B 9**, 1353 (1991).

[25] H.K. Wickramasinghe, "Scanning probe microscopy: current status and future trends," *J. Vac. Sci. Technol.* **A 8**, 363 (1990).

[26] D. Sarid, D.A. Iams, J.T. Ingle, V. Weissenberger and J. Ploetz, "Performance of a scanning force microscope using a laser diode," *J. Vac. Sci. Technol.* **A 8**, 378 (1990).

[27] S.A.C. Gould, B. Drake, C.B. Prater, A.L Weisenhorn, S. Manne, H.G. Hansma, P.K. Hansma, J. Massie, M. Longmire, V. Elings, B.D. Northern, B. Mukergee, C.M. Peterson, W. Stoeckenius, T.R. Albrecht and C.F. Quate, "From atoms to integrated circuit chips, blood cells, and bacteria with the atomic force microscope," *J. Vac. Sci. Technol.* **A 8**, 363 (1990).

[28] R. Möller, A. Esslinger and B. Koslowski, "Thermal noise in vacuum scanning tunneling microscopy at zero bias voltage", *J. Vac. Sci. Technol.* **A 8**, 590 (1990).

[29] F.F. Abraham and I.P. Batra, "Theoretical interpretation of atomic force microscope images of graphite," *Surf. Sci.* **209**, L125, (1989).

[30] S.A. Gould, K. Burke and P.K. Hansma, "Simple theory for the atomic force microscope with a comparison of theoretical and experimental images of graphite," *Phys. Rev. B.* **40**, 5363 (1989).

[31] B.D. Terris, J.E. Stern, D. Rugar and H.J. Mamin, "Contact electrification using force microscopy," *Phys. Rev. Lett.* **63**, 2669 (1989).

[32] J.J. Sáenz, N. Garcia and J.C. Slonczewski, "Theory of magnetic imaging by force microscopy," *Appl. Phys. Lett.* **53**, 1449 (1988).

[33] T.R. Albrecht, S. Akamine, T.E. Carter and C.F. Quate, "Microfabrication of cantilever styli for the atomic force microscope," *J. Vac. Sci. Technol.* **A 8**, 3386 (1990).

[34] O. Marti, S. Gould and P.K. Hansma, "Control electronics for atomic force microscopy," *Rev. Sci. Instrum.* **59**, 836 (1988).

[35] R. Piner and R. Reifenberger, "Computer control of the tunnel barrier width for the scanning tunneling microscope," *Rev. Sci. Instrum.* **60**, 3123 (1989).

[36] J. Maps, "Simple raster generator for use with for scanning tunneling microscopes," *Rev. Sci. Instrum.* **62**, 357 (1991).

[37] D. Jeon and R.F. Willis, "Feedback system response in a scanning tunneling microscope," *Rev. Sci. Instrum.* **62**, 1650 (1991).

[38] B.A. Morgan and G.W. Stupian, "Digital feedback control loops for scanning tunneling microscopes," *Rev. Sci. Instrum.* **62**, 3112 (1991).

[39] R.S. Robinson, T.H Kimsey and R. Kimsey, "A digital integrator and scan generator coupled with dynamic scanning for scanning tunneling microscopy," *Rev. Sci. Instrum.* **62**, 1772 (1991).

[40] R.D. Cutkosky, "Versatile scan generator and data collector for scanning tunneling microscopes," *Rev. Sci. Instrum.* **63**, 960 (1990).

[41] Y. Kuk and P.J. Silverman, "Scanning tunneling microscope instrumentation," *Rev. Sci. Instrum.* **60**, 165 (1989).

[42] A. Brown and R.W. Cline, "A low cost, high performance imaging system for scanning tunneling microscopy," *Rev. Sci. Instrum.* **61**, 1484 (1990).

[43] A.J. Hoeven, E.J. van Loenen, P.J.G. van Hooft and K. Oostveen, "A multiprocessor data acquisition and analysis system for scanning tunneling microscopy," *Rev. Sci. Instrum.* **61**, 1668 (1990).

