Accurate positioning for nonintrusive near-field optical microscopy

Diploma paper by

Daniel Akenine

LRAP-187
Lund, November 1995

Abstract

In non-intrusive scanning near-field optical microscopy (SNOM) an optically trapped microscopic light source is accurately positioned with respect to the studied sample. This diploma paper describes the development of a windows-based computer interface for the piezoelectric x-y scanning stage. The resolution, speed and linearity of the computer/stage system is investigated. Finally, 200 nm fluorescent test objects are constructed and the system is used as an integrated part in non-intrusive SNOM test measurements.

Table of contents

1. Introduction

This introduction is an overview of different microscopical techniques focusing especially on the Scanned Near-field Optical Microscope and its design. Also a non-intrusive scanned near-field microscope will be described.

1.1 Overview

The quest for finding a way to make the smallest things in nature visible to the eye tests the malleability of the laws of nature.

Since Abbe in 1886[12] constructed a high resolution microscope only minor progress in optical imaging has been made. Development of this traditional design has mostly focused on new types of optical lenses and on minimising glass deficiency and surface imperfections. The limitation is no longer technical but rather due to the physical nature of light. Since light can be described as an electromagnetic wave the diffraction limits the resolution R to[10]

$$R \approx 0.61\lambda \ , \qquad (1)$$

where λ is the wavelength, making the resolution to be a couple of hundred nm for visible light. New techniques must therefore be developed to improve resolution.

Since the resolution is dependent on the wavelength of the electromagnetic wave a natural approach is to use radiation with shorter wavelengths, e.g. X-rays. X-ray microscopy has shown resolutions 5-10 times better than visible-light microscopy[1]. The limiting effect is in this case not diffraction but rather the difficulty to construct lens elements that can focus X-rays well . However, a negative aspect of X-rays is that the high energy they carry per quanta could cause radiation damage in the studied object.

Another approach is to use medium or high energy electrons which have a very short associated de Broglie wavelength, thus making them ideal for very high resolution imaging. Electron microscopy, first described in 1932[12] has since then reached atomic resolutions. The drawback with electron microscopy is the complicated process for studying biological materials. They have to be dehydrated, stained, fixed, sectioned and placed in vacuum, thus making it difficult to study living objects.

Different scanning probe techniques which scans very small probes over the material have also been developed. An example of this technique is the Atomic Force Microscope (AFM) which uses the forces between atoms to build an image of the studied surface. The Scanning Tunnelling Microscope (STM), that use the quantum mechanical effect of electron tunnelling to measure the distance between the probe and the surface has also shown good results. Another possibility is to use a technique called Scanned Near-field Optical Microscopy (SNOM), which is discussed in the next section. The problem with scanning-probe techniques is that they set restrictions on the studied objects properties making some materials more suitable than others.

1.2 Intrusive and non-intrusive scanned near-field optical microscope

One way to exceed the resolution of the ordinary optical microscope is the Scanned Near-field Optical Microscope (SNOM).

It was first suggested in the late 1920:s[11] but has only for the last two decades been more in focus for researchers. The basic idea is that light, which is transmitted through a small aperture will act as a lightsource with dimensions limited by the size of the aperture rather than the wavelength. If one makes this aperture smaller than the wavelength of the light it will create a very small "spotlight" (say 50 nm diameter) that can be scanned over a surface. The transmitted or reflected light can then be collected by a detector and give information of the studied material.

