UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF COLUMBIA

IN RE PAPST LICENSING GMBH & CO. KG LITIGATION **This Document Relates To:** **Casio, No. 06-cv-1751**	**Misc. Action No. 07-493 (RMC)** **MDL Docket No. 1880**

CASIO COMPUTER CO., LTD.'S AMENDED REPLY TO PAPST'S COUNTERCLAIMS

Counter-Defendant, Casio Computer Co., LTD ("Casio Computer") for its Amended Reply to defendant Papst Licensing GmbH ("Papst")'s Counterclaims responds, in like numbered paragraphs, as follows:

Parties

23. Papst Licensing is a corporation existing under the laws of The Federal Republic of Germany, and has a principal place of business at Bahnhofstrasse 33, 78112 St. Georgen, Germany.

ANSWER: Admitted.

24. Casio Inc. has alleged that it is a New York corporation in the business of, among other things, selling electronic goods, with its principal place of business located at 570 Mount Pleasant Avenue, Dover, New Jersey 07801.

ANSWER: Admitted.

25. Upon information and belief, Casio Computer Co., Ltd. is a corporation existing under the laws of Japan, and has a principal place of business at 6-2, Honmachi 1-chome, Shibuya-ku, Tokyo 151-8543, Japan.

ANSWER: Admitted.

Jurisdiction And Venue

26. This Court has federal question jurisdiction of this counterclaim pursuant to 28 U.S.C. §1331 and 1338(a) because this counterclaim arises under the patent laws of the United States. 35 U.S.C. §§ 1, et seq.

ANSWER: Admitted.

27. Venue over Casio Inc. and Casio Computer Co., Ltd. is proper in this district pursuant to 28 U.S.C. §§ 1391(c) and 1400(b).

ANSWER: Admitted.

Patents At Issue

28. Papst Licensing is the lawful owner, by assignment, of the entire right, title, and interest in and to the United States Patents identified in paragraphs 29-30.

ANSWER: Casio Computer has insufficient information to either admit or deny this allegation, and so denies this allegation.

29. United States Patent No. 6,895,449 ("the '449 patent") was duly and legally issued on May 17, 2005.

ANSWER: Denied.

30. United States Patent No. 6,470,399 (the '399 patent") was duly and legally issued on October 22, 2002.

ANSWER: Denied.

Claim For Relief

31. Papst Licensing repeats and realleges each and every allegation in the foregoing paragraphs as though fully set forth herein.

ANSWER: Casio Computer incorporates by reference its response to each and every allegation of paragraphs 23 through 30 above in response to defendant Papst Licensing's counterclaims.

32. Upon information and belief, Casio Inc. and Casio Computer Co., Ltd. have or have caused to be made, used, sold, or offered for sale to customers in the United States, or imported into the United States, products that embody the elements of one or more claims of the '449 and '399 patents and, therefore, infringe those patents under the U.S. patent laws, 35 U.S.C. § 271.

ANSWER: Denied.

33. Upon information and belief, a reasonable opportunity for further investigation is likely to provide evidentiary support that Casio Inc. and Casio Computer Co., Ltd. committed the said infringements willfully.

ANSWER: Denied.

34. Upon information and belief, Casio Inc. and Casio Computer Co., Ltd. have been and still are committing the said infringements and will continue to do so unless enjoined by this Court.

ANSWER: Denied.

35. These actions by Casio Inc. and Casio Computer Co., Ltd. have damaged Papst Licensing in an amount to be determined at trial and have caused, and will continue to cause Papst Licensing irreparable injury for which Papst Licensing has no adequate remedy at law.

ANSWER: Denied.

JURY DEMAND

Papst demands a trial by jury on all issues triable by jury as of right.

ANSWER: This demand for a jury requires no answer.

CASIO COMPUTER'S ANSWER TO PAPST'S PRAYER FOR RELIEF

ANSWER: With respect to Papst's Prayer for Relief, Casio Computer denies that Papst is entitled to the relief requested in paragraphs A through G of the Prayer for Relief in its Counterclaims. Casio Computer further denies that Papst is entitled to any relief whatsoever with respect to its Counterclaims.

CASIO COMPUTER'S AFFIRMATIVE DEFENSES

In addition to the defenses below, Casio Computer expressly reserves the right to allege additional defenses as they become known through the course of discovery or otherwise.

FIRST AFFIRMATIVE DEFENSE

1. Casio Computer has not directly infringed, contributorily infringed or induced infringement of, and does not directly infringe, contributorily infringe or induce infringement of any claim of the '399 or '449 patents.

SECOND AFFIRMATIVE DEFENSE

2. One or more claims of the '399 or '449 patents are invalid for failing to comply with the conditions for patentability set forth in Title 35, United States Code § 101 *et. seq*., including, without limitation, §§ 101, 102, 103 and/or 112.

THIRD AFFIRMATIVE DEFENSE

3. By reason of the prosecution before the United States Patent and Trademark Office ("PTO") leading to, concurrently with, and/or following the issuance of the '399 and '449 patents, and by reason of statements and admissions made by or on behalf of the applicant, Papst is estopped from claiming infringement by Casio Computer of one or more of the claims of the '399 and '449 patents.

FOURTH AFFIRMATIVE DEFENSE

4. To the extent Papst and/or any licensee(s) failed to properly mark or give notice to Casio Computer under 35 U.S.C. § 287(a), Papst is precluded, or at least limited, from collecting damages from Casio Computer

FIFTH AFFIRMATIVE DEFENSE

5. Papst's claims are barred in whole or in part by the doctrines of equitable estoppel, waiver, laches and/or acquiescence.

SIXTH AFFIRMATIVE DEFENSE

6. Papst is not entitled to injunctive relief at least because any alleged injury to Papst is not immediate or irreparable, and Papst has an adequate remedy at law.

SEVENTH AFFIRMATIVE DEFENSE

7.1 Papst's allegations of infringement of the '399 and '449 patents are barred because the '399 and '449 patents are unenforceable pursuant to 37 C.F.R. § 1.56 and the doctrine of inequitable conduct. On information and belief, during the course of prosecuting the applications

leading to the ’399 and ‘449 patents, the named inventor, Mr. Michael Tasler, and/or others substantially involved in the prosecution (collectively, “the applicants”) engaged in inequitable conduct by intentionally withholding material information, including prior art, from the United States Patent and Trademark Office (“USPTO”). The applicants’ inequitable conduct is described more fully below.

7.2 In or about 1996, Michale Tasler (“Tasler”), the sole inventor of the ‘399 and ‘449 patents, authored a thesis entitled *Design and Construction of a Universal Data Acquisition and Control System for Scanning Probe Microcopy* (the “Thesis”). The Thesis was in partial fulfillment of the requirement for Tasler’s Masters of Arts degree at the University of Texas at Austin. A true and correct copy of the Thesis is attached hereto as Exhibit A.

7.3 On information and belief, the Thesis was published and made publicly available on or about 1996, before the filing of the ‘399 and ‘449 patent applications, and is prior art to the ‘399 and ‘449 patents. Tasler, as an author of the Thesis, was aware of the publication of the Thesis.

7.4 On information and belief, the Thesis was made publicly available at least through publication in or about December 1995 at the "Texas Instruments 1995 DSP Solution Challenge" contest and through cataloguing in or about August 1996 by the University of Texas in the OCLC Online Computer Library Center, before the filing of the application for the '399 and ‘449 patents, and is prior art to the '399 and ‘449 patents. Ex. A, p. 87. Tasler, as an author of the Thesis, was aware of the publication and cataloguing of the Thesis.

7.5 The Thesis is material because a reasonable examiner would consider the information contained in the Thesis important in deciding whether to allow the applications to issue into the ‘399 and ‘449 patents.

7.6 The Thesis is directed towards an interface device to interface a scanning electron microscope and a PC using an industry standard printer port. Ex. A, p. 1, 19. The Thesis is directed to solving the same two problems as the patents, i.e. PCs are not fast enough to handle the large amount of data using a direct connection to the scanning electron microscope (p. 14), and the printer port provides compatibility with many different hosts because it is standard on most PCs. Ex. A, p. 19. The Thesis describes "a Universal Data Acquisition and Control System for Scanning Probe Microscopy" that interfaces with a PC through the standard printer port. Ex. A, p. 1, 19. The Thesis further describes the system as comprising a DSP motherboard with memory (p. 18), a first connecting device for interfacing with the PC through the standard printer port (p. 19-21), a second connecting device, the TLC320AC01, for interfacing STM scanner with the DSP motherboard (p. 4-7, 23), where the DSP motherboard is configured to communicate with the PC via the printer port (p. 19-20) and includes software that communicates with the PC via the standard printer port (p. 30-31).

7.7 Tasler and, on information and belief, others substantially involved in the prosecution of the applications resulting in the '399 and '449 patents were aware or should have been aware of the Thesis and its contents during the time that the applications resulting in the '399 and '449 patents were pending in the USPTO.

7.8 During the time the applications resulting in the '399 and '449 patents were pending in the USPTO, the Thesis was never disclosed to the patent examiner.

7.9 On information and belief, Tasler and/or others substantially involved in the prosecution of the applications resulting in the '399 and '449 patents intentionally failed to disclose

the Thesis with the intent to deceive or mislead the patent examiner, and/or acted in reckless disregard of their duty to disclose material information.

CASIO COMPUTER'S PRAYER FOR RELIEF

WHEREFORE Counter-Defendant Casio Computer respectfully requests that the Court enter judgment against Papst Licensing GmbH & Co. KG, including:

a. a declaration that Casio Computer has not infringed, and is not infringing the '449 and '399 patents;

b. a declaration that each of the claims of the '449 and '399 patents are invalid;

c. a declaration that the '449 and '399 patents are unenforceable;

d. an injunction prohibiting Papst from alleging infringement of the '449 and '399 patents by Casio Computer;

e. an award of damages Casio Computer has sustained;

f. a declaration that this case is an "exceptional case" within the meaning of 35 U.S.C. § 285 due to, *inter alia*, the above actions of Papst;

g. an award of costs and attorneys fees and other expenses Casio Computer has been forced to incur; and

h. such further relief as the Court may deem just and proper.

Respectfully submitted,

DATED: June 25, 2008

/s/ J. Kevin Fee

J. Kevin Fee (D.C. Bar No. 494016)
Morgan, Lewis & Bockius, LLP
1111 Pennsylvania Avenue, NW
Washington, D.C. 20004
Tel.: (202) 739-3000
Fax: (202) 739-3001

Scott D. Stimpson (pro hac vice)
The Law Office of Scott D. Stimpson

Suite 1102
445 Hamilton Avenue
White Plains, New York 10601
Tel: (203) 258-8412

Attorneys for Plaintiff and Counter-Defendant Casio America Inc. and Counter-Defendant Casio Computer Co., Ltd.

Copyright
by
Michael Tasler
1996

Design and Construction of a Universal Data Acquisition and Control System for Scanning Probe Microscopy

by

Michael Tasler

THESIS

Presented to the Faculty of the Graduate School
of The University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF ARTS

THE UNIVERSITY OF TEXAS AT AUSTIN

May, 1996

Design and Construction of a Universal Data Acquisition and Control System for Scanning Probe Microscopy

APPROVED BY
SUPERVISING COMMITTEE:

Chih-Kang Shih

John Markert

Chapter 1: Introduction

(1.1) Motivation:

The interest in Scanning Probe Microscopy (SPM) has been increasing ever since the Scanning Tunneling Microscopy (STM) was invented by G. Binnig and H. Rohrer in 1981. Several novel technologies were developed in the following years to investigate many different surface properties. All of those techniques share the task of discovering, exploring and resolving more precise information of an expanding variety of surface structures and materials. The decreasing size of those structures give rise to new effects like the unique electronic properties of quantum heterostructures. As a consequence the design and the control of the scanning probe microscopes and data acquisition has to fulfill higher standards.

While the recent quality of the mechanical design of scanning probe microscopes already allows theoretically scanning as precise as about 1/500 Å, many control and data acquisition systems are less accurate. However, the question to be solved is where the physically meaningful limit of scanning control is to be set. Temperature related effects can only be minimized within a certain range by changing the environmental temperature of the probe and the microscope. Time dependent disturbances of the signal stability, like drift effects, influence the method of measuring as well as disturbances caused by the observation technique itself. Thus a very thorough discussion of system requirements and inevitable disturbances is essential. Only then a reasonable optimization of the control and data acquisition units is practicable.

To achieve the determination and the spatial resolution of electronic states within a GaAs/AlGaAs/GaAs quantum well formation with a STM is my future goal and therefore the overall motivation, which is used as a guideline throughout the work. At the same time it is

to be kept in mind to maintain a high flexibility of the system in order to fulfill the needs of other SPM experiments.

(1.2) General SPM Setup:

The general idea of all SPMs is to place a sharp tip close enough to the surface of interest so that atomic interactions between tip and sample can occur. The magnitude of the detected interaction signal is dependent on the effective separation of tip and sample. To keep this signal constant is the task of a feedback circuit, which can adjust the tip to sample distance for that purpose. If this feedback system is turned on and two dimensional scanning across the surface is started, the tip position will reflect a plane of constant interaction of tip and sample.

(1.2.1) General Mechanical SPM Design:

Three dimensional nanoscale tip positioning is commonly done with piezoelectric devices, which expand and contract roughly proportional to an applied voltage. One possible design is a scanning tube as shown in figure 1.1.

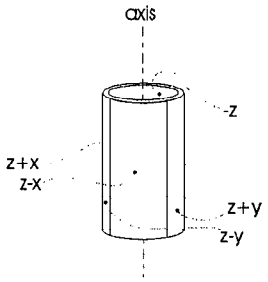


Fig 1.1: Scanning tube. The drawing shows the four segments of the tube and their wiring including the common center contact.

If different voltages are applied on opposite sections of the tube, their different expansions will lead to a bending of the tube and therefore to a movement of the axial mounted tip. This movement is roughly perpendicular to the axis. By changing the offset voltage of all segments simultaneously, the tube length will change steadily and move the tip in axial direction. Positioning within several µm range is achievable with this technique.

Although this range is plenty for atomic resolution scanning, it is too small for practical purposes. Thus a micropositioning unit is necessary, to bring the sample into the scan range of the tip. A widely used technique is a two dimensional translation stage (Walker) as shown in figure 1.2.

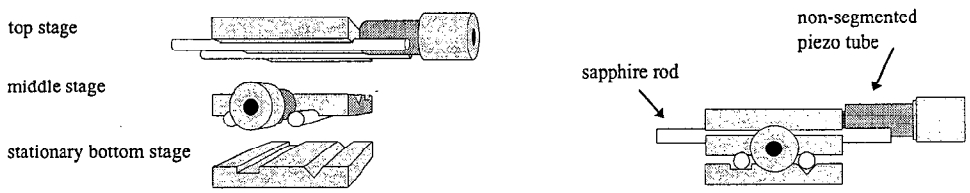


Fig 1.2: Walker design

The basic idea is to provide two perpendicular orientated motion directions. Each is represented by an independent stage of the walker. The driving force is generated by nonsegmented piezo tubes which are mounted with one end to the stage and with the other to a counterweight. If a voltage is applied so that the length of the piezo is changed, a force will act on the stage and the counterweight. Due to friction of the stage, the force has to exceed a certain threshold before motion will occur. The wave form of the applied voltage signal thus determines direction and speed of the movement. Optimal movement is achieved by cycloids as they are shown in figure 1.3. Each cycle lets the walker jump by about 50 nm. The

average step size can be adjusted by varying the frequency and the amplitude of the applied signals. Common settings are 225 V peak voltage and 1 kHz cycle frequency.

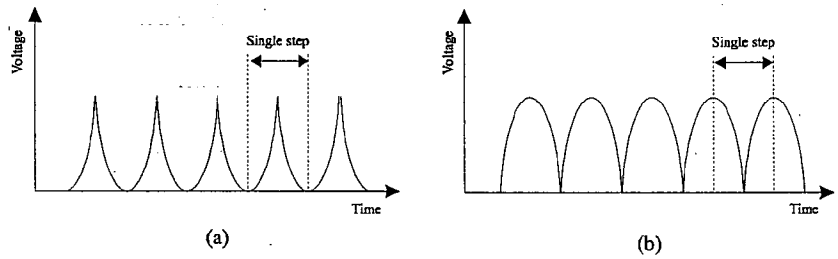


Fig. 1.3: Driving signal for walker piezo tube for (a) forward and (b) backward movement

The function for the time dependence of the voltage in figure (1.3) is:

$$(a)\quad V(t) = V_0 \cdot \sqrt{1-x^2}\ ,\ |x| \le 1$$

$$(b)\quad V(t) = V_0 \cdot \left(1-\sqrt{1-x^2}\right),\ |x| \le 1$$

where V_0 is the maximum voltage.

(1.2.2) Electrical SPM setup in the case of Tunneling Microscopy:

According to the above mentioned overall motivation of doing a certain STM study, the general description of SPMs is replaced in this chapter by a more detailed introduction into the electrical schematic of a Scanning Tunneling Microscope.

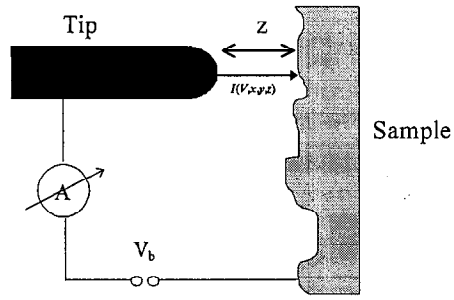


Fig. 1.4: Simplified plot of the STM constellation

If the tip to sample distance z is small enough, a tunneling current $\mathbf{I_t}$ is detected. This current is dependent of the sign and the absolute value of the applied bias voltage $\mathbf{V_b}$ and reflects directly the Local Density of States Contour. It decays exponentially with the distance **z**:

$$I \sim \exp(-\kappa\, z)$$

where

$$\kappa = \sqrt{\frac{2m\varepsilon}{\hbar^2}} \approx \frac{1}{\text{Å}}$$

and ε is the energy shown in the figure. Here, $\varepsilon_{f,t}$ and $\varepsilon_{f,s}$ are the Fermi levels of the tip and the sample. The required energy to move an electron away from either the tip or the sampl is described by ϕ_t and ϕ_s which are the work functions of tip and sample.

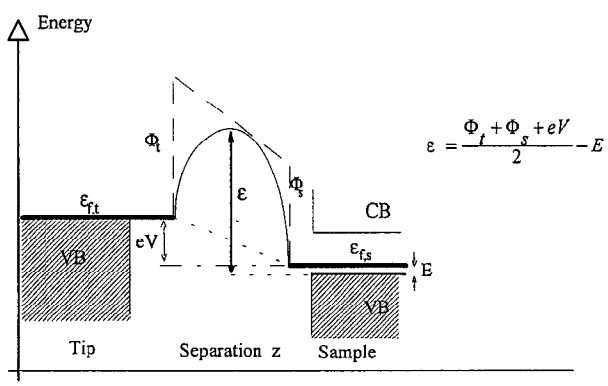


Fig. 1.4: Energy diagram of the tunneling junction. (VB) is the Valence band, (CB) is the Conduction band. E is the energy of the electronic state of the sample, which is involved in the tunneling process.

The tunneling current at a given voltage V and position z can be expressed by:

$$I(V) = \int \rho_s(E) \cdot \rho_t(E - eV) \cdot \exp[-2\kappa(E) z] dE$$

where ρ_t and ρ_s are the local density of electronic states of the tip and the sample.

As a result the direct mapping of the energy distribution of the electronic local density of states is made possible by detecting the tunneling current $\mathbf{I_t}$ at a given position (**x**, **y**) and distance **z**, while the applied bias voltage $\mathbf{V_b}$ is scanned over a given range. Due to the exponential decay of $\mathbf{I_t}$ with **z**, a precise and stable tip position is very essential. However, this z dependence can be used to expand the dynamic resolution of the acquisition system. By moving the tip closer to the sample the current signal can be magnified by several orders if necessary. This technique is of interest if the bias voltage is scanned over a wide range including zero voltage.