[44] S. Grafström, J. Kowalski, R. Neumann, O. Probst and M. Wörtge, "A compact scanning tunneling microscopy control and data acquisition system based on a Macintosh II workstation," *J. Vac. Sci. Technol.* **A 8**, 357 (1990).

[45] A. Schummers, H. Halling, K.H. Besocke and G. Cox, "Controls and software for tunneling spectroscopy," *J. Vac. Sci. Technol.* **B 9**, 615 (1991).

[46] S. Alexander, L. Hellemans, O. Marti, J. Schneir, V. Elings, P.K. Hansma, M. Longmire and J. Gurley, "An atomic-resolution atomic force microscope implemented using an optical lever," *J. Appl. Phys.* **65**, 164 (1989).

[47] R.S. Robinson, T.H. Kimsey and R. Kimsey, "Desktop computer-based management of images and digital electronics for scanning tunneling microscopy," *J. Vac. Sci. Technol.* **B 9**, 631 (1991).

[48] G.Y. Shang, J.E. Yao and J. He, "A new scanning tunneling microscope with large field of view and atomic resolution," *J. Vac. Sci. Technol.* **B 9**, 612 (1991).

[49] H. Strecker and G. Persch, "Application of scanning tunneling microscope in magnetic storage device manufacturing," *J. Vac. Sci. Technol.* **B 9**, 663 (1991).

[50] N.A. Burnham and R.J. Colton, "Measuring the nanomechanical properties and surface forces of materials using an atomic force microscope," *J. Vac. Sci. Technol.* **A 7**, 2906 (1989).

[51] P. Schroer and J. Becker, "Computer automation for scanning tunneling microscopy," *IBM J. Res. Dev.* **30**, 543 (1986).

[52] J. Becker, "Scanning tunneling microscope computer automation," *Surf. Sci.* **181**, 200 (1987).

[53] U.H. Bapst, "Automated scanning tunneling microscope," *Surf. Sci.* **181**, 157 (1987).

[54] O. Probst, S. Grafström, J. Kowalski, R. Neumann and M. Wörtge, "A tunneling atomic force microscope with inertial tip-to-sensor approach," *J. Vac. Sci. Technol.* **B 9**, 626 (1991).

[55] R.C. Barrett and C.F. Quate, "Optical scan-correction system applied to atomic force microscopy," *Rev. Sci. Instrum.* **62**, 1391 (1991).

[56] R.G. Carr, "Finite element analysis of PZT tube scanner motion for scanning tunneling microscopy," *J. Microscopy* **152**, Pt.2, 379 (1988).

[57] O. Nishikawa, M. Tomitori and A. Minakuchi, "Piezoelectric and electrostrictive ceramics for STM," *Surf. Sci.* **181**, 210 (1987).

[58] C.V. Newcomb and I. Flinn, "Improving the linearity of piezoelectric ceramic actuators," *Electron. Lett.* **18**, 442 (1982).

[59] H. Kaizuka, "Application of capacitor insertion method to scanning tunneling microscopes," *Rev. Sci. Instrum.* **60**, 3119 (1989).

[60] H. Yamada, T. Fujii and K. Nakayama, "Linewidth measurement by a new scanning tunneling microscope," *Japn. J. Appl. Phys.* **28**, 2402 (1989).

[61] J.E. Griffith, G.L. Miller, C.A. Green, D.A. Grigg and P.E. Russell, "A scanning tunneling microscope with a capacitance-based position monitor," *J. Vac. Sci. Technol.* , **B8**, 2023 (1992).

[62] J.E. Griffith, D.A. Grigg, M.J. Vasile, P.E. Russell and E.A. Fitzgerald, "Scanning probe metrology," *to appear in J. Vac. Sci. Technol.* , (1992).

[63] D.A. Grigg, P.E. Russell and J.E. Griffith, "Rocking beam force balance approach to atomic force microscopy," *to appear in Ultramicroscopy*, (1992).