A problem is that the aperture will act as a small point-source and emit very divergent light due to diffraction. This makes it necessary to locate the aperture in close proximity to the studied object. Scanning that close can create additional problems with interactive forces between the atoms or molecules. A disadvantage for an ordinary SNOM is that the probe requires mechanical access to the object making it difficult to image e.g. living biological material with intervening membranes without destroying it. In an attempt to solve these problems the so called Trapped Particle Optical Microscope (TPOM) has been developed[1,2]. It uses a very small particle (≈ 50 nm) that is optically trapped at the focus of a laser beam (Fig 1.1). This makes the particle act as a lightsource with no mechanical connection to the environment. The optical trapping is created by the refractive and the reflected forces of the light when it interacts with the particle[1]. The trapped particle may now be positioned e.g. behind membranes without damaging the studied object. To be able to separate the light that is emitted by the particle from the light in the laser beam, a particle of lithium niobate is used. This non-linear crystal particle frequency doubles the light passing through it, if the intensity is strong enough[10]. This effect makes it possible to filter out the trapping radiation from the photomultiplier.

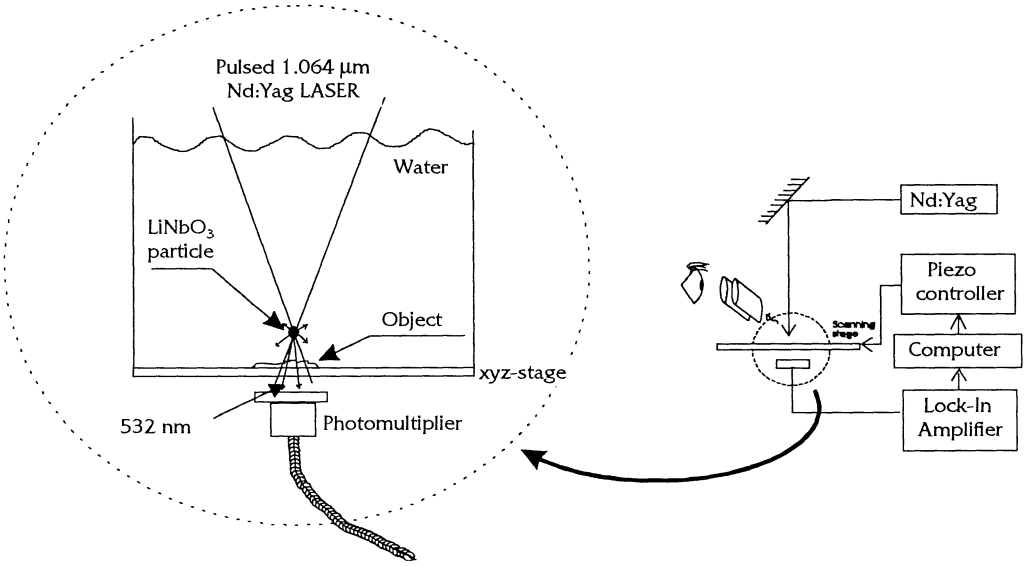


Fig 1.1 The figure shows the experimental arrangement for a TPOM microscope.

2 Scanning properties

2.1 Theoretical scanning limits

To be able to make an automatic TPOM scan one has to consider built-in limitations in the design.The stability of the trap has to be considered since the trapping is sensitive and must not be lost during scanning. An investigation of the maximum scanning speed and resolution must therefore consider the trapping force as well as the movement inside the trap. The movement of the particle in the trap is influenced by the radiation pressure from the laser and on the Brownian motion which will set the maximum resolution. As an example the root mean square displacement ($\sqrt{<r^2>}$) in the trap using a cw 100 mW, λ=1.06 μm focused to FWHM (Full Width Half Maximum) = 450 nm is approximately[3] 25 nm. Also the rms in the z-direction is important as it sets a limit how close the particle will be able to scan over the surface. Typically the rms displacement in the z-direction is ≈ 50 nm assuming the same conditions as above[3].

Another limiting factor is attractive van der Waals forces which are trying to adsorb the particle on the studied surface and are competing with the trapping forces in the laser beam. If the particle comes to close to the object the trapping will be lost and it will be adsorbed by the surface. The safe distance for this is normally in the range of 250 nm but varies with the test object and the surrounding media[4]. However, by repulsive short-range electrostatic forces this distance can be reduced 10 times.