However, the complex interaction of **z**, $\mathbf{V_b}$ and $\mathbf{I_t}$ must be studied well before any result can be derived from the set of data obtained by this method of **z** and $\mathbf{V_b}$ variation. This was first done by Feenstra *et al.* [1], who established the simultaneously scanning of **z** and $\mathbf{V_b}$ where

$$z = \xi V_b + z_{\min}.$$

Although this method allows a better insight in the energy distribution of electronic states it still bears several uncertainties. Physically relevant results can only be derived if all data values refer to the same reference position $\mathbf{z_0}$, which turns out to be a major problem of the simultaneous **V** and **z** ramping method, since no pair of two adjacent values can be found, where either z or V is identical.

How to choose a relation, which estimates the **z** and **V** dependence of $\mathbf{I_t}$ plays a very essential role in relating all data values back to a constant reference position $\mathbf{z_0}$ and will therefore affect the derived physical results of the measurement.

It is part of this work to introduce a new method to increase the dynamic resolution of $\mathbf{I}_t$ with a different **z** variation technique.

Chapter 2: Design of the SPM Control Unit

(2.1) Theoretical Discussion of the System Requirements and Limits:

As mentioned in the introduction, the discussion of system requirements is basically done with respect to a STM study of the electronic states of an GaAs/AlGaAs/GaAs quantum well.

(2.1.1) Electronic Noise:

Recently electronic noise of the STM controlling system and its non-expandable design are limiting the capabilities of the STM. How much the intrinsic, unavoidable noises, like shot noise, Johnson noise and flicker noise, are limiting the expectable resolution is discussed in this chapter as well as thermal drift effects and filtering.

(2.1.1.1) Shot Noise:

Statistic fluctuations of any current I, which are caused by the finite quantization of the electric charges, lead to an inevitable white noise. This fluctuation has a Gaussian amplitude distribution and is known as shot noise. Since this noise is white and the relative fluctuation is larger for small currents, good filtering and a narrow bandwidth Δf is essential for the precise detection of the low tunneling current.

Assuming that 40,000 data points are taken each second, Δf is limited to 40 kHz by the relation $\Delta f \cdot \Delta t \geq 1$. For a tunneling current of 1 nA, the rms fluctuation is:

$$I_{shot\ noise}(rms) = \sqrt{2eI_t\Delta f} \approx \sqrt{2 \cdot 1.6 \cdot 10^{-19}\,\mathrm{As} \cdot 1\ \mathrm{nA} \cdot 40\ \mathrm{kHz}} = 3.6\ \mathrm{pA} = 0.36\% \cdot \mathrm{I_t}\ .$$

(2.1.1.2) Johnson noise:

In order to be able to analyze the tunneling current, a conversion to voltage and a preamplification of the signal within the UHV chamber of the STM by a factor of 10^7 is done. The basic idea is to measure the voltage, which falls off at a 10 MΩ resistor, while $\mathbf{I_t}$ is flowing through it. Additional to the shot noise, a temperature dependent white noise occurs, which is created by fluctuations of this resistance. The rms fluctuation of this so called Johnson noise is like the shot noise dependent on the bandwidth of the acquisition system. Assuming that the temperature **T** of the chamber is 80 K and that Δf is limited to 40 kHz, the effect of this noise is:

$$\frac{V_{Johnson\ noise}(rms)}{R\,I_t} = \frac{\sqrt{4k_B T\,R\,\Delta f}}{R\,I_t} = 0.42\%$$

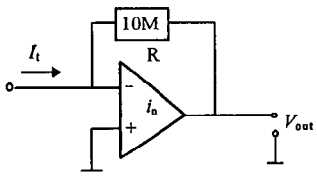
(2.1.1.3) Flicker noise:

A third kind of noise is the flicker noise, generated by the resistor as well. It has typically a 1/f dependence and causes a maximum rms noise of 0.2 μV per decade for metal-film resistors. Assuming again a bandwidth of Δf = 40 kHz, which is 4.5 decades, this noise will not exceed the 1 μV limit (which is less than 0.01% of the signal voltage).

(2.1.1.4) Noise of a JFET Preamplifier:

To obtain a low impedance output, a JFET instrumentation amplifier is used. The overall error, caused by the previously mentioned effects can be estimated for the amplifying system

according to Horowitz [3]. The internal noise $i_n/\sqrt{f}$ of JFETs below 50 kHz is about 5 fA/Hz$^{1/2}$.



The noise of this inverting unit can be expressed by the relation:

$$\frac{\Delta I}{I} = \frac{\sqrt{\left(i_n^2 + 4k_B T \frac{1}{R}\right) \cdot \Delta f}}{I} \approx 0.4\% \quad \text{for } \boldsymbol{I} = 1 \text{ nA.}$$

The accuracy of the whole system is therefore limited to 0.4%.

(2.1.2) Required accuracy:

In order to get a rough estimate for the accuracy required of the tip positioning and the control signals, we need to compute their effect on the tunneling current and to claim that those influences do not exceed the 0.4% level of the intrinsic noise.

(2.1.2.1) Accuracy of the Tip Positioning:

First of all, the tip vibration Δz must be considered. According to the approximation

$$I - \Delta I \propto e^{-\kappa(z+\Delta z)} = e^{-\kappa z} e^{-\kappa \Delta z} \approx e^{-\kappa z}(1 - \kappa \Delta z)$$

$$\Rightarrow \frac{I - \Delta I}{I} = 1 - \kappa \Delta z \;\; \Rightarrow \frac{\Delta I}{I} = \kappa \Delta z \overset{!}{\leq} 0.4\%$$

and with a realistic decay constant of $\kappa = 1.1 \cdot 10^{10}\, m^{-1}$, Δz must fulfill:

$\Delta z \leq \frac{1}{250}$ Å. That is mechanically already practicable.

(2.1.2.2) Required Accuracy of the Bias Voltage:

Another point is the ramping of the bias voltage. Referring to Feenstra's research [2], the tunneling current shows a proportionality to $(V\text{-}V_0)^{\alpha}$, where eV_0 is the band edge energy and $\alpha \approx 1.5$. From this information, it is possible to derive:

$$\frac{1}{I}\frac{dI}{dV} \approx \frac{\alpha}{V - V_0} \;\Rightarrow\; \frac{dI}{I} \approx \alpha \frac{dV}{V - V_0} \overset{!}{\leq} 0.4\%\,,$$

$$\Rightarrow \;\; (V - V_0) \geq 370 \cdot \Delta V_{step}.$$

This determines the minimal value of the energy resolution around a band edge to be three to four hundred times larger than the stepping size of the voltage ramp.

At the given temperature T = 80 K, which means that $k_B T \approx 7$ meV, it does not make sense to look for any details below 10 meV resolution. If we want to resolve details at this limit, which means that $(V\text{-}V_0)$ is minimal 10 mV, the corresponding maximal ΔV step size can be estimated by using the above derived relation to be less than 30 μV. In the case that we are scanning over a ±2.5 V bias voltage range with ten ascending intervals of 0.5 V length, every window requires a resolution of

$\frac{30\mu V}{0.5V} = \frac{1}{1.7 \cdot 10^4}$, which equals exactly 14 bit accuracy of AD and DA ICs. To obtain this resolution over the entire ±2.5 V range, a 17 or 18 bit AD/DA system would be required. That is far beyond a trivial task and exceeds my abilities. Therefore I concentrated on smaller windows of 0.5 V range each.

(2.2) Practical Realizations:

Besides the unavoidable noises, other influences like 60 Hz and 120 Hz line noise and thermal drift effects are reducing the stability and the resolution of the system.

It seems therefore to be very essential to keep the length of all signal wires short. One idea is to place the AD and DA boards within the UHV chamber of the SPM itself. First of all, the data acquisition would be closer to the origin of the signals. Secondly the metal surface of the chamber would act like a shielding system and the lower temperature in the chamber would reduce the above mentioned intrinsic noise. The only problems are that the electronic devices are not running at temperatures as low as 80K and that the number of connections into the UHV chamber is limited. The circuit boards must therefore be located in a different part of the chamber than the sample itself, which might cause some difficulties. The number of wires is kept low by providing fast serial ports which are optimized for different purposes. To explain those purposes it is helpful to think about various modes of the data acquisition, especially concerning the timing of the in- and outgoing signals.

(2.2.1) Timing Requirements for different Scanning Modes:

There are basically two different ways of using an STM: the constant current topography (CCT) mode and the constant current tunneling spectroscopy (CITS) mode.
In the CCT mode the tip scans continuously in **x** and **y** direction over the sample. The feedback circuit has to adjust **z** to provide a constant tunneling current. The measured information is **z**.
For this purpose a smooth **x** ramp and a stable, uniformly stepping **y** signal are needed. Good low pass filtering and good timing can increase the quality of the DA output signals. The feedback circuit should respond quickly and almost unaffected by any kind of filtering. In order to realize a high flexibility of the system, a digital feedback should be provided. Thus the adjustment of any feedback parameters can be easily done by software.
Those tasks make two basic sets of data converters necessary:
a medium fast set of ADCs and DACs with programmable timers, filters and gain
and a fast set of DACs with a high output slew rate.

In addition to this scanning technique an energy mapping can be done at each x,y position to obtain information about electronic states of the sample surface. Since the energy relates to the bias voltage, a measurement of $I_t(V)$ provides a method of mapping the energy distribution of the density of electronic states. This mode is called CITS. It requires stable and uniformly stepping **x** and **y** signals to scan in single steps over the sample. After each step the feedback is turned off and both **V** and **z** are scanned simultaneously and smoothly over the given range while $I_{x,y}(z,V)$ is measured. This requires slow **x** and **y** outputs, medium-fast, simultaneously timed **V** and **z** outputs, and a fast feedback response output.

Beside nontrivial programming this comes down to the same two kinds of needed sets of hardware as for the first discussed CCT mode. The slow x and y signals can be generated by medium fast DACs with a very low filtering frequency.

(2.2.2) The Digital Control Unit:

Not yet mentioned is the load of information which must be handled by the master unit of the acquisition system, which in our case is an IBM compatible PC. Although the processor of a modern PC might be fast enough to handle this amount of data, it is impossible to get it all in and out in time through any standard PC port. Thus a secondary device is to be designed, that takes over the complete SPM control, but that itself is driven by a master PC. In this case only the starting conditions and the final results must be send and received by the PC.

Especially for the fast analysis and control of multiple signal systems DSPs, Digital Signal Processors, are available. After comparing the features of many different types of DSPs, I finally chose the TMS320C50 from Texas Instruments. The advantage of this DSP is that it has a relatively large internal memory, a big set of highly optimized commands, fast and easy programmable bus systems, and the fact that a starter kit with very good manuals is offered. This starter kit provides all the features of the single DSP chip, but already contains boot software, which allows the convenient programming over the COM port of the PC.

(2.3) The Realization of an independent Processor System:

To allow a multiple timer system to work properly, a sophisticated way of handling the requests for processor time must be found. A well known way are interrupt-requests, IRQs, which interrupt the processor immediately to start to work at another part of the program. The problem is that some processes are so important that they cannot be interrupted and others are so important that they have to interrupt immediately. Therefore an interrupt manager (IM) with freely definable priorities is desired. It is very important that this IM

must not have any dead time. It must always be ready to detect an incoming request, even while its registers are read by the main processor, the DSP.

Since some wait-states for simultaneously occurring events are still inevitable, buffers have to be provided for each port for temporary storage of the data. Especially the communication between DSP and PC must be buffered well, to keep them as independent as possible.

Parallel digital I/O ports with and without line drivers should also be included in the design, to allow switching of external signals with relays or TTL signals.

PCs are in general strong radiators of noise and should be located away from the SPM. The separation of the PC and the DSP unit will also help to reduce induced noise. It is part of this project to invent a communication of DSP and PC via the standard parallel printer port of any Personal Computer system. The idea is not only to allow a convenient way of separating the devices, but also to provide compatibility to different PC systems, including Laptop computers.

Further signal improvements can be achieved by separating the ADCs and DACs from the DSP unit and by connecting them with the DSP over serial ports, as mentioned above.

Texas Instruments and Analog Devices turned out to be the best choice for 14 bit products with serial ports and programmable features. TI's device TLC320AC01 fulfills all the requirements for the medium fast purposes. It includes a programmable gains and has its own timers with adjustable offsets. The fact that it has analog in- and outputs within one single IC and that it can be linked together with ICs of the same kind allows simultaneous control of different signals, while keeping the layout dimensions small.

Analog Devices offers many high quality products for the faster purposes. A good choice is the AD7244, a fast dual DAC with serial ports.

To increase the effective dynamic range of all DAC outputs it is meaningful to keep signal offsets low and constant. This is especially important if the tip scanning plane of

constant **z** is not parallel to the surface of the SPM sample. An analog *x, y, z*-coordinate rotation can help to translate the sample orientation into the **x,y** plane of the computer scanning area. This coordinate rotation can be achieved with a combination of several ICs of Analog Devices.

For reference purposes a summary of all setup ideas is shown in the block diagram of Appendix A.

(2.4) Concluding Comments on System Limits:

The quality of the described data acquisition system is definitely worse than the estimations made in the theoretical discussion in (2.1), since the mentioned types of noise are the theoretical error limits and do not contain non-linear behaviors of amplifiers, time drift and piezo hysteresis effects.

In general time independent non-linear disturbances are predictable and measurable and can be compensated by software algorithms.

Time dependent drift effects, however, are disturbing the measurements significantly. A compromise concerning the optimal scanning speed must be found. While on one hand slower scanning allows better filtering, it increases on the other hand the total scanning time, so that drift effects are getting more important. One possible solution is the reduction of the image size, so that less data points must be taken. Therefore each measurement can be done with the needed accuracy without exceeding a total time limit. This is particularly important if a lock-in amplifier is used to measure $\mathbf{d}I/\mathbf{d}V$ curves. The adjustable bandwidth of that amplifier influences the quality of the result significantly.

Chapter 3: Hardware Design

Before starting the design of the circuits a very strict overall guideline has to be defined. To obtain optimal system performance, it is essential to determine a clear idea of the whole setup in advance.

It is the goal to create a flexible system that can be adjusted to a large variety of experimental requirements. This makes it especially important to design an easily expandable system, so that components can be added as needed.

This purpose is achievable by creating an optimized bus architecture through which the processor unit and the system components can communicate in a well defined pattern. All main bus control circuits should be combined on one board, the DSP motherboard. This board will put all relevant signals to the pins of a single connector which will be called the Bus Connector from now on. All additional boards only have to provide an identical set of bus connectors for the incoming bus and the outgoing bus, to which the next board will connect.

The DSP Starter Kit TMS320C5x (DSK), which is used as the processor unit, is well described in TI's "TMS320C5x User's Guide" [4] and is not further discussed in this chapter. Only the facts are important, that it is programmed over the COM port of a PC and that all important signals are available on four 24 pin connectors. The on board Analog Interface occupies the built in serial port of the DSP and will be used for the output of constant reference signals.

(3.1) The DSP Motherboard:

The basic idea of any bus system is the sharing of a fixed set of signal lines to distribute data to all bus parties. So called Enable (EN) Signals are completely under the control of the DSP and determine which party is active and allowed to read or write data from or to the bus.

The TMS320C50 is a 16 Bit fixed-point DSP with 10KB on-chip memory. It offers a set of 124 instructions with 50 ns cycle time. 16 of the available 65536 I/O port addresses are accessible with fast memory mapped registers. Only those I/O ports will be used in the following design. A simple 16 channel demultiplexer (74 HC 154) transfers the binary available I/O-port-address into 16 separated Enable lines (EN00 to EN15).

Interrupt Request (IRQ) Lines allow external parties to request the right to use the bus in order to communicate with the DSP. It is up to the programmer of the DSP system to enable or disable certain IRQs and to set IRQ priorities. This technique guaranties that urgent requests are processed earlier than usual requests. Furthermore unused components can be turned off by disabling (masking) their IRQ.

Although the DSP offers some IRQ lines, the number and the timing criterias of those signals do not fulfill the requirements.

(3.1.1) The Interrupt Manager (IM):

The design of a new 8 channel IRQ-Manager (IM) is shown in Appendix B.
Each channel is connected to a flip-flop, which switches its output state if a low to high transition occurs on the input. An 8 Bit comparator (74 AS 855) compares the state of all 8 flip-flops with a previously stored value. If any incoming IRQ changes the state of the

according flip-flop, the comparator will recognize the change and notify the DSP. As soon as the DSP reads the 8 flip-flop states, the comparator stores this constellation as the reference value, so that the output of the comparator switches back to 'equal'. The next incoming IRQ will change the constellation again and switch the comparator back to 'unequal'. The DSP will be notified immediately, and so forth.

By setting the Preset Pin of any flip-flop to zero, its output state is forced to be high. As a result incoming IRQs on this line are unable to toggle the state of this particular flip-flop. The IRQ is masked.

The state of any of those Preset Pins can be determined by the DSP by writing a bitsignificant value to the Interrupt Manager (IM) address (050h). Reading the same address will give the current constellation of the IRQ-flip-flops. By comparing this value to the previously read value, the DSP can easily determine if and which IRQ occurred since the last reading.

The great advantage of this method is that even extremely short IRQ pulses can trigger the IM to notify the DSP. Furthermore the IM has no deadtime, since any IM reading by the DSP will set the IM comparator value simultaneously.

A priority list can easily be programmed to determine which events have to be taken care of before others.

The detailed schematics for this part are shown in Appendix B.

(3.1.2) The Connection to the Master PC:

Since almost any modern PC system offers a parallel Line Printer Port (LPT), the communication with the DSP will be established using this standard interface.
The pinout of this port and the I/O address of the corresponding function is shown in table 3.1.
Although the LPT is only designed for monodirectional output, it can be used for bidirectional I/O operations. The idea is to read data through the 5 handshaking input bits,

which are usually the indicators for "Paper Out", "Printer Error", "Printer Selected", "Request Acknowledge" and "Busy".

To determine when to transmit or receive which information over the port the 4 availablc handshaking output pins are used. The different modes are shown in table 3.2.
Missing codes can be avoided by using one of the output signals as Data Ready (DR) trigger. A pulse on this line indicates the DSP-Interface that the selected mode is valid.
DR is automatically generated, if the Init-Signal (Pin 16) is toggled. A selected mode can simply be changed by writing the value of the new mode with Bit 2 (Init) toggled to the control register of LPT. Due to the internal setup of older LPTs some modes can only be followed by certain other modes. The allowed sequences are shown in table 3.3.

The interface of the DSP is buffered to separate the DSP and the PC as much as possible The Busy line (Pin 11 of LPT) indicates if those buffers are empty, non empty or full (see table 3.2).

Detailed information can be obtained if the Status mode is selected. The bitsignificant status value is explained in table 3.4.

If the DSP communicates with the buffers, only Bits 0 to 7 are used. After reading from the buffers, the higher bits (8 to 15) contain the status of the buffers and the recent selected LPT mode. See table 3.5 for the meaning of each bit.

The detailed schematics for this part is shown in Appendix B.

Pin	Description	I/O	Address	Bit
1	-Strobe	Output	Base+2	0
2-9	Data Bits 0-7	Outputs	Base	0-7
10	-Acknowledge	Input	Base+1	6
11	Busy	Input	Base+1	7
12	-Paper Out	Input	Base+1	5
13	-Printer Selected	Input	Base+1	4
14	-Auto Linefeed	Output	Base+2	1
15	-Printer Error	Input	Base+1	3
16	Init	Output	Base+2	2
17	-SelectIn	Output	Base+2	3
18-25	GND			

Table 3.1. Printer Port Pinout. The Standart Base Address is 0378h

Mode	Description	'Busy'-Bit
0, 4	Reset '←' buffer	unused
1, 5	Read Status	unused
2, 6	Load '←' buffer	unused
3, 7	Idle	128 if '←' buffer is full
8, 12	Reset '→' buffer	unused
9, 13	Load Bits 4-7 '→' and latch next	unused
10, 14	Reset DSP-Devices	unused
11, 15	Load Bits 0-3 '→'	0 if '→' buffer is empty

Table 3.2: Allowed Port Modes. The Mode Address is Base+2. '→' stands for DSP-to-PC transmition. '←' represents PC-to-DSP communication.