[64] G. Binnig, H. Rohrer, Ch. Gerber and E. Weibel, "Tunneling through a controllable vacuum gap," *Appl. Phys. Lett.* **40**, 178 (1982).

[65] G. Binnig, H. Rohrer, Ch. Gerber and E. Weibel, "Surface studies by scanning tunneling microscopy," *Phys. Rev. Lett.* **49**, 57 (1982).

[66] G. Binnig and H. Rohrer, "Scanning tunneling microscopy," *Surf. Sci.* **126**, 236 (1983).

[67] G. Binnig and D.P.E. Smith, "Single-tube three-dimensional scanner for scanning tunneling microscopy," *Rev. Sci. Instrum.* **57**, 1688 (1986).

[68] J.A. Golovchenko, "The tunneling microscope: A new look at the atomic world," *Science* **232**, 48 (1986).

[69] D.J. Thomson and J. Song, unpublished report (1992).

[70] D.J. Thomson, private communication.

[71] R.H. Bernhardt, "Molecular microplacement using a scanning tunneling microscope," *Master's Thesis*, University of Manitoba (1990).

[72] G. Binnig and H. Rohrer, "Scanning tunneling microscopy," *IBM J. Res. Develop.* **30**, 355 (1986).

[73] R.J. Behm and W. Hösler, "Scanning tunneling microscopy," in *Chemistry and Physics of Solid Surfaces VI,* ed. by R. Vanselow and R. Howe, Chap. 14, Springer-Verlag, Berlin, 1986.

[74] Technical Staff, *Digital Instruments NanoScope^{TM} I Scanning Tunneling Microscope Instruction Manual Version 1.1,* Digital Instruments Inc., Santa Barbara, California, 1988.

[75] Technical Staff, *Park Scientific Instruments User's Manual for the SFM-BD2 Scanning Force Microscope Instruction Manual,* Park Scientific Instruments, Mountain View, California, 1990.

[76] S.K. O'Brien, *Turbo Pascal Advanced Programmer's Guide,* McGraw Hill, Berkeley, California, pp. 185 – 217, 1988.

[77] Technical Staff, *Microprocessor and Peripheral Handbook,* Intel Corporation Literature Department, Santa Clara, California, pp. 2-120 – 2-137, 1983.

[78] S.L Pfleeger, *Software Engineering,* Macmillan, New York, N.Y., 1987.

[79] R.C. Gonzalez and P.A. Wintz, *Digital Image Processing,* Addison-Wesley, Reading Mass., 6th Ed., pp. 26 – 27, 1977.

[80] D.W. Pohl, "Some design criteria in scanning tunneling microscopy," *IBM J. Res. Develop.* **30**, 417 (1986).

[81] Sang-il Park and C.F. Quate, "Theories of the feedback and vibration isolation systems for the scanning tunneling microscope," *Rev. Sci. Instrum.* **58**, 2004 (1987).

[82] D.P. DiLella, J.H. Windlass, R.J. Colton and C.R.K. Marrian, "Control systems for scanning tunneling microscopes with tube scanners," *Rev. Sci. Instrum.* **60**, 997 (1989).

[83] T. Tiedje and A. Brown, "Performance limits for the scanning tunneling microscope," *J. Appl. Phys.* **68**, 649 (1990).

[84] D.J. Thomson, "The STM as an information storage device," *J. Microscopy* **152**, 627 (1988).

[85] R.C. Dorf, *Modern Control Systems,* Addison-Wesley, Reading, Mass., 5th Ed., p. 35, 1989.

[86] W.A. Gruver and E. Sachs, *Algorithmic Methods In Optimal Control,* Pitman Pub., Boston, Mass., 1981.

[87] M.G. Say and E.O. Taylor, *Direct Current Machines,* Wiley, New York, New York, pp. 297-304, 1980.