Considering that the particle has a lifetime of 3-30 minutes in the trap a fast scan is desirable. However, as the particle is surrounded with water a fast scan could cause the trapping to fail. Therefore it's necessary to get an idea of how strong the trapping really is. A calculation of the trapping force can be done using the Rayleigh theory which may be used for particles with dimensions up to 20 % of λ giving an error < 2 %. For larger particles Mie theory should be applied. It is necessary to calculate both the strength of the gradient force as well as the scattering force of the particle since they both work together in the capturing. Using Rayleigh theory the gradient force (F_{grad}) is identical with the Lorentz force and may be written[1].

$$\overline{F}_{grad} = (\overline{p}\cdot\nabla)\overline{E} + \frac{\overline{1}}{c}\cdot\frac{d\overline{p}}{dt}\times\overline{B} \qquad (2)$$

where p is the dipole moment of the particle. Using vector analysis and Maxwell's equations the gradient force may be transformed to

$$\overline{F}_{grad} = \alpha(\nabla(\frac{E^2}{2}) + \frac{1}{c}\cdot\frac{1}{dt}(\overline{E}\times\overline{B})) \qquad (3)$$

where $\alpha = \frac{\overline{p}}{\overline{E}}$ is describing the polarizability and c the speed of light. Normally the second term is negligible resulting in

$$\overline{F}_{grad} = -\frac{n_s}{2}\cdot\alpha\nabla E^2 = -\frac{n_s^3 a^3}{2}\frac{(m^2-1)}{(m^2+2)}4\pi\varepsilon_0\nabla E^2 \qquad (4)$$

where m is $n_{p(article)}/n_{s(urrounding\ media)}$, a the radius of the particle and ε_0 the vacuum permeability. The opposite directed force can be calculated by looking at the scattered intensity at the particle and integrating it over a sphere (the particle) thus resulting in a total scattered power of

$$P_{sca} = \iiint_{sphere}\frac{16\pi^4 a^6(\varepsilon_1-\varepsilon_2)^2}{r^2\lambda^4(\varepsilon_1+2\varepsilon_2)}\sin^2\phi = \frac{128\pi^5 a^6(m^2-1)^2}{3\lambda^4(m^2+2)^2} \qquad (5)$$

where ϕ is dipole axis angle to the scattering direction, a the particle radius and ε_1 and ε_2 the dielectric constants of the particle and the medium, respectively.The intensity is asummed to be uniform.

The Rayleigh theory now says that this scattering power results in $F_{sca}=n_pP_{sca}/c$ which gives

$$F_{sca} = \frac{128\pi^5 a^6 n_p (m^2-1)^2}{3\lambda^4 c(m^2+2)^2} \cdot I_0 \quad (6)$$

At the equilibrium point you now have the resulting force F_{sca}-F_{grad} to be zero. By calculating the energy it takes to escape from the equilibrium one can get an idea of how willing the particle is to stay in the trap. Plotting the total force on the particle shows that a scan that creates forces > 0.1 pN could cause the trapping to fail[1]. This indicates that the viscous forces from the surrounding water on the particle when scanning should not be greater than that.

As a conclusion these discussions are indicating that the upper limit of resolution due to brownian motion is around 25 nm. A positioning resolution in that range should therefore be aimed for. Considering the maximum scan-speed experiments has shown[5] on stabile trapping at speeds around 100 μm/s using 200mW trapping power and 60 nm SiO_2. Such speeds are no problem with regard to the linear movement.