Recent Mode	Allowed Following Modes
0, 1, 2, 3	4, 5, 6, 7, 12, 13, 14, 15
8, 9, 10, 11	4, 5, 6, 7, 12, 13, 14, 15
4	0, 1, 2, 3, 8, 9, 10, 11
5	1, 3, 9, 11
6	2, 3, 10, 11
7	3, 11
12	8, 9, 10, 11
13	9, 11
14	10, 11
15	11

Table 3.3: Allowed Mode Changes.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Busy	← E	← F	→ E	→ F	NC	NC	NC
see tab 3.2	0 = empty	0 = full	0 = empty	0 = full			

Table 3.4: Status Byte (Base + 1) for modes 1 and 5

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
read write	r	r	r	r	r	r	r	r	r/w	r/w	r/w	r/w	r/w	r/w	r/w	r/w
use	M3	M2	M1	M0	←E	←F	→E	→F	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0

Table 3.5: The Buffer readout of the DSP. 'M' = PC-Mode-Bit. 'D' = Data-Bit (read and write possible)

(3.2) The TLC320AC01 Serial Port:

The TLC320AC01 contains an active serial interface, which provides all needed synchronization signals. It tells the DSP side of the interface when to transmit and receive data. Thus the DSP side of the interface is called passive. Since the DSP has to concentrate on many different tasks the design of the passive serial port (PSP) will contain a two level buffer of FIFO type (first in first out). Simultaneous transmission and receiving of serial data guarantees high throughput rates. Synchronizing flip-flop units are used to obtain reliable transmission of buffered data with the highest possible speed.

Different IRQ signals are generated to indicate whether one or two values have been send from the current buffer stack.

Two general PSP transmission modes are hardware selectable. For the case that the DSP stops sending data to the buffer the PSP will send the last buffered value again if the continuous mode is chosen. Otherwise, if the single-shot-mode is selected, the PSP shuts down the communication to the TLC320AC01 and will not send further IRQ signals to the DSP.

The detailed schematics for this part is shown in Appendix B.

(3.3) The AD7244 Serial Port:

The AD7244 includes two independent DACs with passive serial ports. Simultaneous conversion is only achievable by applying the same synchronization signals to both DACs. To provide those signals is the task of the active interface that has to be designed for the DSP. It is called from now on the Dual Active Serial Interface (DASI) since it provides two independent serial ports with equal synch signals. Although not used by the AD7244 both ports of the DASI can also be used to receive data.

In general two synch signals are used for serial transmission, the shift clock (CLK) and the frame synch (FS) signal. The latter will go low for exactly 16 CLK cycles during transmission of 16 bit values. Each CLK cycle will end the transmission of one bit, starting with bit 15. While data are latched out of the DASI on rising edges of CLK, data bits are latched into DASI on falling edges of CLK.

Additionally a Data OUT (DO) signal is sent to the AD7244 to trigger the output of the converted analog signals. DO is generated just before FS will go low, to latch the previously sent and meanwhile converted value out. As a result the DAC output occurs when the next value is transmitted. This is to be kept in mind when programming the devices.

The detailed schematics for this part is shown in Appendix B.

(3.4) The Parallel Digital I/O Port:

This little board just simply provides some line drivers with one level buffers for two of the I/O addresses of the DSP. While one of them is designed for TTL data in and output, the other address provides power outputs for external relay switches. For testing purposes a third output driver is provided to connect a 14 segment LCD display with two digits.

The detailed schematics for this part is shown in Appendix B.

(3.5) The Analog Boards:

The analog boards are designed very simply. Their dimensions are kept small so that they fit into the UHV chamber of the SPM.

Since all required handshaking is done on the digital boards, the analog circuits are basically just connecting the ADCs and DACs with the incoming signal plugs. Outgoing

analog signals are preamplified and impedance matched on the board itself. Large grounded copper areas are placed on critical positions to shield analog signal wires from digital signals. The PCB layouts of those boards are shown in Appendix C.

The board which is designed for the coordinate transformation consists of a combination of amplifiers, multipliers and sin(x) generators. It realizes the following mathematical equations:

$$x' = x_{gain} \cdot x \cdot \cos\alpha + y_{gain} \cdot y \cdot \sin\alpha + x_{ofs}$$

$$y' = -x_{gain} \cdot x \cdot \sin\alpha + y_{gain} \cdot y \cdot \cos\alpha + y_{ofs}$$

$$z' = z_{gain} \cdot z + x_{slope} \cdot x' + y_{slope} \cdot y' + z_{ofs}$$

where

$x, y, z, x_{ofs}, y_{ofs}, z_{ofs}, x_{gain}, y_{gain}, z_{gain}, x_{slope}, y_{slope}$ and α

are the input signals.

Chapter 4: The Final Assembly

Once the schematics are translated into the final layout, PCB boards can be manufactured. Although it is a major task to create and to test such layouts and to assemble all electrical devices, it is not part of this work to describe the single steps.
The individually assembled and tested digital boards are finally added up to a complete stack of parties on the bus system. The stack order is shown in figure 4.1.

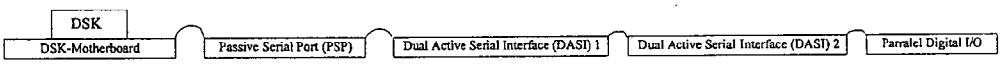


Figure 4.1 Stack Order of Bus Parties

For shielding purposes the complete stack of digital boards, including the power supply is mounted in a closed aluminum box. A small internal fan circulates the enclosed air, to help distribute locally generated heat towards the walls of the box, which will radiate it to the environment. This is enough to keep the system from overheating. However, the power supply shows temperature related problems after about half an hour. A secondary, external fan is recommended to be utilized during idle times of the system, i.e. programming of the DSP or data processing of the PC. This secondary fan should only be turned off for real measurements, when high accuracy is needed.

For future references the outgoing connectors of the digital box are shown in figure 4.2. The names refer to the previously discussed boards, where DSP represents the DSP motherboard. The front panel of this box provides the LPT and COM connectors and a 14 digit LCD display that can be accessed by the programmer for software testing and debugging purposes as well as for mode display features.

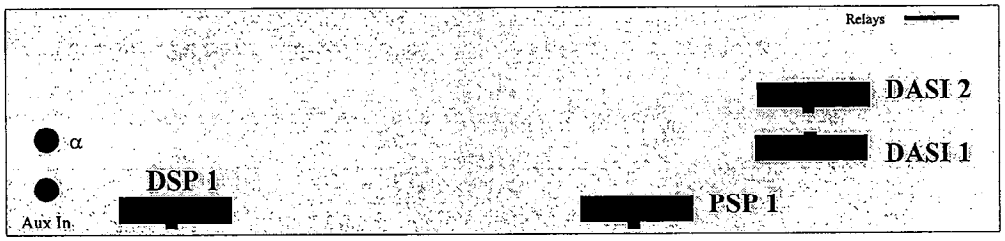


Figure 4.2 The Back Panel of the Digital Box.

Unlike the digital boards, the analog circuits need to be individually shielded with little closed metal boxes. Those boxes are carefully linked with shielded twisted wire pairs. All internal analog signals are balanced signals, which means that one wire of the twisted pair is exactly the negative signal of the other. Thus crosstalk and noise induction are kept low.

As long as those boxes are not placed directly within the vacuum chamber they are put all together into an aluminum box for further shielding.

BNC connectors are used for the final output of all analog ground-referenced signals. The front panel of the box is displayed in figure 4.3. The system can be used to drive both, tripot and tube scanner devices. The latter is selected by switching the shown switch to "**w = z**".

All scanner outputs and walker signals (w±x, w±y, ±z, WalkerL, WalkerH) are capable of driving up to 100 mA load with ±14 V voltage swing. Scan amplitudes and slope rates can be adjusted with 1% accuracy anywhere between -1 and +1 of the computer generated signals. Unfortunately the used potentiometers turned out to be the wrong choice, so that especially at high signal frequencies inductive effectes are recognizable.

To complete the documentation of the final assembly, figure 4.4 is added. It shows the back panel of the analog box. The connectors are named according to the previously mentioned definitions. The final addresses and IRQ lines are noted in this figure as well. They are determined by the order of how the digital components are strung together. Thus if this order is changed, those addresses will change accordingly.

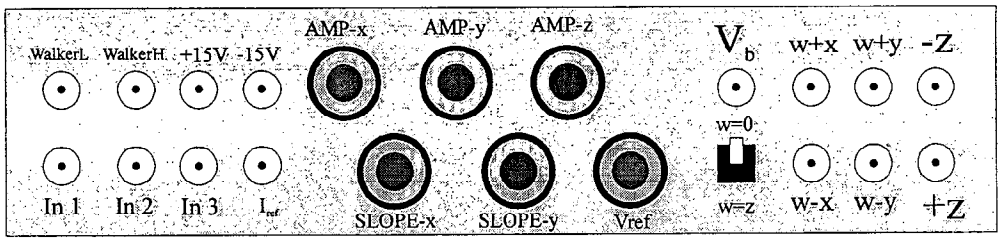


Figure 4.3 Front Panel of the Analog Box.

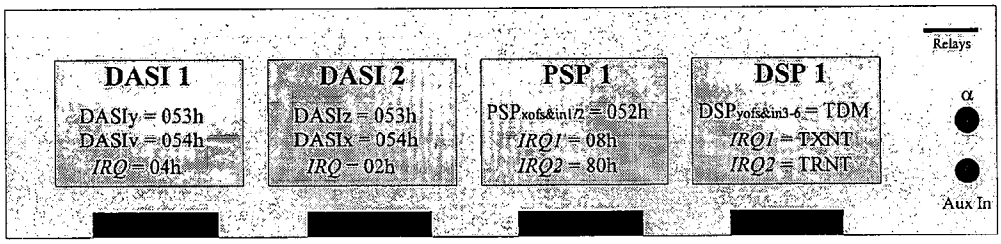


Figure 4.4 Back Panel of the Analog Box.

Chapter 5: The Software

(5.1) TI's Assembler:

Texas Instruments delivers the TMS320C50 Digital Signal Processor Starter Kit (DSK) with very simple but extremely useful assembler and debugger programs. Those can be used to convert assembly language source code into a format that is understood by the driver software that comes also with the DSK. It is the task of those drivers to transfer the DSP programs to the DSK and to start them.

The debugger is a neat tool to analyze simple programs. However, it is not a good help if real time processes are handled by the DSP. In this case the Debugger is just too slow to allow the DSP to run properly. Unfortunately most of all SPM controlling purposes have to be done in real-time. The only possible way of analyzing those programs is the usage of the LCD display of my hardware design. Its DSP address is 0059h and can be easily accessed by the programmer.

In general assembly language is not that hard to learn as it might appear to a beginner. Its small set of instructions makes it fairly easy to get a good overview of the processor's abilities. Some insights into the design of the electronics are of course necessary to be able to use this language properly.

Due to its limited set of instructions, programs are very abstract and hard to read if they are not well documented. To improve the readability it is extremely helpful to divide the program into separated function groups. Furthermore this technique helps to trace errors. However, the larger the number of program parts, the more time consuming is the editing and changing of the program. It would be very helpful to have a software tool that allows fast and easy editing and switching between program parts.

(5.2) The DSK Programmer Tool:

The motivation is to create a new PC Software Tool that covers within one single program all main features like selecting, editing, compiling and starting of DSK programs. This will allow effective and fast writing of DSP software units.

Once this task has been established, the PC software can be extended to include further options, like communication to the DSP over the printer port and final data processing.

The communication with the DSP depends on the standards which are defined by the program that has been started in the DSP. Thus it is meaningful to define a short set of general instructions that allow the PC complete control over the DSP. As soon as a standard of communication is defined, the PC only has to send a short list of numbers to tell the DSP which of the programmed options are to be executed with which parameters.

If the DSP program is written in a well organized structure, future options can be added easily.

It should be the goal of each module to be able to run independently of the PC. This helps to keep the PC available for other purposes. The PC has now the option, to let the DSP run completely self-controlled which means that the DSP executes along the programmed line of modules. However, the PC can always interrupt and choose between three basic options:
1) execute next module, 2) execute until a certain event occurs and 3) execute a single module that is not the next in line.

For general SPM image taking control, it is important to establish auto approach and scanning procedures. If the feedback is not done by analog circuits a corresponding algorithm has to be provided as well.

The auto approach is done completely digitally. It consist of three modules:

1) Withdraw the tip with a smoothly generated ramp.
2) Step the walker one step forward
3) Expand the z piezo drive while constantly checking the incoming signal. In the case of a STM the tunneling current is monitored.

As long as no tunneling current is detected, the DSP will follow those three steps in a circular motion. Once tunneling occurred the scanning mode is activated. It consists of basically two modules which let the tip scan one step at a time in either the x or y direction.

If digital feedback is provided, the x and y positioning has to be done very smoothly with constant monitoring of the incoming signal, i.e. the tunneling current.

While scanning along the surface of the sample, the DSP transmits pre-processed image data to the PC. Meanwhile the PC is displaying and storing all information.

Thanks to *Günter Guttroff* I did not need to spent much time on writing display procedures for the PC. Günter kindly let me include parts of the object code of one of his data processing programs.

All available DSP instructions are mentioned in table 5.1. The precise usage can be read out of the source code of the program, which is partly printed in the appendix. Only selected procedures could be added to the appendix since the complete source code of the PC program covers almost 100 pages in small letters. Nonetheless a complete printout of the DSP program is included in the appendix, since it is very essential for future references.

Another program module has not yet been mentioned. The following chapter will therefore concentrate on the topic of dynamic range expansion.

Instruction Code	Description	Instruction Code	Description
00h	Halt any process	stuv14h	Set Walker Amplitude
88h	Continuos mode	stuv24h	Set Dots per Cycle
77h	Next Step	stuv34h	Set I_{ref}-hi for Approach
y9h	(See Table 5.2)	stuv44h	Set I_{ref}-range (Approach)
16h	Walk left	stuv54h	Set I_{ref}-hi for Scanning
26h	Walk forward	stuv64h	Set I_{ref}-range (Scanning)
36h	Walk right	stuv74h	Set x Step Size
46h	Walk backward	stuv84h	Set y Step Size
56h	Contract z	stuv94h	Set Bias Voltage
66h	Expand z with Read	stuvA4h	Set x Position
67h	Scan x until end of line	stuvB4h	Set y Position
68h	Scan Bias Voltage	stuvC4h	Set Bias Voltage Step Size
stuv5h	Set Step counter	stuvF4h	Set z Position

Table 5.1 Instructions.

Instruction Code	Execute until	Instruction Code	Execute until
19h	Contract z	59h	Scan x
29h	Walk	69h	Scan until End of Image
39h	Expand z with Read	79h	Scan until z Out of Range
49h	(used for PSP Setup)	D9h	Scan y

Table 5.2: 'Execute until' Instruction. The DSP will stop before the chosen Event will be executed.

(5.3) Dynamic Range Expansion:

First done by Feenstra *et al.* [2], the method of expanding the dynamic range of the acquisition system by simultaneous z and V scanning still bears mathematical problems.

The idea is simple in principle. The detected tunneling current I_t of an STM experiment follows roughly a power law dependence on the applied bias voltage V_b. Thus if V_b is varied over a large range, I_t will change dramatically. Resolution problems are unavoidable unless efforts are taken to boost up small current signals. For this purpose Feenstra could establish a very precise tip movement that pushed the tip closer to the sample while V_b was decreasing. This method takes the exponential z dependence of I_t into account which will amplify the detected signal if z is decreased. The applied V and z signals are shown in figure 5.1

Problems occur as soon as physically relevant results need to be drawn from the obtained set of data. To be able to calculate band energy profiles for the studied sample all tunneling information must refer to a constant separation z_0 of tip and sample. Since all data points were taken with simultaneously changed z and V values it is impossible to compare any two data points with each other. A very precise $I(V,z)$ relation is thus necessary to refer all data to the reference position z_0. Unfortunately this relation is not yet known perfectly. Only estimates were done which are assumed to be equal for all x,y positions of the scanning area. The quality of the derived physical conclusions thus depends significantly on the accuracy of this estimate.

In the following a different method is introduced that describes a new technique of expanding the dynamic range of the acquisition system based on the same physical background.

Assuming that the tunneling current is a very low noise signal, methods of linear algebra can be used to compute a polynomial relation between two sets of data, which were taken for identical V_b values, but at two different z levels. In contrast to the previously described

method, z is held constant for each set of data. The applied V and z signals are plotted in figure 5.2.

The idea is to let the feedback move the tip to a start position z_0, before it is stopped. Then V_b stepping is started, until I_t drops below a determined threshold. The feedback is turned on again to bring the current back into range. The new z position is z_1 and the feedback is stopped again. V_b will be changed back by N steps and scanning will be continued. Thus the first upcoming N data points will have identical V_b values as the last N data points taken at z_0. A polynomial P_1 of order $(N-1)$ can be computed, that relates those N $I_t(z_1,V)$ values back to the $I_t(z_0,V)$ values taken before. If V_b scanning is continued, further data points $I_t(z_1,V)$ can be taken until I_t drops again under the determined threshold. While taking those data the DSP can calculate in real-time what the according $I_t(z_0,V)$ data would have been, just by using the polynomial relation P_1. The PC will thus receive immediately the computed information, all referenced to the same position z_0.

The feedback can be turned on again to bring I_t in range. z_2 is the new position where the feedback is stopped. V_b is steeped back, scanning will be started and a new polynomial P_2 can be determined to relate z_2 values back to z_1. Since P_1 is still known, z_2 values can be related even back to z_0 by combining P_2 and P_1. This can be done in one step, if P_2 is calculated by using the already z_0 related z_1 values in this computation.

This method can be repeated several times, before V_b scanning is finished.

It is not required to send any z-position information to the PC.

No problems will occur as long as the tip positioning is stable enough and the tunneling current steady. However, if too many z steps are taken then involved errors of the polynomial fitting will increase towards the end of the scanning, since all values at z_j are related to z_0 by using all previous polynoms P_1 to P_j. Thus a very careful choice of N and the I_t threshold is essential.

If the feedback is not used to reposition the tip, but if equal sized z steps are taken, then a further improvement is possible. Since the tip has to move first towards the sample and then, after V_b crosses zero, away from the sample, there will be identical z -positions for two different V_b ranges. If hysteresis and drift effects of the piezo drivers can be neglected then the related polynomials should be identical for those two V_b ranges with same z level. This can be taken into account to eliminate parts of the involved mathematical error.

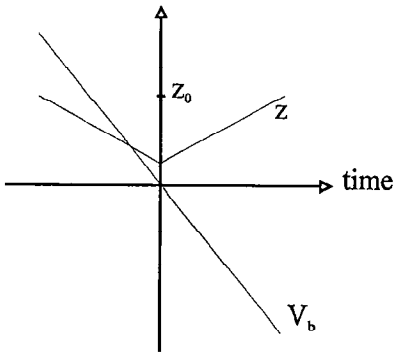


Figure 5.1 Feenstra's method.

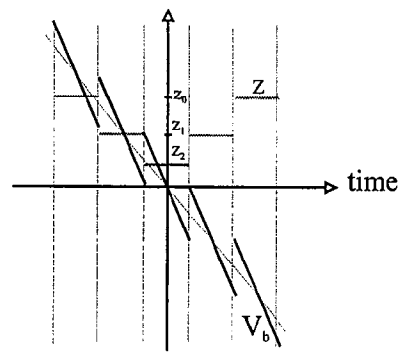


Figure 5.2 Tasler's method.

The calculation of the polynomial is very simple using matrices. Assuming $N=6$, P_1 will be of the order 5. The second set of data is called x_1 to x_6 and needs to be referred to the first set, called y_1 to y_6. The equation for x_1 is:

$$P_1(x_1) = a + bx_1 + cx_1^2 + dx_1^3 + ex_1^4 + fx_1^5 = y_1$$

By writing also all other equations down, we will obtain the system matrix S:

$$S := \begin{bmatrix} 1 & x1 & x1^2 & x1^3 & x1^4 & x1^5 \\ 1 & x2 & x2^2 & x2^3 & x2^4 & x2^5 \\ 1 & x3 & x3^2 & x3^3 & x3^4 & x3^5 \\ 1 & x4 & x4^2 & x4^3 & x4^4 & x4^5 \\ 1 & x5 & x5^2 & x5^3 & x5^4 & x5^5 \\ 1 & x6 & x6^2 & x6^3 & x6^4 & x6^5 \end{bmatrix}$$

A mathematical rule says, that the coefficient a is expressed by:

$$a = \frac{\det[A]}{\det[S]}$$

where A is formed, by replacing the first column of S with y_1 to y_6

$$A := \begin{bmatrix} y_1 & x_1 & x_1^2 & x_1^3 & x_1^4 & x_1^5 \\ y_2 & x_2 & x_2^2 & x_2^3 & x_2^4 & x_2^5 \\ y_3 & x_3 & x_3^2 & x_3^3 & x_3^4 & x_3^5 \\ y_4 & x_4 & x_4^2 & x_4^3 & x_4^4 & x_4^5 \\ y_5 & x_5 & x_5^2 & x_5^3 & x_5^4 & x_5^5 \\ y_6 & x_6 & x_6^2 & x_6^3 & x_6^4 & x_6^5 \end{bmatrix}$$

The coefficient b is calculated analogously to a. The matrix B is created by replacing the second column of S with y_1 to y_6.