[88] *Turbo Pascal Version 6.0, Turbo Assembler Version 2.0,* and *Turbo Debugger Version 2.0,* Borland International, Scotts Valley, California, 1990.

[89] *80386-based personal computer with ATI Integra video card and TTX colour Multisync monitor,* Powerland Computers Ltd., Winnipeg, Manitoba.

[90] *ATI Integra Enhanced Video Graphics Array card,* ATI Technologies Inc., Scarborough, Ontario.

[91] ARIS-6100 UHV STM System, Burleigh Instruments, Inc., Fishers, New York, 1992.

[92] TopoMetrix TMX 2000 Scanning Probe Microscope System, TopoMetrix, Santa Clara, California, 1992.

[93] PR-2 Prototype Card for the IBM PC, JDR Microdevices, San Jose, California, 1986.

[94] A.S. Glassner, Ed., *Graphics Gems,* Academic Press, San Diego, California, 1990.

[95] J. Arvo, Ed., *Graphics Gems II,* Academic Press, San Diego, California, 1991.

[96] J.D. Foley, A. van Dam, S.K. Feiner and J.F. Hughes, *Computer Graphics: Principles and Practice,* Addison-Wesley, Reading, Mass., 1990.

Appendix A

Glossary of Acronyms Used

ADC Analogue to Digital Converter, a chip on the Printed Circuit Board to convert analogue signals into digital representations.

AFM Atomic Force Microscope, or Atomic Force Microscopy, depending upon the context.

BIOS Basic Input and Output System on the PC, a very low-level program built into ROM and necessary for the PC's operation.

CD Clock Divider, a 12-bit counter used to time the interrupts generated by the DSP's Serial Communications Interface.

CVR Command Vector Register, a register on the DSP that may be written to by the PC. Used by the PC to cause interrupts on the DSP.

DAC Digital to Analogue Converter, a chip to convert values on the DSP into voltages on the Printed Circuit Board.

DDB DSP Developer's Board, the DSP mounted on a printed circuit board with all the necessary memory, clocks, and so on.

DSP Digital Signal Processor.

EOI End Of Interrupt signal sent to the Programmable Interrupt Controller inside the PC. The EOI indicates that an interrupt has finished.

HBIB Host Bus Interface Board, a card provided with the DSP Developer's Board to sit inside the PC and provide an interface between the DSP and the PC.

HC Host Command bit in the Command Vector Register, written by the PC to start an interrupt on the DSP.

HCR Host Control Register on the DSP, written by the DSP and, in conjunction with other variables, determines the conditions under which an interrupt of either processor may occur.

$\overline{\text{HEN}}$ Host Enable signal; when $\overline{\text{HEN}}$ is asserted data may be transferred between the PC and the DSP.

HCIE Host Command Interrupt Enable, a bit in the Host Control Register used to enable vectored interrupts of the DSP.

$\overline{\text{HREQ}}$ Host Request pin on the DSP, the pin that is connected to the PC's Interrupt Request 7 pin, and is used by the DSP to interrupt the PC.

HTIE Host Transmit Interrupt Enable, a bit in the Host Control Register used to enable the PC to interrupt the DSP simply by reading from the DSP's data transmit registers.

HV Host Vector, 5 bits in the Command Vector Register used to select the exception address when the PC initiates a command vector interrupt on the DSP.

ICR Interrupt Control Register, an 8-bit register on the DSP that the PC uses to control interrupts.

IRQ Interrupt Request, one of several interrupt levels available on the PC's Programmable Interrupt Controller.

ISR Interrupt Status Register, an 8-bit register on the DSP that the PC may use to query the status of an interrupt.

ISA Industry Standard Architecture, a standard that defines the physical and electrical characteristics of the interface in the backplane of the PC.

PC Personal Computer.

PCB Printed Circuit Board, in this case meaning specifically the board designed for the controller and not the DSP board.

PCC Port C Control Register, used to set the Port C interface as either general purpose I/O or as a serial interface.