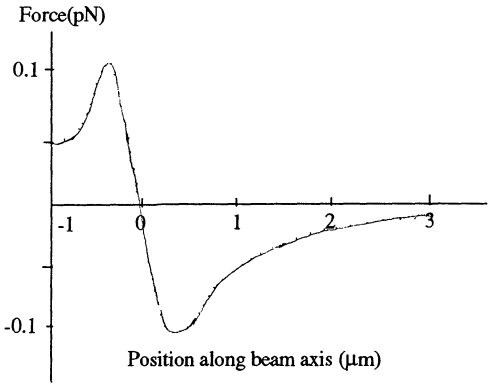
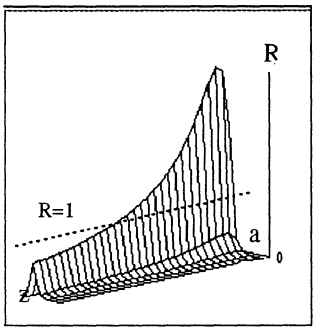


Fig 2.1 The total force acting on the trapped particle. The trapping point is where the total force is zero.

However, there could be accelerations exceeding that speed. This should not cause a problem if the movement in each step is small enough, say ≈100 nm. This is because the trap has a size of ≈ 0.5 μm causing the particle to fall back into the equilibrium. These results are in good agreement with the assumption that the viscous drag force on the particle is $F_{drag}=6\pi\eta v_c a$, (a is the particle radius ,η the viscosity of the medium and v_c the speed of the medium versus the particle).

Fig 2.2 The example shows the ratio between F_{grad}/F_{scat}(=R) depending on the particle radius a. At the equilibrium these forces is equal, thus making R=1



0 < a < 50 nm
0 < z < 4 μm

3. Scanning methods

In order to perform a high-resolution scan several different techniques has been developed .
Wanted qualities are:

* Good linearity of response which insures repeatability of the movement.
* High resolution , preferably around 1 nm.
* Fast and stable positioning.
* Large positioning range.

Some different techniques will be discussed in the next section

3.1 Different techniques

During the years high precision movements has been conducted with fine screws and different types of micrometers. But these mechanical instruments have their limitations when they are reaching the nanometric environment. For instance the finest threads yields about 1 μm resolution[6]. There has been many attempts to try and transfer the well known technology of the macro and micro worlds to the nanoscale but new physical factors makes this technology unsuitable at this level. Why? Well, there are three traditional technologies for linear moving - dovetails, ball bearings and roller bearings[6] (Fig 3.1). Dovetail-motion designs are very simple and effective for long travels with high loads. However they are not appropriate for nanometric precision because of their high friction and stiction (break-away friction) which creates a bad repeatability. In the ball bearing system the sliding motion friction is replaced with the much lower friction of rolling motion. The problem is that mounting a high load on the ball will cause permanent damage thus limiting its use.In crossed roller bearing stages the point loading of the ball is replaced with a line contact to the bearing making it able to carry greater loads. However, this means the friction once again increases. It is also obvious that any small particle in the bearing could disturb the accuracy of the motion. These disadvantages has directed the attention to non-friction techniques by exploiting the fundamental properties of various materials. For instance elastic, plastic thermal-expansion, piezo-electric and thermoelectric properties. However, when designing equipment for use in the submicron world one need to consider new problems that are disturbing precise movement. For example thermal expansion that can create 100 nm fluctuations per °C in a 10 mm bar of steel[6]. Also one has to consider that dimensions of components will change by pressure and can change under the influence of electric and magnetic fields. Often it is necessary to use several different materials in combination with each other to minimize these effects. Of the various different methods to achieve nanometric precision two techniques has been noticed to give the best results and that's the piezoelectric actuators and flexure stages. The piezoelectric devices are using the fact that some materials change their shape when an electric field is applied. Piezoelectrics can respond in microsecond time constants and the positional resolution is only limited by the noise of the power supply. With different electronic controls positional resolution of a few nanometers are obtainable with a linearity of response of 1 % or better. Flexure stages are relying on the elastic deformation of solid materials. The flexing element merely constrains the motion so that the resulting stage moves in the desired direction, in orthogonal directions the rigidity is very large. Flexure stages have no internal friction, high stiffness , high load capacity , and a high