Also the coefficients c through e can be computed using the same technique.

While the system determinant can be simplified:

$$\det[S] = \prod_{i=2}^{N} \prod_{j=1}^{i-1} (x_j - x_i)$$

it is not possible to simplify A, B, C, D and E in general. However, the calculation comes down to a relatively small number of multiplications and summations. The DSP is able to compute a product and a sum in 50 ns, so that the calculation is done in several milliseconds

Chapter 6: Results and Conclusions

This work was able to discuss and design a new setup of a universal SPM control and data acquisition unit. Furthermore all schematics were transformed into PCB boards, tested and finally assembled. A unique and flexible way of communication between the new system and any PC is established and a software tool written that allows easy programming and executing of DSP programs with an IBM compatible PC.

A digital feedback algorithm is also included and successfully tested. Not yet finished is the program described in Chapter 5. However, parts have already been tested and appear to work as predicted.

Unfortunately no high quality STM was available during the last weeks of this project, so that not even preliminary data could be taken. However some studies with an old STM, which is not yet optimized, could be achieved. Although the low frequency components of the initial noise of that system exceeded 15% of the measured signal amplitude, first rough images of a graphite sample could be taken.

The result is shown in figure 6.1. Although the patterns look like noise effects, it appears to be a real structure. The image contains 256*256 data points, each of which took 0.45 seconds to be recorded. This seems to be unusually long but is explained by the averaging that has to be done to filter the immense noise. The change in the pattern of the image is apparently related to vibration noise that displaced the tip permanently.

The future goal is to set up the above described control system with a highly optimized STM. To obtain the resolution of electronic states of an GaAs/AlGaAs/GaAs quantum well a low temperature STM in UHV is necessary. The program is already capable of taking topographic images. All parameters, like bias voltage and reference currents can be set by software. Thus an adjustment of the program should be practicable, especially if an STM experienced person can provide the information about optimal settings.

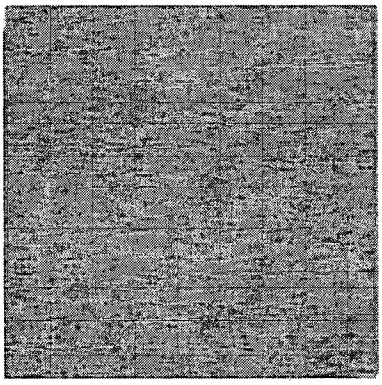
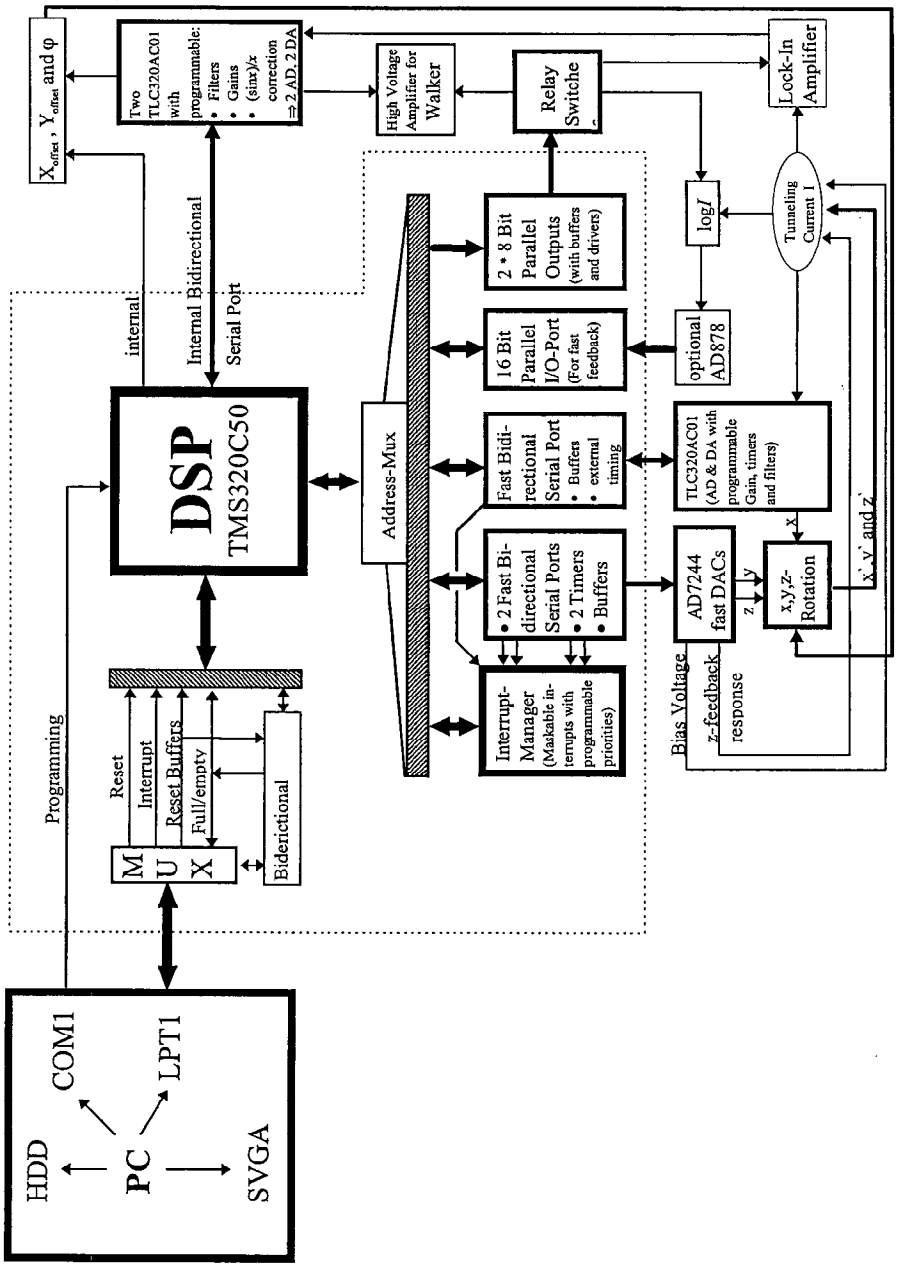


Figure 6.1 First Graphite Image.

Appendix

(Appendix A): Block Diagram of SPM Control Unit



(Appendix B): Schematics:

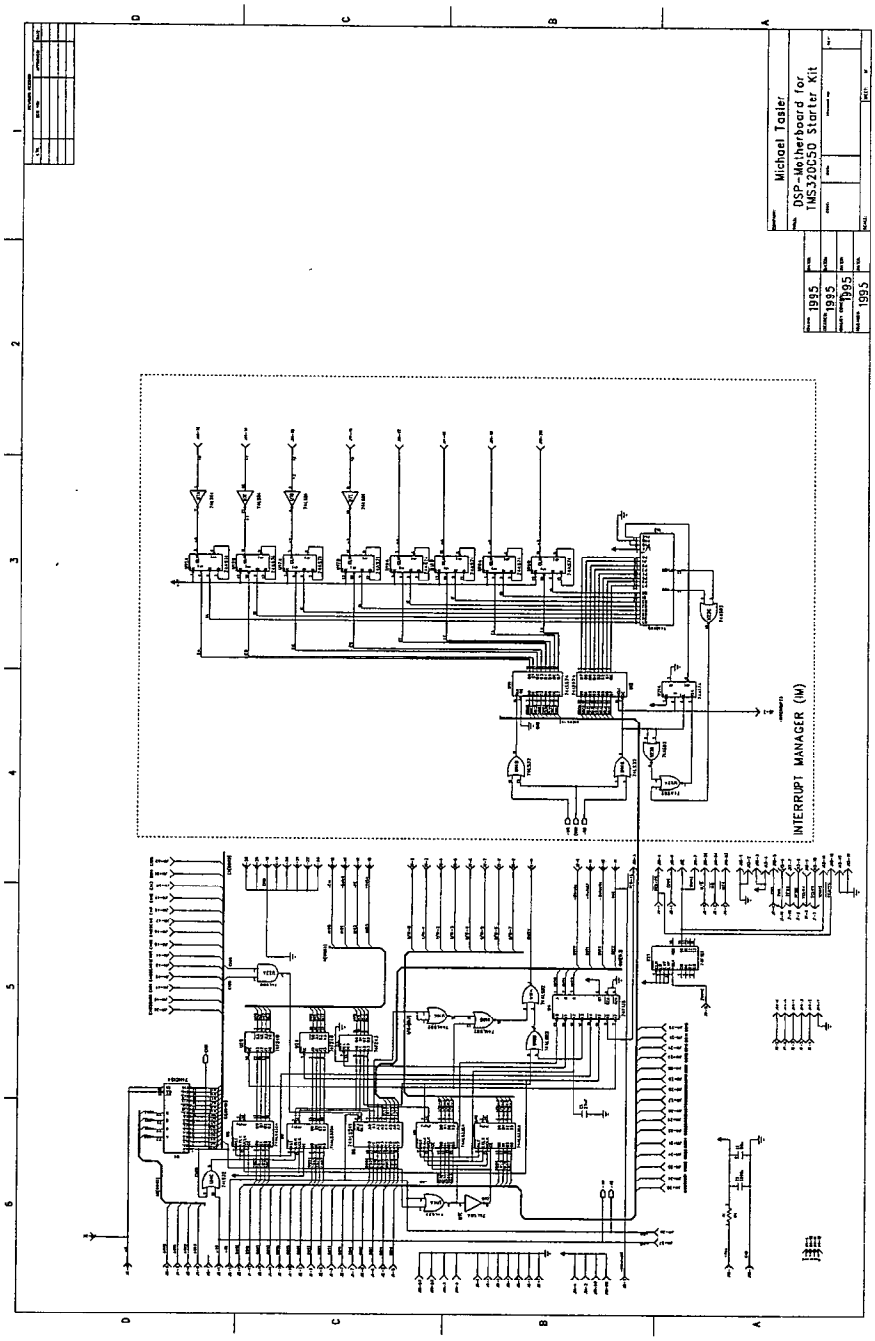


Figure B.1: Schematic of the DSP-Motherboard

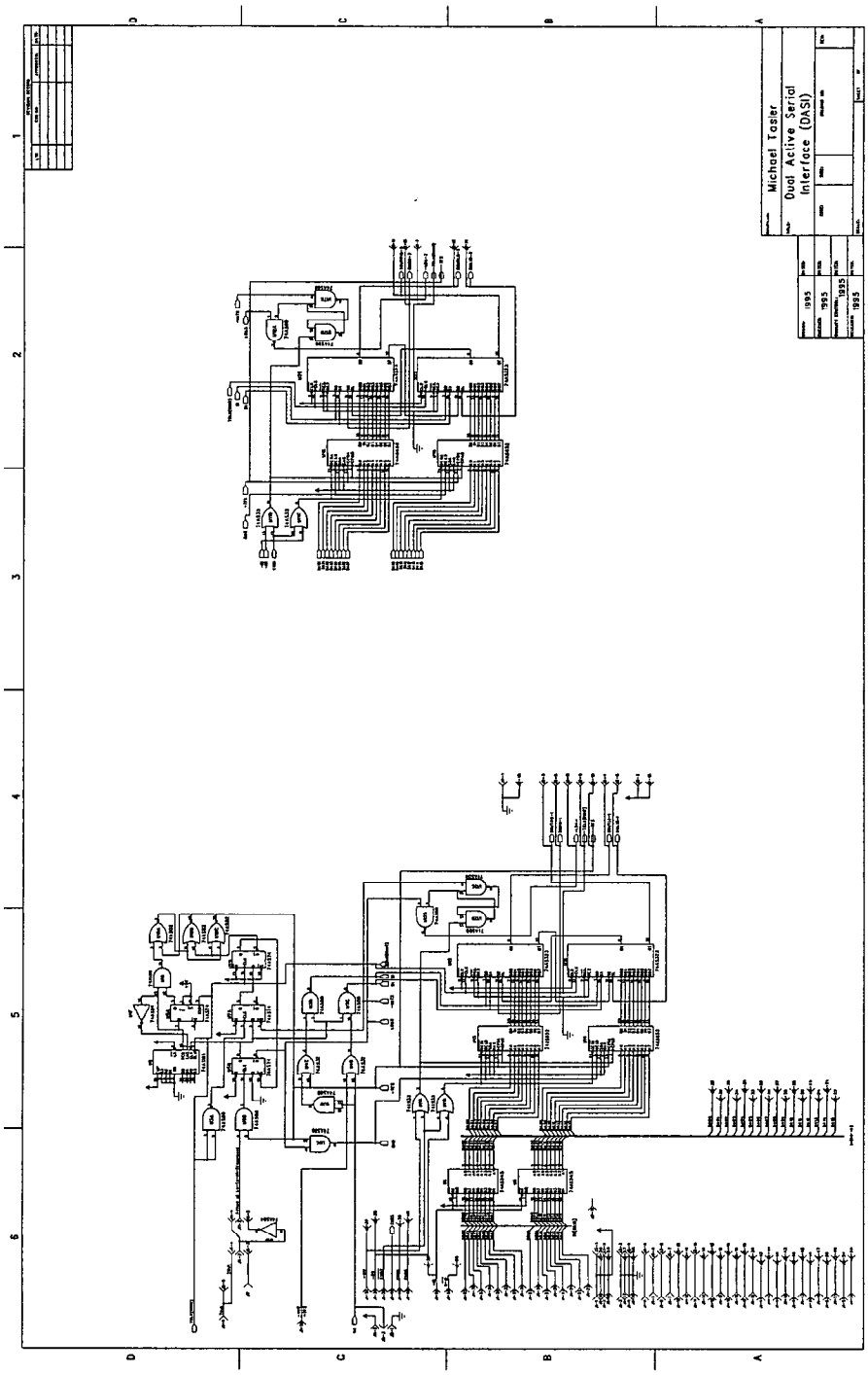


Figure B.2: Schematic of DASI

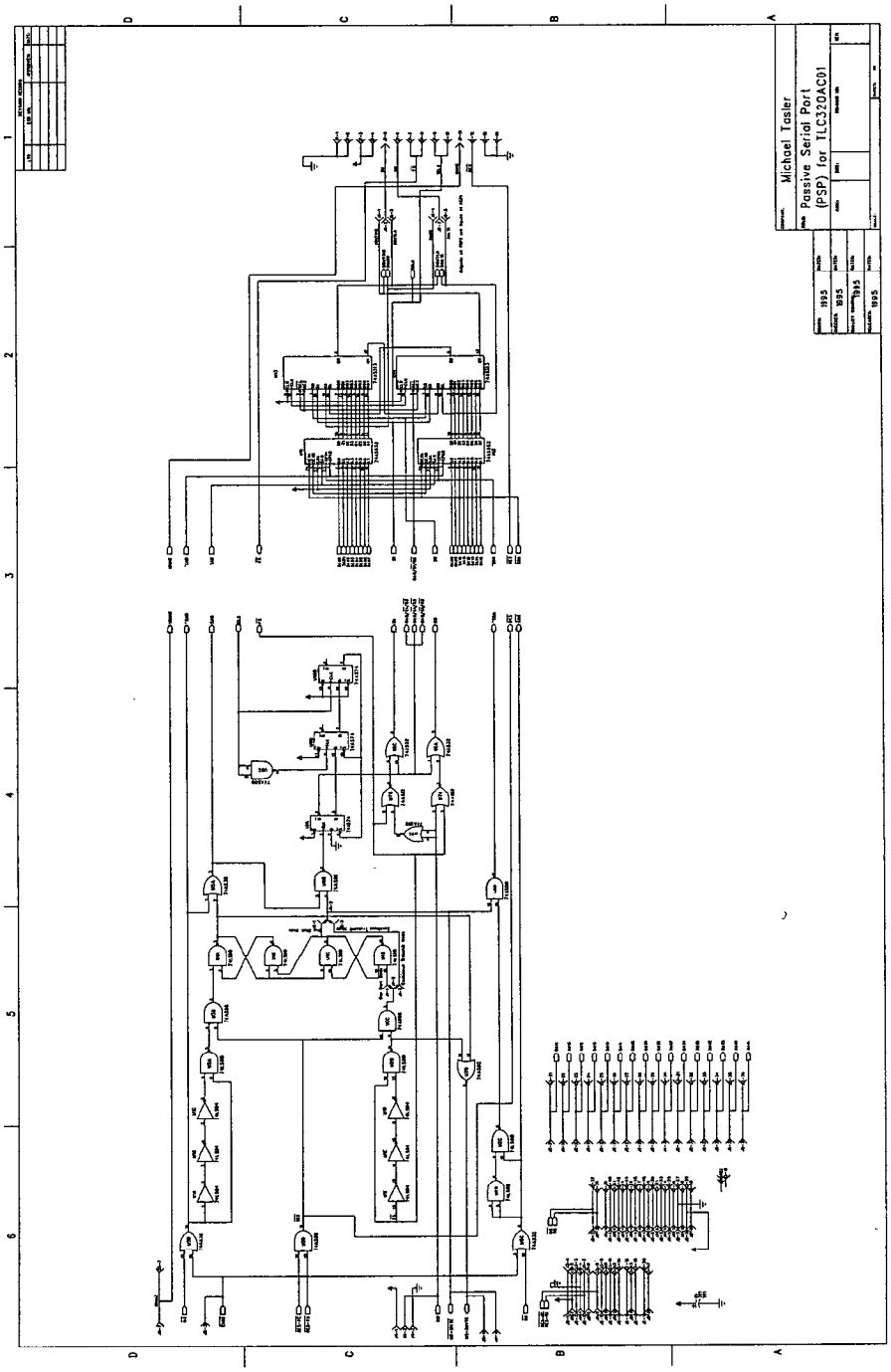


Figure B.3: Schematic of PSP

(Appendix C): PCB Layouts:

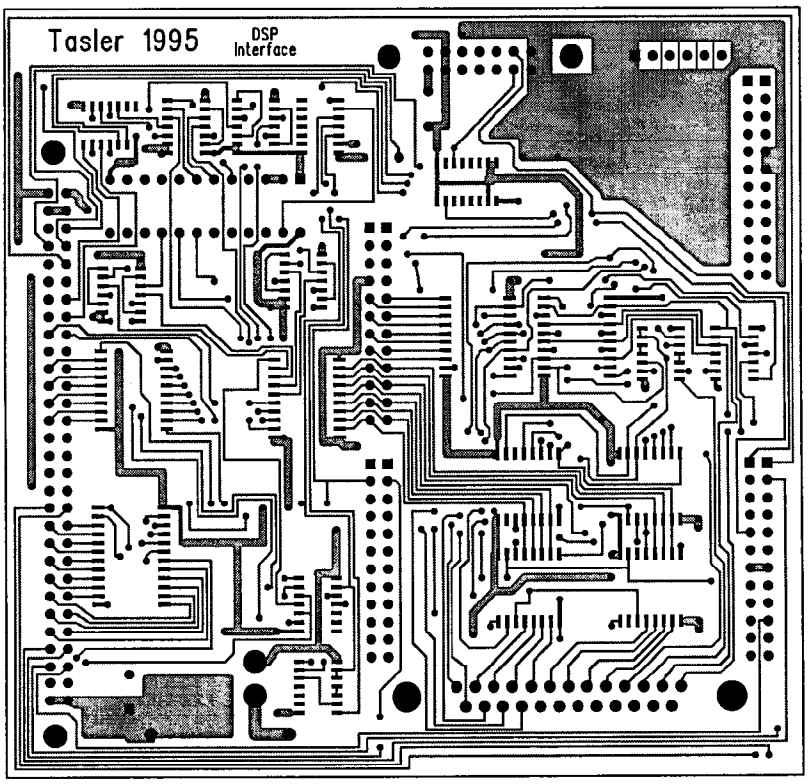


Figure C.1.1: PCB Layout of the DSP Motherboard (Component side)