PCDDR Port C Data Direction Register, used to indicate which bits of the Port C Interface are to be inputs and which are to be outputs.

PCD Port C Data Register, a data register we configure as outputs and use to control the stepper motor.

PI Proportional Integral, describing the type of feedback implemented on the controller.

PIC Programmable Interrupt Controller, a chip built into the PC used to control all hardware interrupts.

PSPD Position-Sensitive PhotoDetector, mounted within the AFM head and used to detect cantilever deflection.

RREQ Receive Request Enable, a bit within the Interrupt Control Register used to enable DSP interrupts of the PC via the Host Request pin.

RXDF Receive Data Register Full, a bit within the Interrupt Status Register indicating that the DSP has placed data on its transmit registers and the PC may now read them.

SCCR SCI Clock Control Register, a 16-bit register that controls the Serial Communications Interface clock.

SCI Serial Communications Interface, a port on the DSP with a built-in programmable clock designed for data transmission. We use the clock to generate periodic interrupts on the DSP and vector these interrupts to update the tip position and height.

SCP SCI Clock Prescaler selects a divide by 1 or divide by 8 prescaler for the Clock Divider.

SCR SCI Control Register, a 16-bit register used to control the Serial Communications Interface.

STM Scanning Tunneling Microscope or Scanning Tunneling Microscopy, depending upon the context.

TMIE TiMer Interrupt Enable, a bit in the Serial Communications Interface Control Register used to enable the Serial Communications Interface timer interrupt.

Appendix B

Programming Details

B.1 Scanning

The scan size, scan speed, and scan offset are closely related in this system and for that reason whenever one of these parameters is changed the new value must be checked against existing values for the other parameters before the new value is accepted and passed to the DSP.

B.1.1 Raster timing

The rate at which the scanning probe rasters over the surface is set by the experimenter. To be useful the tip speed must not only approximate the requested speed, but it must also remain constant throughout any given scan. Therefore, the timing of the tip raster signal can not be regulated simply by a delay within a loop or any other unpredictable technique. The scanning of the tip must be governed by a timer yielding the same clock rate independent of the current status of the DSP. The 56001 DSP's Serial Communications Interface (SCI) is capable of generating a timer interrupt if the TiMer Interrupt Enable (TMIE) bit of the SCI Control Register (SCR) is set high. When the TMIE is set, timer interrupt requests occur at a rate set by the SCI Clock Control Register (SCCR). A 12-bit counter is loaded

from the SCCR Clock Divider (CD) with a number representing the period between interrupts; when enabled, it starts to count down. When the counter reaches zero, an interrupt is generated and program execution jumps to the SCI timer interrupt vector located at p:$001C_{16}$; meanwhile, the counter is reloaded from the CD and the countdown starts over.

The interrupt frequency depends on the oscillation frequency of the DSP's system clock, on the CD, and on whether a SCI Clock Prescaler (SCP) bit in the SCCR is set or cleared. This bit selects a divide by 1 (SCP = 0) or divide by 8 (SCP = 1) prescaler for the counter in order to form the interrupt clock. The relation between the SCP bit, the CD, and the interrupt frequency is expressed by

$$\text{Interrupt frequency} = \frac{\text{frequency}_{\text{osc}}}{64 * (7 * \text{SCP} + 1) * (\text{CD} + 1)} \tag{B.1}$$

Since our DSP is clocked at 20.48 MHz we may rephrase this equation to find the CD needed to yield a desired interrupt frequency.

- SCP = 0

$$\text{CD} = \frac{320 * 10^3}{\text{desired interrupt frequency}} \tag{B.2}$$

- SCP = 1

$$\text{CD} = \frac{40 * 10^3}{\text{desired interrupt frequency}} \tag{B.3}$$

Note that since the counter CD may only be set with integer values between 0 and $(2^{12} - 1)$, the exact frequency desired may be lost to round-off and therefore unavailable.