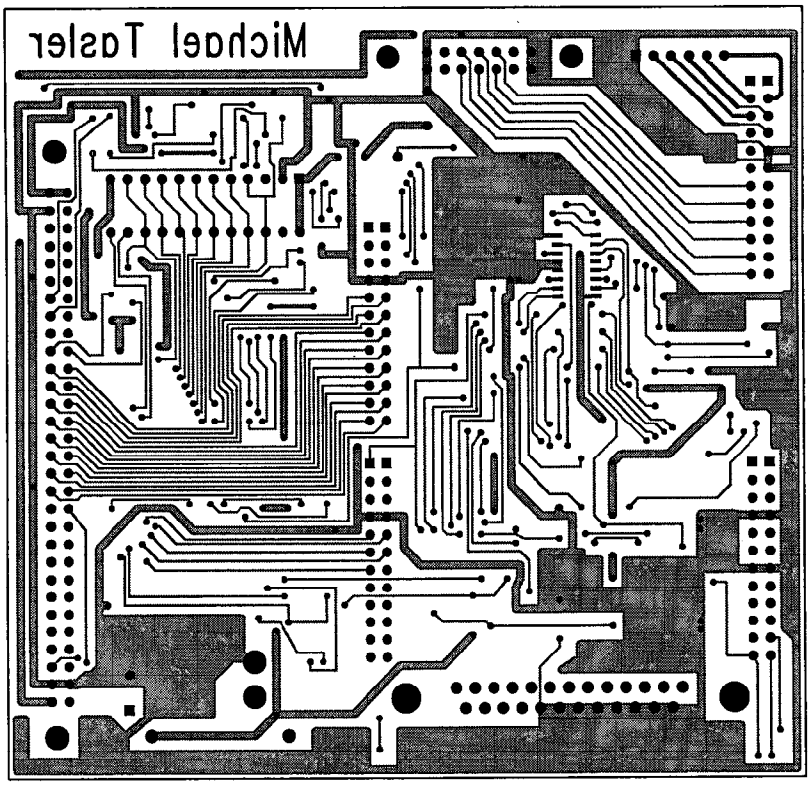


Figure C.1.2: PCB Layout of the DSP Motherboard (Solder side)

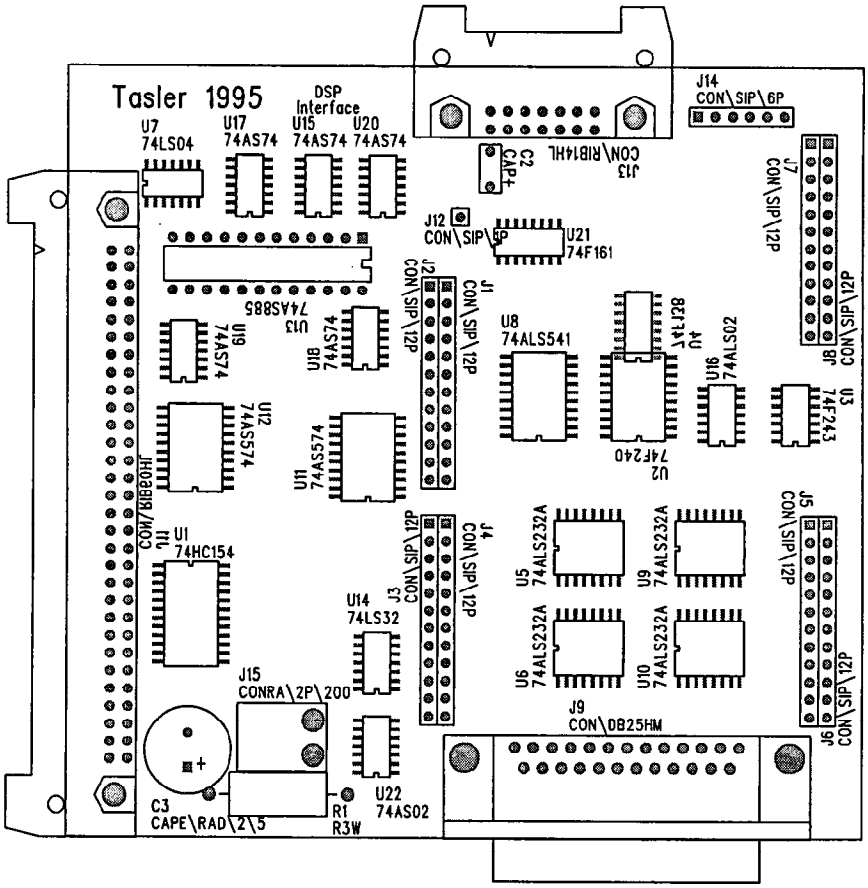


Figure C.1.3: PCB Layout of the DSP Motherboard (Components)

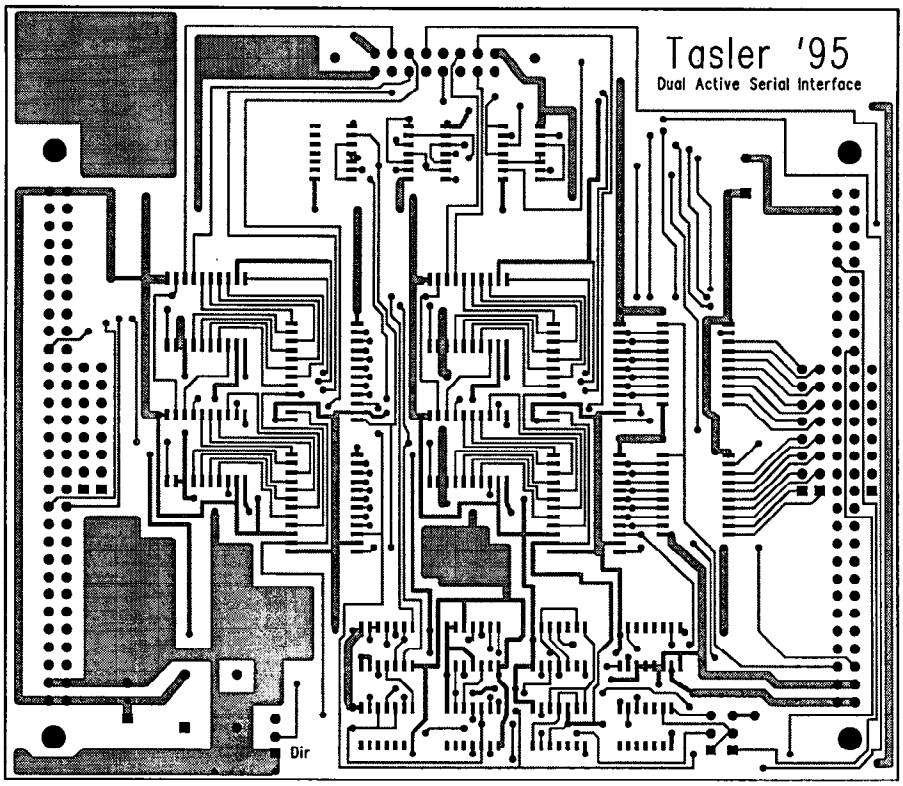


Figure C.2.1: PCB Layout of DASI (Component side)

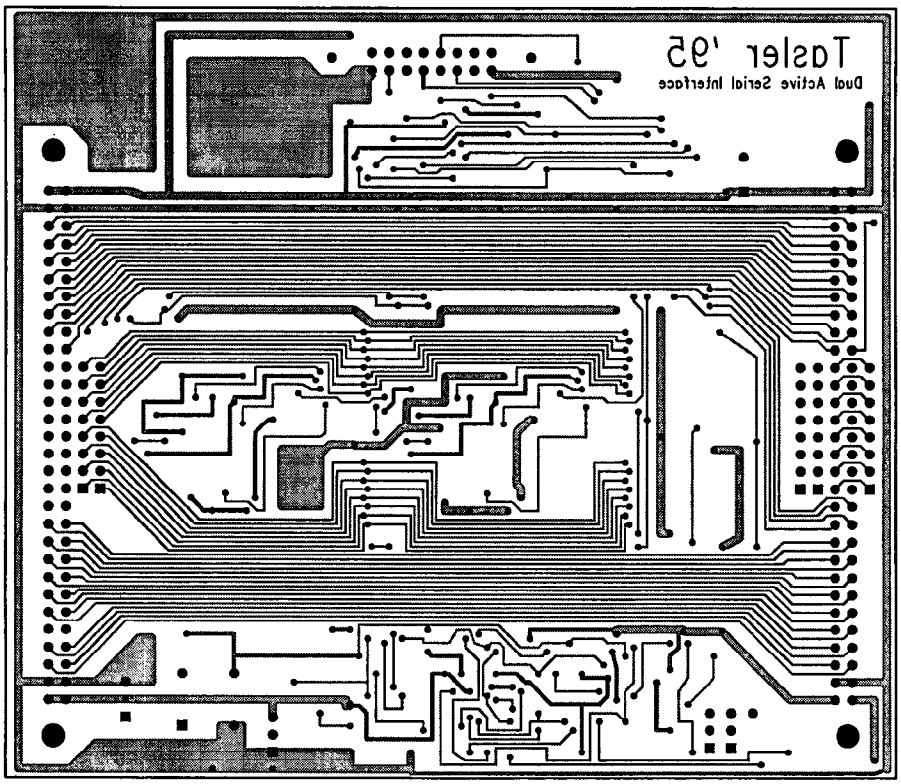


Figure C.2.2: PCB Layout of DASI (Solder side)

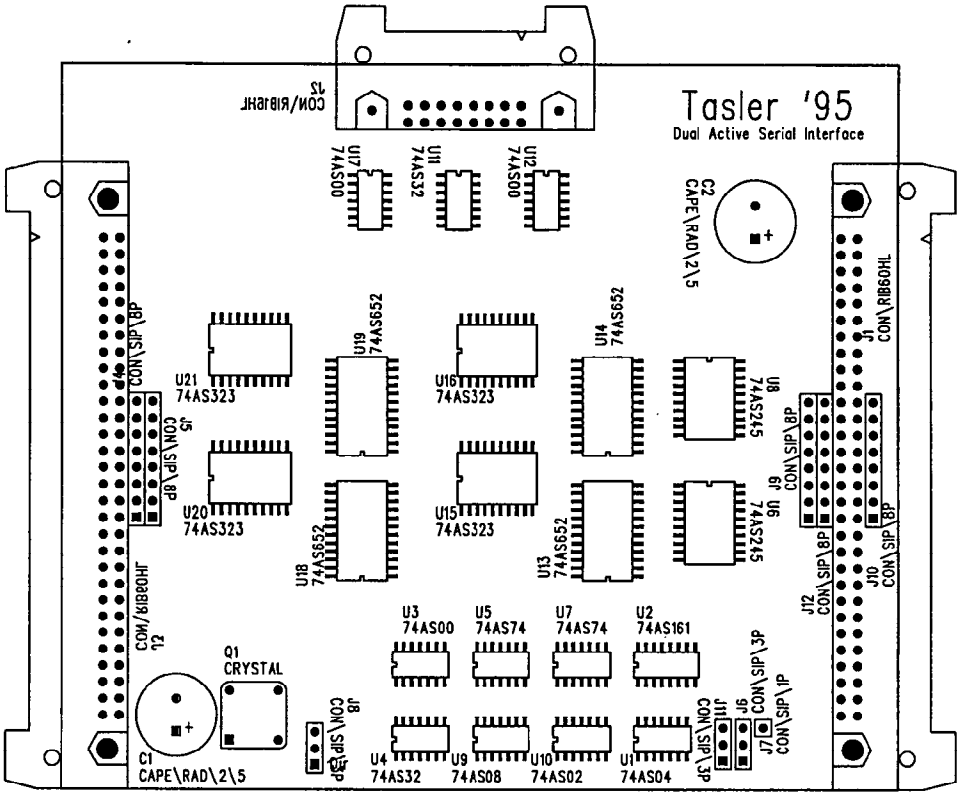


Figure C.2.3: PCB Layout of DASI (Components)

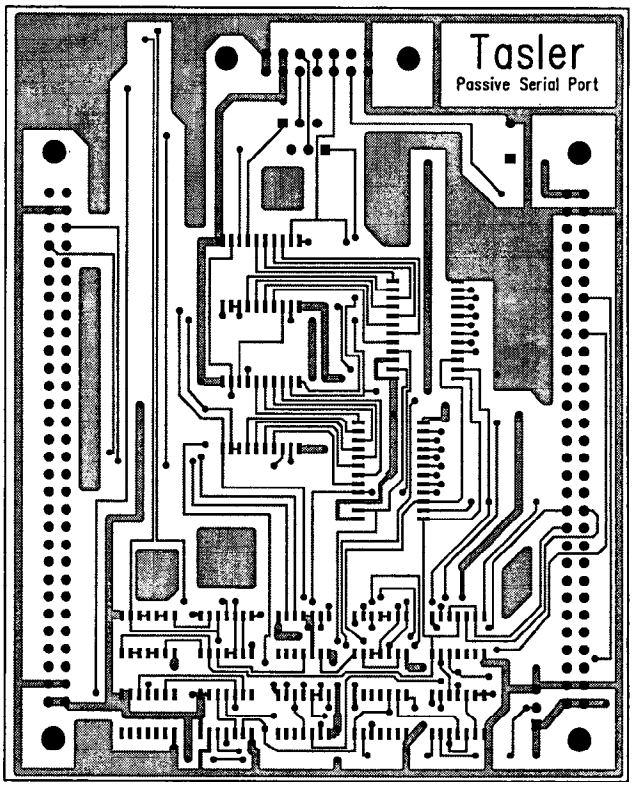


Figure C.3.1: PCB Layout of PSP (Component side)

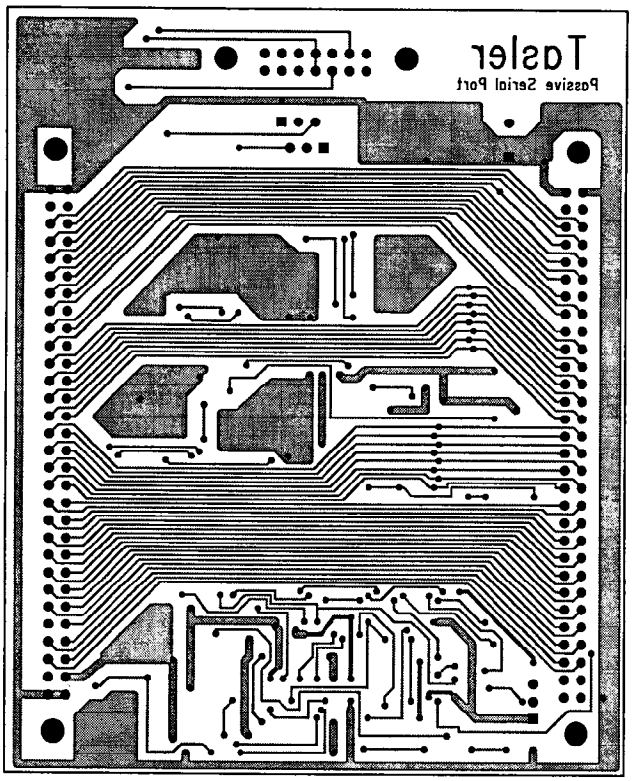


Figure C.3.2: PCB Layout of PSP (Solder side)

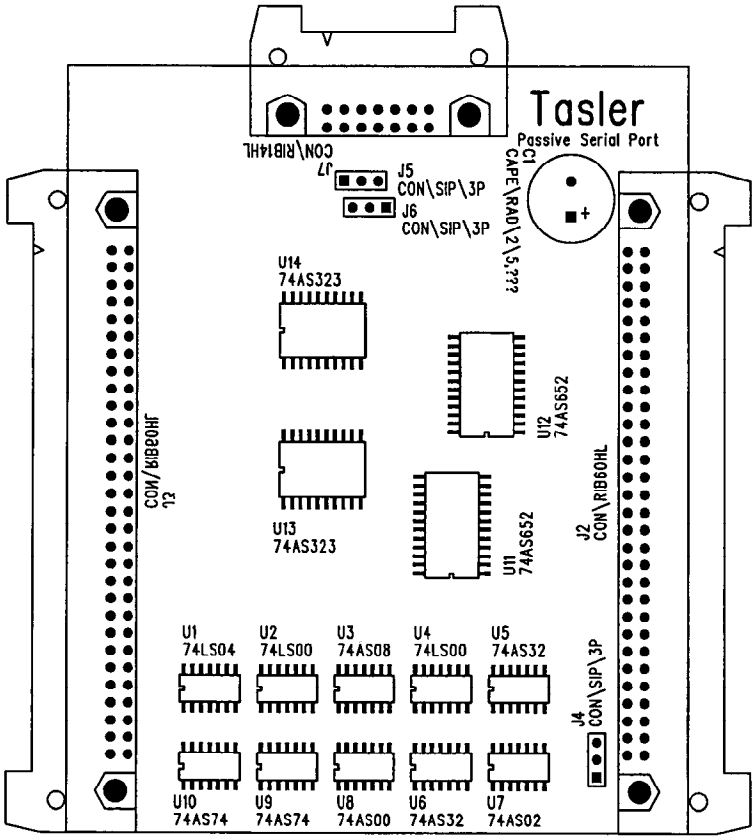


Figure C.3.3: PCB Layout of PSP (Components)

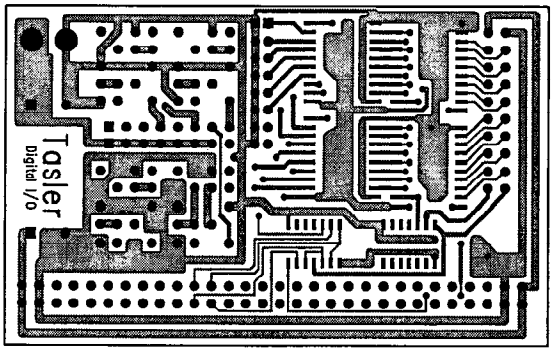


Figure C.4.1: PCB Layout of PIO (Component side)

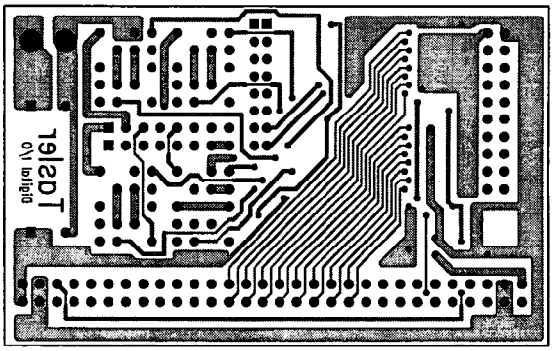


Figure C.4.2: PCB Layout of PIO (Solder side)

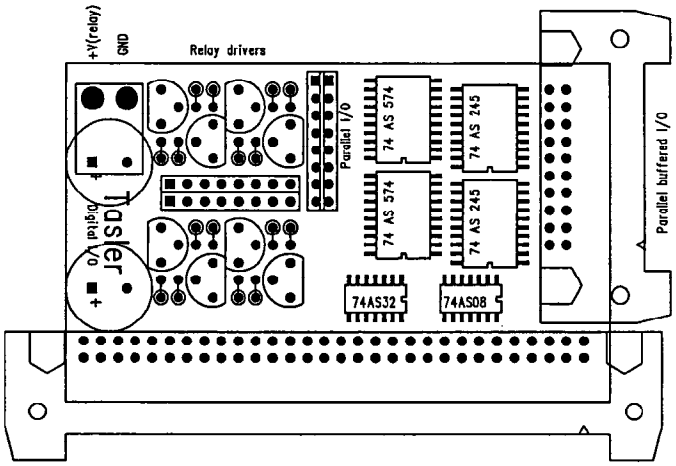


Figure C.4.3: PCB Layout of PIO (Components)

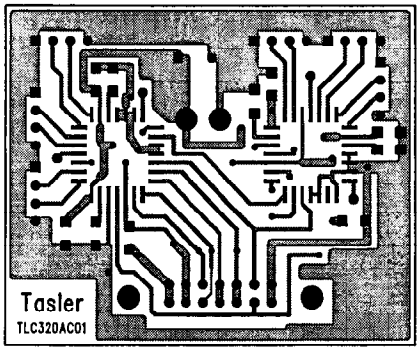


Figure C.5.1: PCB Layout of TLC320AC01 (Component side)

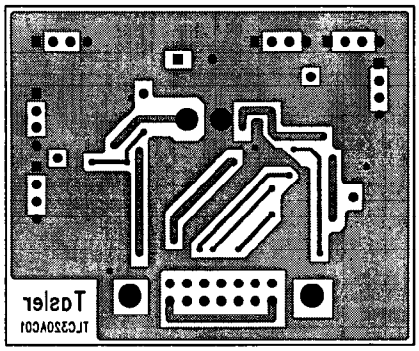


Figure C.5.2: PCB Layout of TLC320AC01 (Solder side)

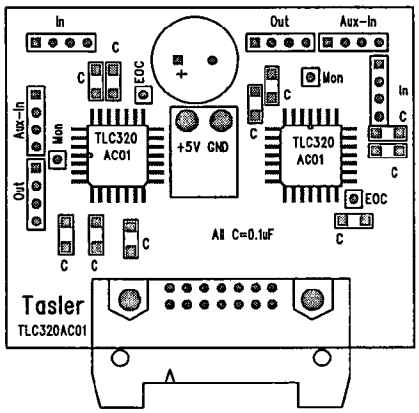


Figure C.5.3: PCB Layout of TLC320AC01 (Components)

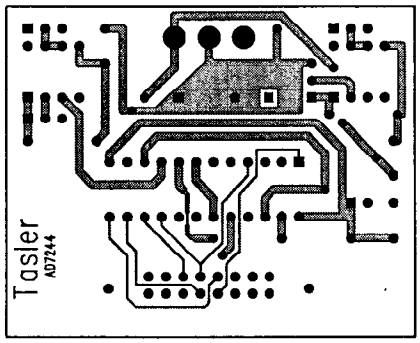


Figure C.6.1: PCB Layout of AD7244 (Component side)

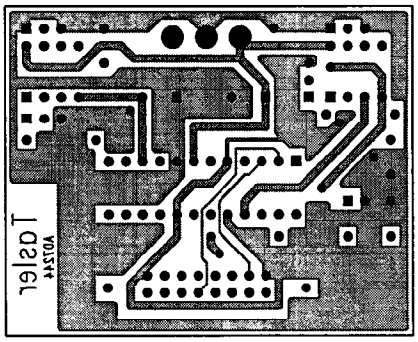


Figure C.6.2: PCB Layout of AD7244 (Solder side)

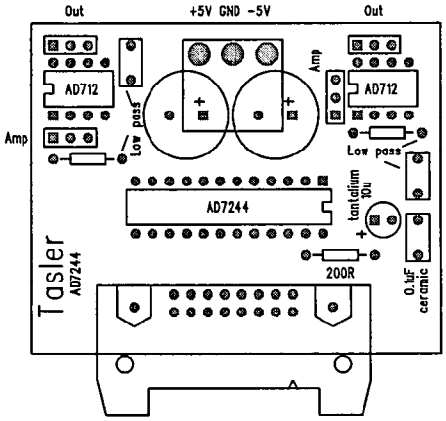


Figure C.6.3: PCB Layout of AD7244 (Components)

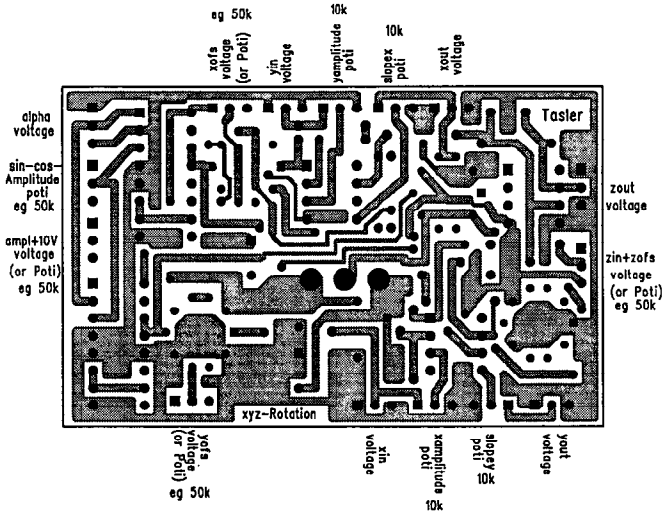


Figure C.7.1: PCB Layout of Coordinate Rotation (Component side)

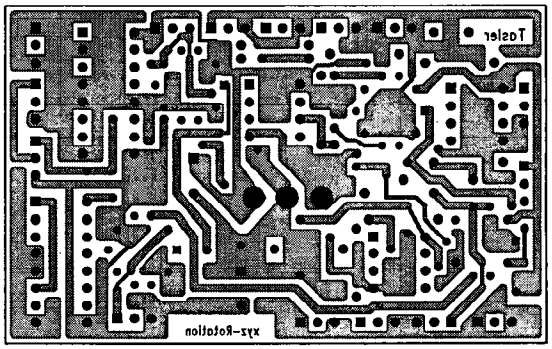


Figure C.7.2: PCB Layout of Coordinate Rotation (Solder side)

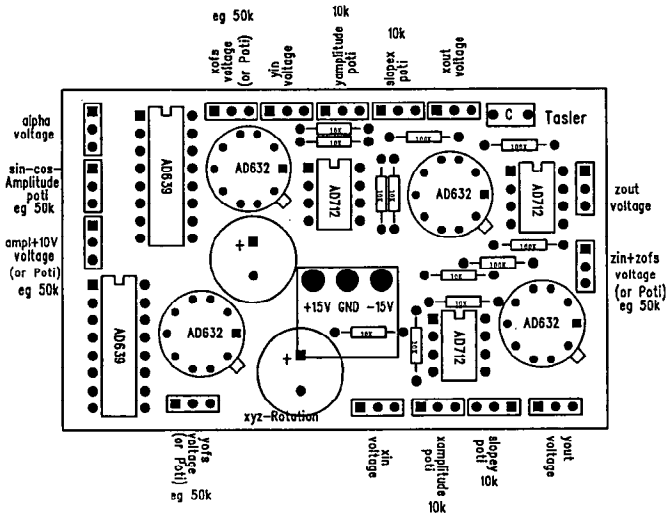


Figure C.7.3: PCB Layout of Coordinate Rotation (Components)

(Appendix D): The PC Software:

This appendix contains the source code of selected C++ procedures of the softwar tool, which is discussed in Chapter 5.

The PC to DSP Comunication Procedures:

```
int outinstruction(int inst1, int inst2, int inst3)
{ // 0 is transfered to 1,1 while 1 is transfere to 1,2
 unsigned char readback[21];
 unsigned char check[20];
 unsigned char check2[20];
 int giveback=0;
 int cnt=0;
 int cnt2=0;
 int a1,a2,a3,a4;
 int i;
 char *ptr;
 inst1=inst1&15;
 do {
  do {
   strcpy(readback,outvalue(inst1));
   check[0]=129;
   check[2]=0;
   check[3]=0;
   if(inst1>1) check[1]=inst1;else {check[1]=1;check[2]=1+inst1;}
   ptr=strstr(readback,check);
   if(!ptr)
   {//not found
    cnt=0;
    check[0]=255;
    check[1]=255;
    check[2]=1;
    check[3]=1;
    check[4]=0;
    while((cnt<6)&&(!(ptr=strstr(outvalue(0),check)))) cnt++;
    if(cnt==6) giveback=-1;
   } else giveback=1;
   cnt2++;
  } while((giveback==0)&&(cnt2<3));
  i=0;
  if(giveback>0)
  { // here inst1 is send successfully.
   ///////////
   if(inst1!=5)
   {
    giveback=0;
    strcpy(readback,outvalue(inst2));
    if(inst1!=4)
    {
     check[0]=255;
     check[1]=255;
     check[3]=0;
     check[4]=0;

     if(inst1>1) check[2]=inst1;else {i=1;check[2]=1;check[3]=1+inst1;}

    } else
```

```
   {
    check[0]=126;
    if(inst2>1) check[1]=inst2;else {check[1]=1;check[2]=1+inst2;}
    check[2]=0;
   }
   ptr=strstr(readback,check);
   if(!ptr)
   {//not found
    cnt=0;
    check[0]=255;
    check[1]=255;
    check[2]=1;
    check[3]=1;
    check[4]=0;
'/
    do {
     strcpy(check2,outvalue(0));// just before cnt2++ BEFORE else. After check[4]=0
     ptr=strstr(check2,check);
     cnt++;
    } while((cnt<6)&&(!ptr));

    while((cnt<6)&&(!(ptr=strstr(outvalue(0),check)))) cnt++;
    if(cnt==6) giveback=-1; else giveback=0;
   //} else giveback=1;
   //
   } else if((inst1!=5)&&(inst1!=4))
   {
    if(check2[ptr-check2+3+i]==1) i++;
    if(check2[ptr-check2+4+i]==1) i++;
    giveback=check2[ptr-check2+5+i];
   } else giveback=1;

   cnt2++;
  }
 }
} while((giveback==0)&&(cnt2<3));
////////
if((giveback>0)&&( (inst1==4)||(inst1==5) ))
{
 a1=(inst3&15);
 outvalue(a1);
 a2=(inst3&240)/16;
 outvalue(a2);
 a3=(inst3&3840)/256;
 outvalue(a3);
 a4=(inst3&61440)/4096;
 if((inst1==4)&&(inst2<3)) a4=a4+8*256;
 strcpy(readback,outvalue(a4));
 check[0]=255;
 check[1]=255;
 check[2]=inst1;
 a1=a1+a2*16;
 i=0;
 if(a1>1) check[3]=a1;else {check[3]=1;check[4]=1+a1;i++;}
 a3=a3+a4*16;
 if(a3>1) check[4+i]=a3;else {check[4+i]=1;check[5+i]=1+a3;i++;}
 check[5+i]=0;
 if(!(ptr=strstr(readback,check))) giveback=-1; else giveback=1;
}
return giveback;
}

char* outvalue(int outval)
{
```

```
unsigned char rdback[20];
int out=888;
int control=890;
int in=889;
int invalue;
int cnt;
int fla;
fla=(int)(outval/256);
outval=outval&255;

outportb(control, 3);
outportb(out,outval);
outportb(out,outval);
outportb(out,outval);
outportb(control, 6);
outportb(control, 6);
outportb(control, 6);
outportb(control, 3);
outportb(control, 3);
outportb(control, 3);

// don't forget to establish reset of buffers after dsp-prgstart
 outportb(control,5);
 cnt=0;
 if(fla) delay(fla*4);
 do {
  cnt++;
  invalue=(inportb(in)&120) / 8;
 } while((!(invalue&2))&&(cnt<10000));

 outportb(control,3);

 cnt=0;
 if(invalue&2)
 {
  do {
   outportb(control,15);
   invalue=(inportb(in)&120) / 8;
   outportb(control,11);
   outportb(control,13);
   invalue=255-(invalue+(inportb(in)&120)*2);
   if(invalue>1) rdback[cnt]=invalue;
   else {rdback[cnt]=1;cnt++;rdback[cnt]=1+invalue;}
   cnt++;
   outportb(control,11);
  } while((inportb(in)&128)&&(cnt<19));// not yet empty!!
  if(cnt==19) {rdback[cnt]=1;cnt++;}
 }
 rdback[cnt]=0;
 outportb(control,3);
 // read until empty
 return rdback;
}

int Do_it(int inst1, int inst2, int inst3)
{
 int z=0;
 int ar1;

 do {
  ar1=outinstruction(inst1,inst2,inst3);
  z++;
 } while((z<5)&&(ar1<1));       // Reset counter
```

```
 if(z==5) {beep(1000,300); printf("DSP does not respond!"); ar1=0;}
 return ar1;
}

int GetDSPvalue()
{
 int invalue;
 int control=890;
 int in=889;
 int out=888;
 int cnt=0;

outportb(control,5);
 do {
  cnt++;
  invalue=(inportb(in)&120) / 8;
 } while((!(invalue&2))&&(cnt<10000));
 if(cnt<10000)
 {
  outportb(control,3);
  outportb(control,15);
  invalue=(inportb(in)&120) / 8;
  outportb(control,11);
  outportb(control,13);
  invalue=255-(invalue+(inportb(in)&120)*2);
 } else invalue=-1;
 return invalue;
}
```

The Scanning Procedure:

```
void scan(int xnumber, int ynumber, int factor, int factorz, int voltagez, int referenceintensity)
{
  float size2;
long int summe[1024];
unsigned int readDSP[1024];// or use farmalloc and farcalloc
clrscr();
  ScanGraphicsInit(4);
int invalue;
    int control=890;
    int in=889;
    int out=888;
    int cnt=0;
    int direction;
    int x2;
    int choice;
    long int summ=0;

  for(int y=0; y<ymax; y++)
  {
  outportb(control,5);
  Do_it(9,5,0);
  for(int x=0; x<xmax-1; x++)  /// ;x<=xmax;
  {
   outportb(control,11);
   while((!bioskey(1))&&(!(inportb(in)&128)));
   invalue=(inportb(in)&120) / 8;
   outportb(control,13);
   readDSP[x]=255-(invalue+(inportb(in)&120)*2);
```

```
/*
if(readDSP[x]&3)
{
 outportb(control,11);
 //while(!(inportb(in)&128));// if(bioskey(1)==283) {break;break;}// wait for busy <> 0 (==> not empty)
 invalue=(inportb(in)&120) / 8;
 outportb(control,13);
 readDSP[x]=255-(invalue+(inportb(in)&120)*2);
}*/
//while(!(inportb(in)&128));// if(bioskey(1)==283) {break;break;}// wait for busy <> 0 (==> not empty)
outportb(control,11);
invalue=(inportb(in)&120) / 8;
outportb(control,13);
readDSP[x]+=(long int)(255-(invalue+(inportb(in)&120)*2))*256;
}
if(bioskey(1)) if(bioskey(1)==7181) {x=xmax+1;getch();} else if (bioskey(1)==283) {x=xmax+1;y=ymax+1;getch();} el
getch();// wait for busy <> 0 (==> not empty)
/*
 asm {
  xor di,di;
  mov dx,control;
  mov al,11;
  out dx,al;
  dec dx;
 }
empty1:
 asm {
  in al,dx;
  test al,080h;
  jz empty1;
  shr al,3;
  and ax,0fh;
  test al,3h; //test if lo bits of lo byte where read (must be zero)
  jnz empty1;
  mov cx,ax;
  inc dx;
  mov al,13;
  out dx,al;
  dec dx;
  in al,dx;
  shl al,1;
  and al,0f0h;
  add cl,al;
  inc dx;
  mov ax,11;
  out dx,al;
  dec dx;
 }
empty2:
 asm {
  in al,dx;
  test al,080h;
  jz empty2;
  shr al,3;
  and al,0fh;
  add ch,al;
  inc dx;
  mov al,13;
  out dx,al;
  dec dx;
  in al,dx;
  shl al,1;
  and al,0f0h;
  add ch,al;
```

```
  xor cx,0ffffh;
  mov word ptr [di+&readDSP[0]],cx;
  inc dx;
  mov ax,11;
  out dx,al;
  dec dx;

  inc di;
  cmp di,xmax;
  jle empty1;
  }
 */
 beep(500,50);
 //movedata(FP_SEG(readDSP), FP_OFF(readDSP),FP_SEG(fptr+y*(xmax+1)), FP_OFF(fptr+y*(xmax+1)),xmax+1)
 if(y&1) direction=xmax-1; else direction=0;
 summe[y]=0;
 for(x=0;x<=xmax;x++) summe[y]+=readDSP[x];
 summe[y]/=xmax+1;
 if(y) summ=(int)((float)(summ*(y-1))/y+(float)summe[y]/y);else summ=summe[0];
 for( x=0; x<xmax; x++)
 {
  x2=abs(direction-x);
  //if(readDSP[x]<0) readDSP[x]=255;
  DisplayValue(x2,y,(readDSP[x])/256);
 }
}
/*for(y=0;y<=ymax;y++)
{
 movedata(FP_SEG(fptr+y*(xmax+1)), FP_OFF(fptr+y*(xmax+1)),FP_SEG(readDSP), FP_OFF(readDSP),xmax+1);
 for(int x=0;x<=xmax;x++)
 {
  x2=abs(direction-x);
  DisplayValue(x2-1,y,(readDSP[x]-summ)/128);
 }
}*/
}
```