The SCI timer interrupt vector located at p:$001C_{16}$ is in fact one of the default host vectors described in section 4.4.2. Two one-byte instructions are executed starting at this location. Since we cannot carry out the entire rastering operation within two bytes, the first instruction is a jump to a subroutine, and the second instruction is a NOP. Program execution continues at the subroutine to update the tip position horizontally and vertically as described in Chapter 4. The general strategy in controlling scan size and speed is as follows.

- The user sets the scan size, indirectly choosing the number of feedback points between image points. The scan size chosen is an integer multiple of 256 for simplicity. If the user chooses a scan size of 256 X 256, there are no feedback points between image points. A scan size of 512 X 512 yields 2 feedback points between image points, and so on.
- The PC then chooses a convenient step size between image data points by consulting a look-up table. In general, the larger the scan size requested, the larger the spacing between feedback points. However, this is not always the case due to the discrete nature of the calculations and the desire to keep the total number of steps in a line to a reasonable number.
- The PC sets the DSP's timer interrupt rate to the proper frequency to yield the desired X scan rate by loading the CD.

The PC calculates the desired frequency for the DSP timer interrupts using both the requested X scan frequency and the desired X scan size. Procedure *ComputeEverything* is called every time the experimenter tries to change either the scan sizes, scan offset, or scan rate. This procedure checks to ensure that the requested change will not attempt to force the AFM to scan past the range of the X or Y DAC's. If the DAC is passed an input value outside of its normal range, there is a good chance that a sign bit will be misunderstood; the result of this is that a 1-bit error in the input could result in the DAC output swinging from maximum to minimum (or vice versa) as fast as it can, forcing the piezo to move a large distance very quickly and placing dangerous stresses upon it. Acceptable ranges for the DAC's and for scan frequencies are listed in Table C.1. If procedure *ComputeEverything* decides that the requested change is permissible it calls other routines to implement the changes on the DSP (*Convert_X_Offset*, *Convert_Y_Offset*, *Convert_X_Size*, and *Convert_Y_Size*). Procedure *ComputeEverything* then passes the new variables back to the calling procedure to update the monitor, letting the user know that the changes have been accepted.

The correct interrupt rate must account for the number of raster steps per cycle, and is expressed as

$$\text{Desired interrupt frequency} = \tag{B.4}$$
$$(\text{requested scan frequency}) * (\text{number of columns per row}) *$$
$$(2\frac{\text{rows}}{\text{cycle}}) * (\frac{\text{number of raster points}}{\text{number of image points}})$$

Procedure *ComputeEverything* calculates this and checks that it is within acceptable ranges for the timer before it accepts it. Since the timer is a 12 bit counter, acceptable values fall between 0 and $2^{12} = 4096$.

B.2 Stepper Motor Control

The scanning tip is brought to the sample surface by the fine motion micrometer screw at the rear of the microscope head. Rotation in one direction advances the screw, lowering the tip, while the other direction retracts it, raising the tip. While the screw could be turned manually, more precise control is provided by the stepper motor inside the base of the AFM and an associated control circuit on a chip on the PCB. A stepper motor rotates in discrete steps in response to DC pulses applied to its stator windings. The input pulses may be supplied by a digital controller such as the Sprague UCN-5804B chip. This chip accepts as its inputs signals to

1. enable the motor,
2. select the direction in which to rotate,
3. select whether the motor will turn in whole or half steps, and
4. step (rotate) the motor once.

The chip provides the phase currents to the motor in accordance to the logic levels input to the chip. We have permanently wired the chip to provide signals to rotate the motor shaft in half steps to provide greater precision. The chip's input pins are accessed through the DSP's Port C Data Register (PCD). This port is configured as general purpose parallel I/O by setting the Port C Control Register (PCC) to all zeros, and the pins to be used as outputs are configured as such by setting the corresponding bit in the Port C Data Direction Register (PCDDR)

high; *i.e.*, to designate pins 0, 1, and 2 in the PCD as outputs we write 1's in bits 0, 1, and 2 of the PCDDR. Table B.1 shows the function assigned to each output bit used in the PCD.

DSP Address	Function	Controller Pin Number
PCD Bit 0	DIRECTION	14
PCD Bit 1	STEP	11
PCD Bit 2	ENABLE	15

Table B.1: Stepper motor controller inputs.

To step the motor once, the stepper motor controller is given signals to enable the motor, select the rotation direction, and toggle the "STEP" bit. The motor is enabled by setting the "ENABLE" pin LOW, and when the "DIRECTION" bit is HIGH the direction of rotation is such that the micrometer screw will be raised when the "STEP" bit is toggled. Therefore to lower the tip the PCD is written with alternating 1's and 3's, while to raise the tip it is written with a series of 0's and 2's. The motor is turned off by disabling the controller; that is, by writing a 4 into the PCD and setting the ENABLE pin HIGH.

Appendix C

Analog/Digital Interface Diagrams and Tables

Device	Resolution	Range
Z piezo DAC	16 bits = 2^{16} = 65536	+/- 10 V
X piezo DAC	14 bits = 2^{14} = 16384	+/- 5 V
Y piezo DAC	14 bits = 2^{14} = 16384	+/- 5 V
V voltage DAC	14 bits = 2^{14} = 16384	+/- 5 V
PSPD ADC	12 bits = 2^{12} = 4096	+/- 5 V

Table C.1: Resolutions and ranges of conversion devices.