The Interface between User and PC (this part shows the basic usage of the defined sets of commands):

```
void menue()
{
 int choice,choice2=0;
 int voltage        = voltagemin;
 int stepsize       = 32;
 int voltagestep    = 8
 int referenceintensity;
 int calibratecontinue;
 long int amplitude=0xa0;//=0222h => Max ampl is SQR(1-(amp*frequ)^2)
 long int dotspercycle=0x190;//=78h ==> dotsperc.*ampl <ffffh.  The real frequency
                                        // is 465kHz/2/dotspercycle
 long int Irefa=0,dIrefa=0,Irefs=0,dIrefs=0,xpos=0,ypos=0,zpos=0,vbias=0;
 int k;
 long int novell;

 steps[0]=1;
 clrscr();
 menuewin();
 statuswin();
do {
 while(choice != ESC)
 {
  _setcursortype(_NOCURSOR);
  checkkey();
  choice = get_key();
  statuswin();
  switch(choice)
  {
   case UP  : {
                Do_it(6,5,0); //Reject Tip
                delay(3);
                Do_it(5,5,steps[0]); // Set # of steps
                last=Do_it(6,2,0);      // Walk forward
                clrscr();
                printf(" Walker moved %d steps forward.",steps[0]);
                menuewin();
                break;
               }

   case DOWN : {
                Do_it(6,5,0); //Reject Tip
                delay(3);
                Do_it(5,5,steps[0]); // Set # of steps
                last=Do_it(6,4,0);      // Walk forward
                clrscr();
                printf(" Walker moved %d steps backward.",steps[0]);
                menuewin();
                break;
               }
   case LEFT  : {
                Do_it(6,5,0); //Reject Tip
                delay(3);
                Do_it(5,5,steps[0]); // Set # of steps
                last=Do_it(6,1,0);      // Walk forward
                clrscr();
                printf(" Walker moved %d steps left.",steps[0]);
                menuewin();
                break;
               }
```

```
 case RIGHT : {
                    Do_it(6,5,steps[0]); //Reject Tip
                    delay(3);
                    Do_it(5,5,steps[0]); // Set # of steps
                    last=Do_it(6,3,0);      // Walk forward
                    clrscr();
                    printf(" Walker moved %d steps right.",steps[0]);
                    menuewin();
                    break;
                   }

 case F8  : {
                    processing();
                    menuewin();
                    break;
                   }

 case F1   : {
                    DSP_Setup_Menue();
                    menuewin();
                    statuswin();
                    break;
                   }
 }
 if((0==0)||(strcmp(DSP_Prg,"No Program")))
 switch(choice)
 {
  case F2   : {


window(1,1,80,1);
textcolor(BLACK);
textbackground(WHITE);
clrscr();
printf("    D S P  - Parameter Setup,              (c) Michael Tasler 1995");
window(1,2,80,11);
textcolor(BLUE);
textbackground(CYAN);
clrscr();
printf(" F1 : Walker-Amplitude, -dots_per_cycle, and -#_of_steps (%xh, %xh, %xh)",amplitude,dotspercycle,steps[0])
 printf("\n F2 : Iref, dIref for autoapproach (0='_', 8000h='-', FFFF='~')");
 printf("\n F3 : Irefs, dIrefs for scanning (0='_', 8000h='-', FFFF='~')");
 printf("\n F4 : z-position (1='_', 2000h='-', 3FFE='~')");
 printf("\n F5 : x-position ((NOT YET))");
 printf("\n F6 : y-position ((NOT YET))");
 printf("\n F7 : Bias Voltage V (0='-', -2000h='_', 1FFF='~')");
printf("\n F8 : Contract_z: move z slowly to 3fffh");// used to be spectroscopy
printf("\n F9 : Feedback-test");
printf("\n ESC : END ");
printtime();

 _setcursortype(_NOCURSOR);
 checkkey();
 choice2 = get_key();
 statuswin();
 switch(choice2)
 {
  case F1  : {
                    _setcursortype(_NORMALCURSOR);
                    printf("\n Quasi-Amplitude of walker cycles (old value: %xh): ",amplitude);
                    novell=accept_or_change(amplitude);
                    if(novell>=0) amplitude=novell;
                    if(amplitude) Do_it(4,1,amplitude);
```

```
                    printf("\n Dots per walker cycle (old value: %xh): ",dotspercycle);
                    novell=accept_or_change(dotspercycle);
                    if(novell>=0) dotspercycle=novell;
                    if(dotspercycle) Do_it(4,2,dotspercycle);

                    printf("\n Number of walker steps (old value: %xh): ",steps[0]);
                    novell=accept_or_change(steps[0]);
                    if(novell>=0) steps[0]=novell;
                    break;
                  }
case F2:   {
                    _setcursortype(_NORMALCURSOR);
                    printf("\n Iref (old value: %xh): ",Irefa);
                    novell=accept_or_change(Irefa);
                    if(novell>=0) Irefa=novell;
                    if(Irefa) Do_it(4,3,Irefa);

                    printf("\n dIref (old value: %xh): ",dIrefa);
                    novell=accept_or_change(dIrefa);
                    if(novell>=0) dIrefa=novell;
                    if(dIrefa) Do_it(4,4,dIrefa);

                    break;
                  }
case F3:   {
                    _setcursortype(_NORMALCURSOR);
                    printf("\n Irefs (old value: %xh): ",Irefs);
                    novell=accept_or_change(Irefs);
                    if(novell>=0) Irefs=novell;
                    if(Irefs) Do_it(4,5,Irefs);

                    printf("\n dIrefs (old value: %xh): ",dIrefs);
                    novell=accept_or_change(dIrefs);
                    if(novell>=0) dIrefs=novell;
                    if(dIrefs) Do_it(4,6,dIrefs);

                    break;
                  }
case F4:   {
                    _setcursortype(_NORMALCURSOR);
                    printf("\n zpos (old value: %xh): ",zpos);
                    novell=accept_or_change(zpos);
                    if(novell>=0) zpos=novell;
                    if(zpos) Do_it(4,15,zpos);
                    delay(100);
                    if(zpos) Do_it(4,15,zpos);// twice since buffered output
                    break;
                  }
case F7:   {
                    _setcursortype(_NORMALCURSOR);
                    printf("\n vbias (old value: %xh): ",vbias);
                    novell=accept_or_change(vbias);
                    if(novell>=0) vbias=novell;
                    if(vbias) Do_it(4,9,vbias);
                    delay(100);
                    if(vbias) Do_it(4,9,vbias);
                    break;
                  }
case F8: {Do_it(6,5,0);break;}
case F9: {Do_it(6,7,0);break;}

}
                    _setcursortype(_NOCURSOR);
                    clrscr();
```

```
                    menuewin();
                    statuswin();
                    break;
                   }

case F3   : {
                   last=Do_it(7,7,0); // Do next step
                   menuewin();
                    break;
                   }
case F4          : {
                    _setcursortype(_NORMALCURSOR);
                    printf(" This option provides the possibility of setting breakpoints.");
                    printf("\n The DSP stops right before the function according to the value of the");
                    printf("\n breakpoint is about to be executed.");

                    printf("\n\n General float chard: 1-2-3-1-2-3-... until tunneling. Then 5-13-5-13-...-6 / 7");

                    printf("\n\n 0 : <cancel>                    1 : contract z");
                    printf("\n 2 : Step 1 walker step forward      3 : Expand z while monitoring I(z)");
                    printf("\n 4 : <continuous mode>              5 : Scann x position");
                    printf("\n 6 : End of Image                 7 : End of Image or z is Out of Range");
                    printf("\n 13: Scan next Line or until End of Line");
                    printf("\n\n Your choice (ENTER will confirm the old value)? ");

                     novell=accept_or_change(until);
                     if(novell>=0) until=novell;
                     _setcursortype(_NOCURSOR);
                     if(until) {if(until!=4) last=Do_it(9,until,0); else last=Do_it(8,8,0);}
                     clrscr();
                     if(last>0) printf(" DSP is executing until function %d.",until);
                     menuewin();
                     break;
                   }

 case F5         : {
                     last=Do_it(0,0,0); // pause DSP
                     printf(" DSP is paused");
                     menuewin();
                     break;
                   }

 case F6  : {
                     printf("\n Auto Approch:");
                     Do_it(6,5,0);      // rejects tip and gets AR1 away from 5 (is the case if autoapproach was don
before);
                     delay(3);
                     Do_it(5,5,1);      // Reset counter
                     Do_it(9,5,0);      // Wait for end of autoapp.
                     menuewin();
                     break;
                   }
 case F7         : {
                   scanmenue(1,1);// (should also include Voltage ramps);
                   clrscr();
                   menuewin();
                   break;
                   }
 case F10  : {
                     Do_it(6,5,0);// contract z
                     delay(3);
                     Do_it(5,5,1000);
                     Do_it(6,4,0);// walk 1000 steps backward
                     printf("\n Scanner is removed and tip rejected.");
```

```
                    break;
                  }
  }
  //++v=read_voltage(channel)+AD_voltage_offset;
  //++PlotParameters(stepsize,voltage-voltagemin,voltagemax-voltagemin,v);
}

if(strcmp(project,"No Project"))
 {
 beep(1500,50);
 window(1,23,80,25);
 textcolor(YELLOW);
 textbackground(RED);
 clrscr();
 printf("\n        You have to close the DSP-Project before you exit! <F1,F5>");
 choice=0;
 }
} while(strcmp(project,"No Project"));

Do_it(6,5,0); // reject tube
delay(3);
printf("\nEND");
voltagestep=voltagestep;
}
```

(Appendix E): The DSP Software:

This appendix contains the complete source code of the DSP communication softwar

The Root Program:

```
******* fbm7 *******
.mmregs
.include "var01.h"
.include "dspinitn.h"
.include "main.h"

.include "scannc.h"
.include "cont_zmc.h"
.include "read_ipc.h"
.include "walkerc.h"
.include "pspinit.h"
.include "calc0f3n.h" ; runs walker in different direction
.end
```

The Include File "var01.h":

```
; VAR01 contains all variables
DASIx  .set   56h
DASIy  .set   53h
DASIz  .set   55h
DASIv  .set   54h
PSPx0  .set   52h

.ds 0f00h
SIN_SW .word   0000h        ; 0 -> random noise
y      .word   0h
y1     .word   0a1h        ; frequency:  *;*;*; y1 multiplied
                                   ; the "*;*;*;*;*"-sequence must be 10000h !!!!!

; a-b-c, where a*c just smaller then 10000h. b=2c-2
;possible combinations are  249h-de0h-70h=7h,4
;Next try: 276h-ce0h-68h=dh,3
;next: 222h-eeh-0fh,3: that's it

y2     .word   0h
y21    .word   0h
y2h    .word   0h
y3     .word   0h
offi   .word   0h

ta     .word   0
;; max speed is b=10 and d=4;;15&10:super
a      .word   5
aa     .word   5
b      .word   10;... 10 was here togethet wit 4 below at d

bb     .word   2
c      .word   21
cc     .word   4
d      .word   6;4      ;5     ; a,b,c,d are the values which are going
                                          ; to be written in aa,bb,cc,dd respectively.
dd     .word   1
```

```
Ihi     .word    0ffffh
delta_I .word    07a00h
Irefhi  .word    0a000h
dIref   .word    00100h
n03fefh .word    03fefh
n08000h .word    08000h
n02000h .word    02000h

 .ds     01010h
coeff    .word    00h
ccnt     .word    0190h
offset   .word    0h
richtung .word    0h
ystep    .word    0100h
xstep    .word    0100h

rings    .word    00h
;rings   .space   0ee0h
;ringe   .word    01fffh


            .ps 0806h
int3:     B      office4
tint:     B      walker
```

The Include File "dspinitn.h":

```
;dspinit.h : does all the register setups of the dsp and aic's

 .ps 0a00h
 .entry
START:  SETC    INTM            ; Disable interrupts

            ldp     #0
            SPLK    #020h,TCR          ; T
            SPLK    #02ah,PRD          ; for AIC master clock (max:20h & 2ah)

            OPL     #0834h,PMST
            LACC    #0
            SAMM    CWSR            ; Set software wait state to 0
            SAMM    PDWSR            ;

            ldp #0
            CLRC    OVM             ; OVM = 0
            SPM     0               ; PM = 0

            ldp #0
            ;call hier1
            ldp #0
            SPLK    #0ch,IMR        ; useing tint as interrupt
            lacl #04h ; AICinit-mode
            samm AR1
            call AICINIT
            setc intm
            call calc

            ldp #0
           lacl #04h
           samm IMR
           lamm IFR
           samm IFR
           lacl #08h
           samm 51h
```

```
lacl #0
samm PSPx0

ldp #0
splk #03fffh,dbmr
lar AR0,#03ff0h;10h; z-count
lar AR1,#8003h; read_I mode
lar AR2,#02000h; x-count
lar AR3,#02000h; y-count
lar AR4,#02000h; x-slope:02000h=incr,0fdffh=decr
lar AR5,#02000h; y-slope:02000h=incr,0fdffh=decr
lar AR6,#rings; walker-cycle
ldp #xstep
lar AR7,#xstep

lacc #03fffh;#0
ldp #offset
sacl richtung
sfr
sacl offset

ldp #offi
mar *,ar2
lamm ar2
xor ar4
samm DASIx
lamm ar3
xor ar5
samm DASIy

CLRC   INTM          ; enable
lamm 51h
bsar 8
sacl offi
lacc #01ff0h
samm DASIv
lamm ar0
samm DASIz

lacl #0
samm tdxr
samm PSPx0
clrc sxm
```

The Include File "main.h":

```
;; Main.h  : establishes comunication to PC

rpt 15
 lamm 50h
lacl #0
samm 0ah;  preset Status C to "stop before setpoint"
lacl #0
samm 09h; preset walker movement as stop point;
lacl #01
samm 1fh

;0ah=statusC (command); AR1 is recent status; 09h is StatusB (used for
; second transmission and for compare register: "run until AR1=StatusB")
;1fh is counter for walker steps


wait:
```

```
    lamm 5ah
    and #1,11
    bcnd notempty,neq
    lamm AR1;        Status
    sub #1,15;
    bcnd wait,lt;    Wait for current Job to be over (over if <0)

    Setc INTM
    lamm 0ah;        StatusC
    bcnd wait,eq;    =0? => stop
    sub #8;
    bcnd noncont,lt; StatusC<8? =>non-continous mode
    sub #1
    bcnd cont,neq ; JMP for not 9.  C=9 is "continue until ar1=StatusB
     lamm AR1;+;
     xor #1,15;+;
     samm AR1;       Status
    and #0fh
    ldp #0;
    sub 09h; Compare to Status B, which is "run until AR1=StatusB"
    bcnd nc2,neq
     lamm AR1
     or #1,15
     samm AR1; Pause (Bit 15 of AR1 =1)
     lacl #0
     samm 0ah; Pause (C=0)
     b wait
cont:
     lamm AR1;+;
     xor #1,15;+;
     samm AR1;       Status
     b nc2
noncont:
    add #2; check for c=6
    bcnd nc2,neq
     lamm 1fh ; counter
     sub #1   ; decrease
     bcnd nc11,eq ; jmp out if it was 1 (keep it at value '1')
     samm 1fh ; save decreased counter
     call walk
     b nc3

nc11:
     lamm 09h
     and #1
     bcnd nc3,neq
     lacc #0111h
;;     samm DASIv
     b nc3

nc2: lacc #3fffh
    ldp #offset
    sacl richtung
    sfr
    sacl offset
    ldp #0
nc3: lamm 51h
    clrc INTM
    b wait;          and loop

notempty:
     lamm 50h;       Read from PC
     and #0fh,0
```

```
    samm 0ah;        StatusC
     lacl #081h; 129 is my convention for "first instruction received"
     samm 50h
     lamm 0ah
     samm 50h
    lamm AR1
    or #1,15
    samm AR1 ;        Stops current Process

   call getnext
   samm 09h;        StatusB
   bcnd nzero,neq
   lamm 0ah;
   sub #5
   bcnd nzero,eq; only if c=5, then second transmission=0 permittet!
   lacl #0
   samm 0ah ; StatusA: if second transmition is zero, always stop the system

nzero: lamm 0ah ;StatusC
   bcnd send,eq; if c=0 (Pause)
   sub #07
   bcnd single,lt;;; needsd to set AR1
   bcnd conti,gt      ;; continuous-procedure for c>=8
   lamm AR1;+;
   xor #1,15;+;
   samm AR1;        Status
   samm 09h;        StatusB
   b send
conti:
   lamm AR1;+;
   and #7fffh;++;
   samm AR1;        Status
   b send
single: ; singlestep procedure for c<7
   add #2
   bcnd count,eq ; set counter for walker if c=5
   add #1
   bcnd setvalue,eq; set value #B if c=4
   ;;; from now on c=6 or c=1,2 or 3
    lamm 09h; load second transmission value
    sub #1; walk left
    bcnd walk_l,eq
    sub #1; walk forward
    bcnd walk_f,eq
    sub #1; walk right
    bcnd walk_r,eq
    sub #1; walk backward
    bcnd walk_b,eq
    sub #1; contract z
    bcnd reject,eq
    sub #1;; readl(expandz)
    bcnd expand,eq
    sub #1;; scanx
    bcnd send,neq
            lacc #01fffh
            samm DASIv
            lacl #5;+; scanx mode
            samm AR1; pause mode
            lar AR7,#xstep
            ldp #0
            splk #03fffh,dbmr
            clrc sxm
            lacl #0
            samm PSPx0
```

```
            mar *,ar2
    b send

reject: lamm ar0
            ldp #n03fefh
            sub n03fefh
            ldp #0
            bcnd send, geq

            lacl #02h
            samm 51h
            ldp #0
            SPLK   #020h,TCR        ; T
            SPLK   #03ah,PRD        ; for AIC master clock (max:20h & 2ah)
            ldp #Ihi
            lacl #01h;+; contract-z-mode
            samm AR1; pause mode
            b send

count:  ;(c=5)
    lamm 09h
    samm 1fh
    call get12bit
    b send
setvalue:
    lacc #7eh
    samm 50h
    lamm 09h
    samm 50h
    sub #1
    sfl
    sfl
    sacb ; and store #B in ACCB
     call getnext
     samm 1fh
     call get12bit
      call getPC
      clrc C;+1
      adcb ;+1; PC points on this instruction
      add #0fh ;+1; this has to point on num0!!!!!!
      cala ;+1
     lacb  ;+1; ACCB is (#B-1)*4. B=1 and B=2 are for Walker=>ACCB=0 & 4
     sub #4;+1
     cc calc,leq;+2; re-calculate walker cycle
     lacl #1;+1
     samm 1fh;+1
     b send ;+2
getPC:  pop  ;+1
            push ;+1
     ret    ;+1

num1:
            ldp #y1
            lamm 1fh
            sacl y1
            ret
num2:
            ldp #cent
            lamm 1fh
            sacl cent
            ret
num3:
            ldp #Ihi
            lamm 1fh
```

```
        sacl Ihi
        ret
num4:
        ldp #delta_I
        lamm 1fh
        sacl delta_I
        ret
num5:
        ldp #Irefhi
        lamm 1fh
        sacl Irefhi
        ret
num6:
        ldp #dIref
        lamm 1fh
        sacl dIref
        ret
num7:
        ldp #xstep
        lamm 1fh
        sacl xstep
        ret
num8:
        ldp #ystep
        lamm 1fh
        sacl ystep
        ret
num9:
        lamm 1fh
        nop
        samm DASIv;; no xor. Thus 0=0 and +=max and -=min
        ret
numa:
        lamm 1fh
        samm AR2;;;x
        samm DASIx;;; wrong, since xor is missing
        ret
numb:
        lamm 1fh
        samm AR3;;;;y
        samm DASIy   ;;; wrong, since xor is missing
        ret
numc:
        ldp #xstep
        lamm 1fh
        sacl xstep
        ret
numd:
        ldp #xstep
        lamm 1fh
        sacl xstep
        ret
nume:
        ldp #xstep
        lamm 1fh
        sacl xstep
        ret
numf:
        lamm 1fh
        samm AR0;;;z pos : BE CAREFULL: Z-STEP WON'T BE SLOWED DOWN!!
        ldp #n02000h
        xor n02000h
        samm DASIz
        samm DASIz
```

```
            samm DASIz
            samm DASIz
            ret


send:
                        ;(B and 128) will tell PC, whether B>0 or B<0
                        ;(C and 192) will tell PC, whether C <,=,>0
                        ;          =0 =>c=0, =192 => c<0, =64 =>c>0
   lacc #0ffh
   samm 50h
   samm 50h
   lamm 0ah;        StatusC
   samm 50h;
   bcnd null,eq
   sub #8
   bcnd null,eq
   add #1
   bcnd null,eq
    lamm 09h;       StatusB
    samm 50h
    lamm 0bh;       StatusD
    b notnull
null:  lamm ar0
     samm 50h
     bsar 8
notnull:
   samm 50h;
   lamm AR1;        Status
   samm 50h

   sub #1
   bcnd nn2,eq
   sub #2
   bcnd nn1,neq
    lamm ar0;; if next step = Read_i then check if z not already at zmin
    sub #11
    bcnd nn1,gt
nn0:   lamm ar1
     or #1,15
     samm ar1
nn1: lamm 51h
   clrc INTM
   b wait;          and loop

nn2: lamm ar0
   ldp #n03fefh
   sub n03fefh
   bcnd nn0,geq
   b nn1

walk_l: b walk_f
walk_r: b walk_b
walk_f: lacc #3fffh
            ldp #offset
            sacl richtung
            sfr
            sacl offset
            b walk
walk_b: lacl #0
            ldp #offset
            sacl offset
            sacl richtung
```

```
walk:
        lacl #00h
        samm 51h
        lacl #0ch;#08;#0ch
        samm IMR
        lamm IFR
        samm IFR
        ldp #0
        SPLK   #020h,TCR        ; T
        SPLK   #02ah,PRD        ; for AIC master clock (max:20h & 2ah)
        lacl #02;;; lamm 09h
        samm AR1
        mar *,AR6
        lacl *
        samm DASIv
        clrc intm
        b send

getnext:
    lamm 5ah
    and #1,11
    bcnd getnext,eq;; "wait for next value" MUST be in here!!!
    lamm 50h;       Read from PC
    and #0fh,0
    ret

get12bit: ;(first 4 bits need to be in 1fh)
    call getnext
    samm 09h;       StatusB
    lamm 1fh
    ldp #0
    add 09h,4
    samm 1fh
    samm 09h

    call getnext
    samm 0bh;       StatusD
    lamm 1fh
    ldp #0
    add 0bh,8
    samm 1fh

    call getnext
    samm 0bh;       StatusD
    lamm 1fh
    ldp #0
    add 0bh,12
    samm 1fh
    bsar 8
    samm 0bh
    ret

expand:
        lacl #1
        samm 1fh
        lacl #04h
        samm IMR
        lamm IFR
        samm IFR
        lacl #08h
        samm 51h
        lacl #03h;+; rd-I-mode
        samm ar1;
        lamm ar0;; if next step = Read_i then check if z not already at zmin
```

```
        sub #11
        bcnd send,gt
        lamm ar1
        or #1,15
        samm ar1
    b send
```

```
; idea: PC writes to StatusB
;     Interupt-Procedures write to StatusD
;     StatusC will determine the mode: 0=Stop (wait for PC)
;                                >=8 continous (StatusD --> Status)
;                                <8 PC-driven (StatusB --> Status)
;                                      (0 (wait)--> StatusB)
; Status = AR1; StatusB,C,D are 09h,0ah,0bh
; Requirements for StatusB and D: bits 6 and 7 must be 0
```