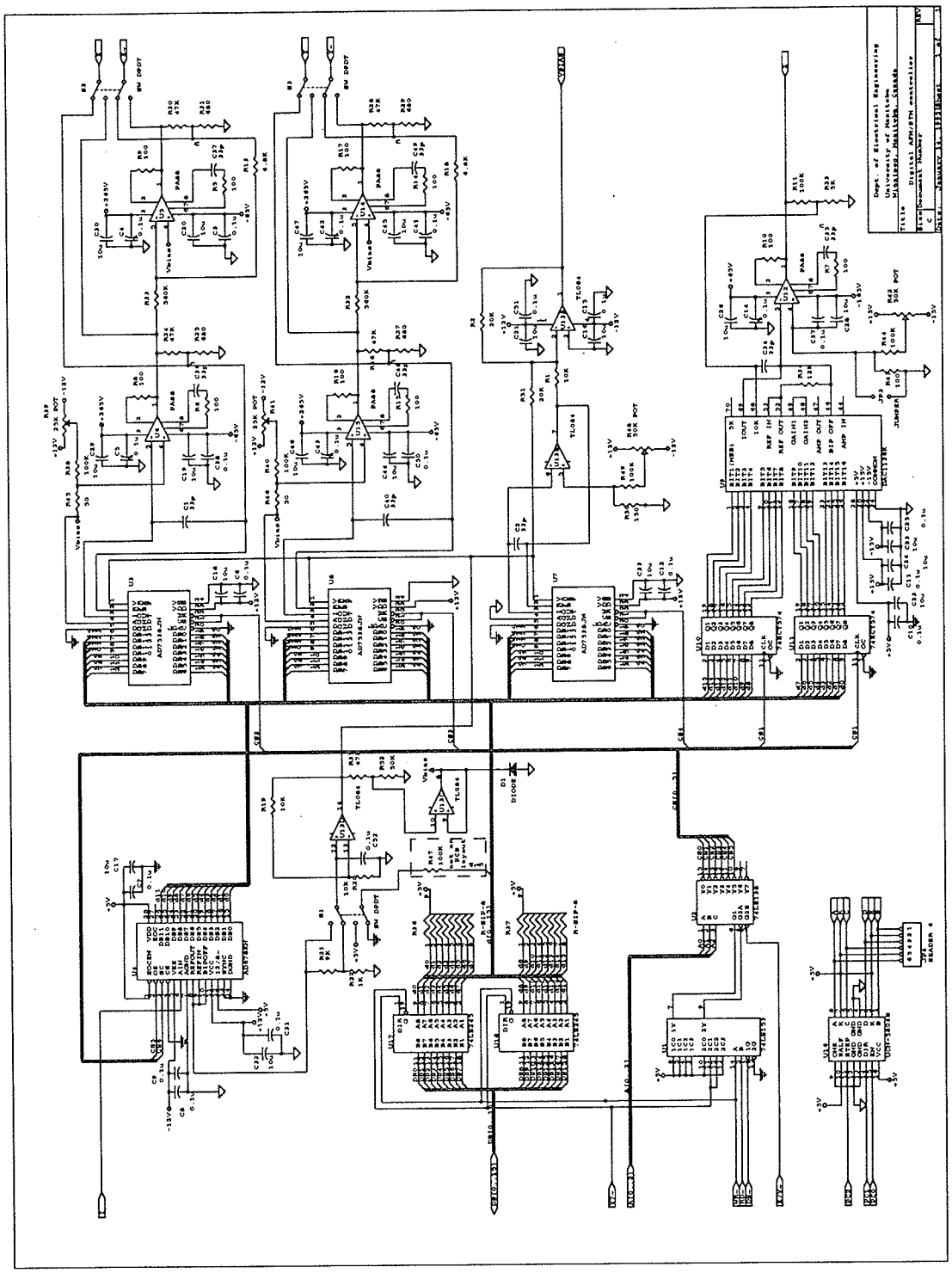

Figure C.1: Printed Circuit Board Schematic.

AFM Head Pin Number	Stepper Motor Output
19	K
20	D
21	B
22	C
23	A

Table C.2: Stepper motor connections to controller.

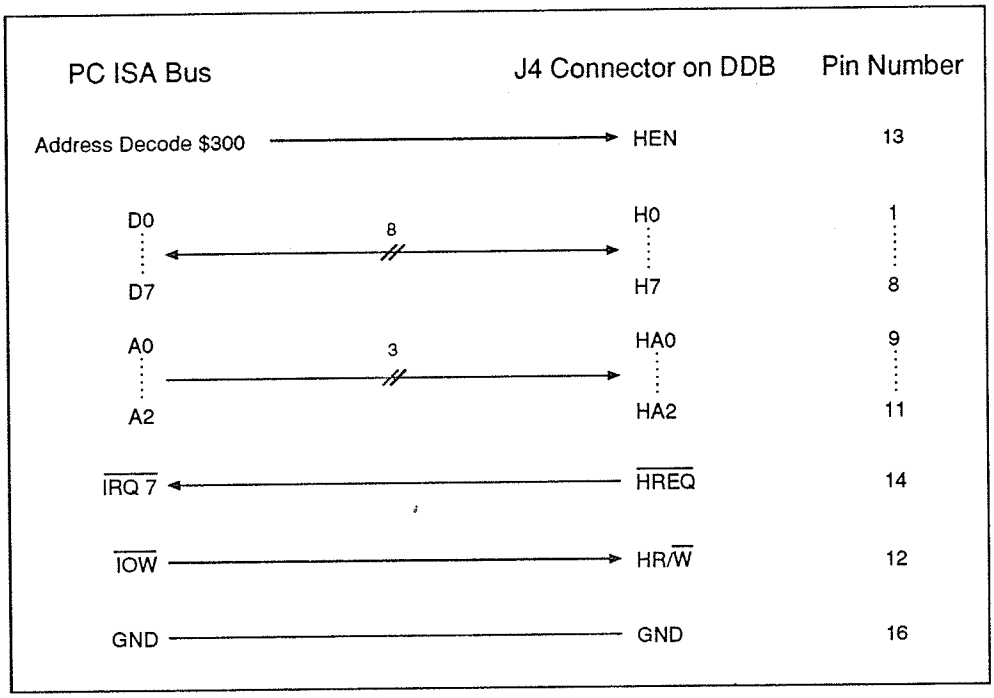

Figure C.2: High-speed interface between PC and DSP Development Board.