The Include File "scannc.h":

```
;scan: scans over whole range
; fullfill all prerequisits before!!!!

; 01ch summ register

office4:;;1=contrz,2=walk,3=readI,4=setAIC,5=scanX,6=EndOfImage,
        ;;7=scanX-OutOfRange,5+8=scanY,7+8=scanY-OutOfRang
        lamm 51h
        lamm AR1
        and #08007h
        sub #1
        bcnd contract_z,eq
        sub #2
        bcnd read_I,eq
        ;and #0dh
        sub #2
        bcnd scan,eq
        sub #2
        bcnd scan,eq
        rete

scan:
  lamm PSPx0
  ;; make summ her: a*oldvalue+b new values (save it in 01ch)
  ldp #0
  xor #08000h
  sacb
  add 01ch,2
  add 01ch
  bsar 2
  samm 1ch
  lacb
  ldp #Irefhi
  sub Irefhi
  bcnd zdcc,gt
  add dIref
  bcnd zinc,lt
scan2:
  lacl #0
  samm 08h;;flag(=1 => dec, =2 => inc)
```

```
 ldp #0
 splk #0,arcr
 bit ar1,0ch
 bcnd scanx,ntc
scany:
  mar *,ar3
  mar *+
  bit ar3,1
  mar *-
  xc 2,tc
   xpl ar3
   xpl ar5
  mar *+,ar7
  lamm ar3
  xor ar5
  samm DASIy
  cmpr 0
  mar *-,ar3
  xc 1, ntc
   rete
  lar ar1,#8005h
   ldp #xstep
   lacl xstep
   sfr ; since indx=2
   samm ar7
  mar *,ar2
  b send_z
scanx:
 mar *+,ar7
 lamm ar2
 xor ar4
 samm DASIx
 cmpr 0
 bcndd scanx1,ntc
  mar *-,ar2
  mar *+
   ldp #xstep
   lacl xstep
   samm ar7
   ldp #0
;warten:
;  lamm 5ah
;  and #1,8
;  bcnd warten,eq
  lamm ar0
  sfl
  sfl
  and #0fffch
  samm 50h
  bsar 8
  samm 50h
scanx1:
 bit ar2,1
 mar *-
 xc 1,ntc
  rete
 xpl ar2
 xpl ar4
 lar ar1,#800dh
   ldp #ystep
   lacl ystep
   sfr ; since indx = 2
   samm ar7
  mar *,ar3
```

```
send_z:
;warten2:
;  lamm 5ah
;  and #1,8
;  bcnd warten2,eq
 lamm ar0
 and #0fffch
 samm 50h
 bsar 8
 samm 50h
 ;;lamm ar0; the summ is 01ch
 ;;and #7fh
 ;;samm 50h
 ;;lamm ar0; the summ is 01ch
 ;;bsar 7
 ;;or #80h
 ;;samm 50h
 ;lamm 01
 ;and #07fh
 ;samm 50h
 ;lamm 01
 ;bsar 7
 ;or #80h
 ;samm 50h
 ;;lacl #0
 ;;samm 01ch
 ;samm 01
 rete

zdec:
        ;;lacl #0
        ;;samm PSPx0
        lamm 08h
        sub #2;  already inc?
        bcnd scan2,eq
        lacl #1 ; set dec mode
        samm 08h

        lamm ar0
        sub #1
        samm ar0
        xor #02000h
        samm DASIz
        lamm ar0
        sub #2
        bcnd zdec2,gt
        ; OUT OF RANGE
         lamm ar0
         add #1
OOR:    samm ar0
        rete

         lamm ar1
         or #02h
         samm ar1
         or #0f
         ldp #0
         sub 09h  ;; compare to Stutus B
         bcnd scan2,neq
         lamm 0ah ;statusc
         sub #09;; to check for "run until"
         bcnd scan2,neq;; stop only if "run until OOR"
         rete
zdec2:
```

```
        ;lamm ar1
        ;and #0dh ;=f-2
        ;samm ar1
        rete

zinc:
        lamm 08h
        sub #1
        bcnd scan2,eq
        lacl #2
        samm 08h
        ;;lacl #0
        ;;samm PSPx0
        lamm ar0
        add #1
        samm ar0
        xor #02000h
        samm DASIz
        lamm ar0
        sub #03ffeh,0
        bcnd zdec2,lt
        ; OUT OF RANGE
         lamm ar0
         sub #1
         ;b OOR
        samm ar0
        rete
```

The Include File "cont_zmc.h":

```
;cont_zmc:(m for minus)(c for controllable)  drives piezo in home position
; for th case that some trouble occures: I changed line 17 from 0c to 08
contract_z:
        lamm ar0
        add #10h;sub #20h      ;;;;;;20!!!
        samm ar0
    ldp #n02000h
        xor n02000h
        samm DASIz
        lacl #0
        samm PSPx0
        lamm ar0
        sub n03fefh; #11h
        bcnd conz1,lt;gt

    lacc #03fefh
    samm AR0

         lacl #00h
         samm 51h
         lacl #08h;0ch
         samm IMR
         lamm IFR
         samm IFR
         ;-;lamm AR1
         lacl #02h;+; walker-mode
         or #1,15
         samm AR1; pause mode
         ;-;lacl #02h; walker-mode
         ;-;samm 0bh; to StatusD
         ldp #0
         SPLK   #020h,TCR        ; T
         SPLK   #02ah,PRD        ; for AIC master clock (max:20h & 2ah)
         ldp #Ihi
```

```
        reti
conz1:  rete
```

The Include File "read_ipc.h":

```
;read_I: expands piezo as long as I within the limits (= no tunneling)

read_I:
        lamm PSPx0
        ldp #Ihi
        xor n08000h
        subs Ihi
        bcnd rdI2,gt
        adds delta_I
        bcnd rdI2,lt

        lamm AR0        ; z-count
        sub #10h;add #10h
        samm AR0
rdI0:
        ldp #Ihi

        xor n02000h
        samm DASIz
        lamm AR0
        sub #11h;n03fefh
        bcnd rdI1,gt;lt
    lacl #10h
    samm AR0
        lacl #02h
        samm 51h
        ldp #0
        SPLK    #020h,TCR       ; T
        SPLK    #03ah,PRD       ; for AIC master clock (max:20h & 2ah)
        ldp #Ihi
        lacl #01h;+; contract-z-mode
        or #1,15
        samm AR1; pause mode
        reti
rdI1:   rete
rdI2:
        lacl #5;+; scanx mode
        or #1,15
        samm AR1; pause mode
        lar AR7,#xstep
        ldp #0
        splk #03fffh,dbmr
        clrc sxm
        lacl #0
        samm PSPx0
        mar *,ar2
    reti
```

The Include File "walkerc.h":

```
;Walker: puts out one walker cycloid

walker:
    lamm ar1
    and #1,15;bsar 15
    retc neq
        mar *,AR6
```

```
        lacl *+
        ldp #richtung
        xor richtung
    add offset;;++
        samm DASIv
        lacl #0
        samm PSPx0
        cmpr 0
        bcnd walk1,NTC
        lamm 1fh
        sub #1
        samm 1fh
        bcnd walk1,neq
        lacl #1
        samm 1fh
        lacl #04h
        samm IMR
        lamm IFR
        samm IFR
        lacl #08h
        samm 51h
        lacl #03h;+; rd-I-mode
        or #1,15
        samm ar1; pause mode
        lacl #0
        samm PSPx0
        reti
walk1:  rete
```

The Include File "pspinit.h":

```
;PSPinit: sets PSP2 to values given in VAR01.prt
* aic initialization data
*
AICINIT:
home:
    lacl #80h
    samm 51h
    ldp #0
    SPLK   #0ffh,IFR

    ;------------------------
    SETC   SXM          ;
    clrc intm   ; enable interrupts
    lamm 51h

    lacl #0h
    samm 52h
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    samm 52h
```

```
        idle
        ;-------------------------
        LDP    #a              ;
        LACl   a
        add    aa,8
        sacl   ta
        CALL   AIC_2ND
        ;-------------------------
        LACl   b
        add    bb,8
        sacl   ta
        CALL   AIC_2ND
        ;-------------------------
        LACl   c
        add    cc,8
        sacl   ta
        CALL   AIC_2ND
        ;-------------------------
        LACl   d
        add    dd,8
        sacl   ta
        CALL   AIC_2ND

        RET

AIC_2ND:

        clrc intm   ; enable interrupts
        lamm 51h
        lacl #03h
        samm 52h
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        lacl  ta
        samm 52h
        idle
       RET
```

The Include File "calc0f3n.h":

```
;calc0f073: calculates Walker-cycloid in "#0fh,03h"-steps

calc:
            ldp #0
            splk #rings,cbsr1
            ldp #ccnt
            lacc ccnt,1
            sub #2
            add #rings,0
            samm cber1
```

```
        samm arcr
        ldp #0
        mar *,AR6
        lar AR6,#rings
        splk #0eh,cbcr
        lacl #0
        ldp #coeff
        sacl coeff
        setc sxm

fgh:
        LDP    #SIN_SW
        bit    SIN_SW,15
        BCND  SUBTRC,TC
        ldp #coeff
        lacl coeff
        add #1
        sacl coeff
        ldp #ccnt
        sub ccnt;    ;*;*;*;*;*;* 20 for 800
        ldp #coeff
        bcnd REC2,LT
        ldp #SIN_SW
        lacl #15
        sacl SIN_SW
        b REC2
SUBTRC:
        ldp #coeff
        lacl coeff
        sub #1
        sacl coeff
        bcnd REC2,GT

        lacl #0
        ldp #SIN_SW
        sacl SIN_SW
        mar *,AR6
        lar AR6,#rings
        ret

REC2:
        clrc SXM
        mpy #0
        lacl #0
        ldp #y1
        Macd coeff,y1
        apac
        sfr          ;==>Max 15 Bit long y1

        ldp #y2
        sacl y2
        sqra y2
        lacl #0
        apac
        sach y2h
        sacl y2l
        lacc #8,15
        rpt  #11
        sfl
        sub #1       ; to get something like 7fffh ...
        sub y2h,16
        sub y2l      ;  (1-(p/pmax)^2), WHERE 1==1*2^30
```

```
            call sqroot ;  (sqroot of ACC, where ACChi = vorkommastellen und
            acclo=nachkommastellen)

            setc SXM
            neg
            add #1fffh,2
            and #0fffch
            sfr;+;;++
            sfr
    ;;--xor  #02h,0ch;; don't turn walker direction _/\_
    ;;--add #1,3
            add #1,2 ;+;;++

            sacl *+
      b fgh


sqroot:
            Bcnd loo4,EQ
            sacl y2l
            sach y2h
            ldp #y3
            lacc #1,15
            sacb
            CLRC SXM
loo1:   sacl y3
            sqra y3
            lacc y2h,16
            add y2l
            setc SXM
            neg
            apac
            clrc SXM
            Bcnd loo2,NC
            lacl y3
            sbb
            sacl y3
loo2:   lacb
            sfr
            Bcnd loo3,C
            sacb
            lacl y3
            adcb
            B loo1
loo3:   lacl y3
loo4:   ret
```

BIBLIOGRAPHY

1) R. M. Feenstra, Physical Review B **50**, 4561 (1994).

2) R. M. Feenstra et al., in Proceedings of the Conference on Physical Properties of Semiconductor Interfaces at Sub-Nanometer Scale, 1992, edited by H. W. M. Salemink (Kluwer, Dordrecht, to be published)

3) P. Horowitz and W. Hill, The Art of Electronics (Cambridge Univ. Press, New York, 1982) p. 288

4) Texas Instruments, TMS320C5x User's Guide, (1993)

5) Texas Instruments, TMS320C5x DSP Starter Kit User's Guide, (1994)

VITA

Michael Tasler was born in Freiburg im Breissgau, Germany on March 31, 1972, the son of Bernd-Rüdiger Tasler and Renate Tasler. After moving to Essen, Moers, Munich and Frankfurt he completed his work at the German high school Hanns-Seidel-Gymnasium at Hösbach, Germany in July 1991. He entered the University of Würzburg, Germany in November 1991. In September 1993 he received his "Vordiplom" in physics.
In September, 1994, he entered the Graduate School of The University of Texas at Austin.

Permanent Address: Schlossberg 2
63773 Goldbach
Germany

Tel.: 01149-602-154-0259
e-mail: MiT@PHYSICS.UTEXAS.EDU

Publications:

This work will be published at the "Texas Instruments 1995 DSP Solution Challenge" contest and will be submitted to Texas Instruments Incorporated before December 31, 1995.

This thesis was typed by the author.

BIBLIOGRAPHY

1) R. M. Feenstra, Physical Review B **50**, 4561 (1994).

2) R. M. Feenstra et al., in Proceedings of the Conference on Physical Properties of Semiconductor Interfaces at Sub-Nanometer Scale, 1992, edited by H. W. M. Salemink (Kluwer, Dordrecht, to be published)

3) P. Horowitz and W. Hill, The Art of Electronics (Cambridge Univ. Press, New York, 1982) p. 288

4) Texas Instruments, TMS320C5x User's Guide, (1993)

5) Texas Instruments, TMS320C5x DSP Starter Kit User's Guide, (1994)

VITA

Michael Tasler was born in Freiburg im Breissgau, Germany on March 31, 1972, the son of Bernd-Rüdiger Tasler and Renate Tasler. After moving to Essen, Moers, Munich and Frankfurt he completed his work at the German high school Hanns-Seidel-Gymnasium at Hösbach, Germany in July 1991. He entered the University of Würzburg, Germany in November 1991. In September 1993 he received his "Vordiplom" in physics.
In September, 1994, he entered the Graduate School of The University of Texas at Austin.

Permanent Address: Schlossberg 2
63773 Goldbach
Germany

Tel.: 01149-602-154-0259
e-mail: MiT@PHYSICS.UTEXAS.EDU

Publications:

This work will be published at the "Texas Instruments 1995 DSP Solution Challenge" contest and will be submitted to Texas Instruments Incorporated before December 31, 1995.

This thesis was typed by the author.

BIBLIOGRAPHY

1) R. M. Feenstra, Physical Review B **50**, 4561 (1994).

2) R. M. Feenstra et al., in Proceedings of the Conference on Physical Properties of Semiconductor Interfaces at Sub-Nanometer Scale, 1992, edited by H. W. M. Salemink (Kluwer, Dordrecht, to be published)

3) P. Horowitz and W. Hill, The Art of Electronics (Cambridge Univ. Press, New York, 1982) p. 288

4) Texas Instruments, TMS320C5x User's Guide, (1993)

5) Texas Instruments, TMS320C5x DSP Starter Kit User's Guide, (1994)

VITA

Michael Tasler was born in Freiburg im Breissgau, Germany on March 31, 1972, the son of Bernd-Rüdiger Tasler and Renate Tasler. After moving to Essen, Moers, Munich and Frankfurt he completed his work at the German high school Hanns-Seidel-Gymnasium at Hösbach, Germany in July 1991. He entered the University of Würzburg, Germany in November 1991. In September 1993 he received his "Vordiplom" in physics.
In September, 1994, he entered the Graduate School of The University of Texas at Austin.

Permanent Address: Schlossberg 2
63773 Goldbach
Germany

Tel.: 01149-602-154-0259
e-mail: MiT@PHYSICS.UTEXAS.EDU

Publications:

This work will be published at the "Texas Instruments 1995 DSP Solution Challenge" contest and will be submitted to Texas Instruments Incorporated before December 31, 1995.

This thesis was typed by the author.

BIBLIOGRAPHY

1) R. M. Feenstra, Physical Review B **50**, 4561 (1994).

2) R. M. Feenstra et al., in Proceedings of the Conference on Physical Properties of Semiconductor Interfaces at Sub-Nanometer Scale, 1992, edited by H. W. M. Salemink (Kluwer, Dordrecht, to be published)

3) P. Horowitz and W. Hill, The Art of Electronics (Cambridge Univ. Press, New York, 1982) p. 288

4) Texas Instruments, TMS320C5x User's Guide, (1993)

5) Texas Instruments, TMS320C5x DSP Starter Kit User's Guide, (1994)

VITA

Michael Tasler was born in Freiburg im Breissgau, Germany on March 31, 1972, the son of Bernd-Rüdiger Tasler and Renate Tasler. After moving to Essen, Moers, Munich and Frankfurt he completed his work at the German high school Hanns-Seidel-Gymnasium at Hösbach, Germany in July 1991. He entered the University of Würzburg, Germany in November 1991. In September 1993 he received his "Vordiplom" in physics.
In September, 1994, he entered the Graduate School of The University of Texas at Austin.

Permanent Address: Schlossberg 2
63773 Goldbach
Germany

Tel.: 01149-602-154-0259
e-mail: MiT@PHYSICS.UTEXAS.EDU

Publications:

This work will be published at the "Texas Instruments 1995 DSP Solution Challenge" contest and will be submitted to Texas Instruments Incorporated before December 31, 1995.

This thesis was typed by the author.

ACKNOWLEDGMENTS

I would like to first acknowledge my advisor Prof. Ken Shih. Thanks to his financial support it was possible for me to finish my studies at the University of Texas at Austin.

I greatly appreciate the tremendous help I received from Prof. Max Scheer and Prof. Manfred Fink, both of whom invested their time in the German partnership program.

I also would like to thank Prof. Keto for his insightful suggestions on various topics.

I also received important support from Texas Instruments Inc., Analog Devices and Motorola, who sponsored my work with sample ICs.

Last but definitely not least I want to acknowledge the enormous help I received during countless conversations with my friends in Dr. Shih's lab and Dr. Martinez' lab. Especially Günter Guttroff who spent quite some time listening to my plans. Gordon Chao and Art Smith were always there to answer my questions and introduce me to the sub-nanometer world.

Ricardo Claps, Andrew McDonald, Michael Barrett and Doris Heinrich turned out to be the source of new motivation to keep me working on my project. I greatly appreciate the time we talked during the breaks we took together.

ABSTRACT

Design and Construction of a Universal Data Acquisition and Control System for Scanning Probe Microscopy

by

Michael Tasler, M.A.

The University of Texas at Austin, 1996
SUPERVISOR: Chih-Kang Shih

Scanning Probe Microscopy (SPM) has been used to study many novel material properties such as single electronic energy levels formed in a quantum corral system. However, electronic states of more realistic structures like quantum wells, which are of interest in optoelectronic applications, have not yet been fully explored using SPM. While this is possible in principle, it requires a more sophisticated SPM control unit which is not available commercially. The development of such a system is the aim of this project. General ideas and problems are discussed.

TABLE OF CONTENTS

ACKNOWLEDGMENTS ..... **iv**

ABSTRACT ..... **v**

Chapter 1: Introduction **1**

(1.1) Motivation: 1

(1.2) General SPM Setup: 2

(1.2.1) General Mechanical SPM Design: 2

(1.2.2) Electrical SPM Setup in the Case of Tunneling Microscopy: 4

Chapter 2: Design of the SPM Control Unit **8**

(2.1) Theoretical Discussion of the System Requirements and Limits: 8

(2.1.1) Electronic Noise: 8

(2.1.1.1) Shot Noise: 8

(2.1.1.2) Johnson Noise: 9

(2.1.1.3) Flicker Noise: 9

(2.1.1.4) Noise of a JFET Preamplifier: 9

(2.1.2) Required accuracy: 10

(2.1.2.1) Accuracy of the Tip Positioning: 10

(2.1.2.2) Required Accuracy of the Bias Voltage: 11

(2.2) Practical Realizations: 12

(2.2.1) Timing Requirements for Different Scanning Modes: 13

(2.2.2) The Digital Control Unit: 14

(2.3) The Realization of an Independent Processor System: 14

(2.4) Concluding Comments on System Limits: 16

Chapter 3: Hardware Design **17**
(3.1) The DSP Motherboard: 18
(3.1.1) The Interrupt Manager (IM): 18
(3.1.2) The Connection to the Master PC: 19
(3.2) The TLC320AC01 Serial Port: 23
(3.4) The Parallel Digital I/O Port: 24
(3.5) The Analog Boards: 24

Chapter 4: The Final Assembly **26**

Chapter 5: The Software **29**
(5.1) TI's Assembler: 29
(5.2) The DSK Programmer Tool: 30
(5.3) Dynamic Range Expansion: 33

Chapter 6: Results and Conclusions **37**

APPENDIX **38**
Appendix A: Block Diagram of SPM Control Unit 39
Appendix B: Schematics 40
Appendix C: PCB Layouts 42
Appendix D: The PC Software 57
Appendix E: The DSP Software 68

BIBLIOGRAPHY **86**
VITA **87**