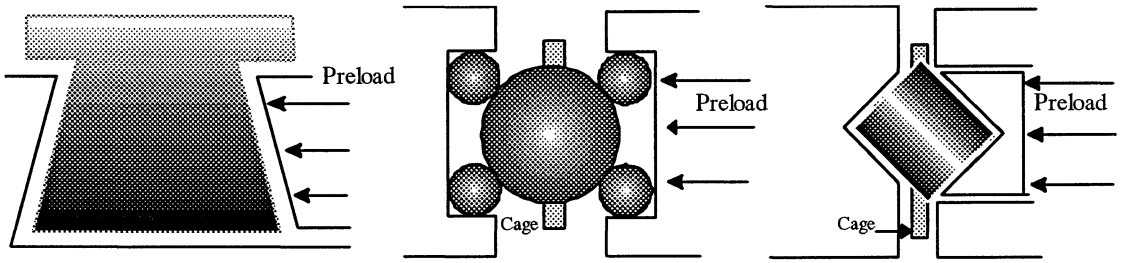


Fig 3.1 Different technologies for linear movement.

resistance to chock and vibrations making them suitable for applications that can accept certain constraints, e.g. very precise mounting and clamping of the flexing element and limited travelling range

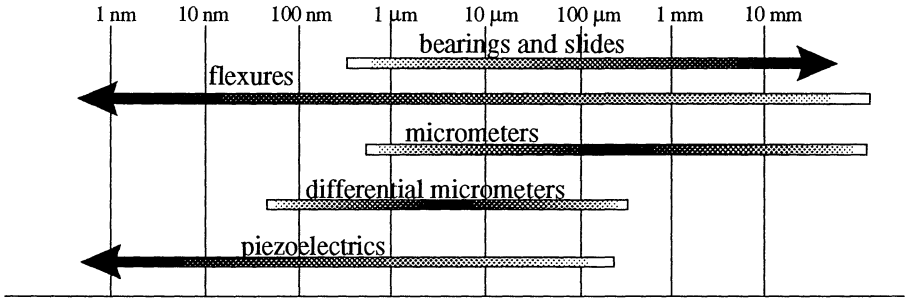


Fig 3.2 Technologies used at different scales.

3.2 Piezoelectric technology

In the 1880:s Pierre and Jaques Curie discovered that some materials can produce a voltage proportional to the pressure placed upon them[6]. Also the opposite effect was noticed. Several natural materials have these properties but most common is to use polycrystalline ceramics such as lead zirconate titanate (PZT)[6]. Piezoelectric ceramics must be poled for them to show piezoelectricity. Looking close to the material one notice that above the Curie temperature the electric dipoles in the material is randomly arranged. If a strong electric field is applied as the ceramics are cooled below the Curie temperature the dipoles remain partially aligned and respond collectively on small field changes thus producing dimension changes in the macro world. By looking at these facts some classifications can be done. For instance one should refer to devices that operate in the ferroelectric region below the Curie temperature as *piezoelectric* and for those operating in the paraelectric region above the Curie temperature as *electrostrictive*[6]. It is also common to name ceramics with an Curie temperature above 300°C as *hard* and consequently materials with a lower Curie temperature as *soft*. Traditional piezoelectric devices has had the disadvantage of needing voltages up to 2000 V for producing a useful extension thus making them expensive and dangerous to handle. The new generation of piezo actuators are able to use much lower voltage (0-150 V or less) making them more suitable for connection with standard electrical equipment. Unfortunately piezo ceramics has several drawbacks as they are associated with effects of nonlinearity, creep and hysteresis (Fig 3.3) making them not ideal for voltage-to-displacement devices. On the other hand they *are* stiction- and friction-free and able to produce movement from hundreds of microns down to the nanometric scale. These capabilities makes it worth the effort in trying to minimize the drawbacks. The hysteresis effect can for instance be reduced significantly if the material is preloaded with a force which will not only reduce hysteresis but also saturation effects and making the zero-truncation disappear (Fig 3.3).

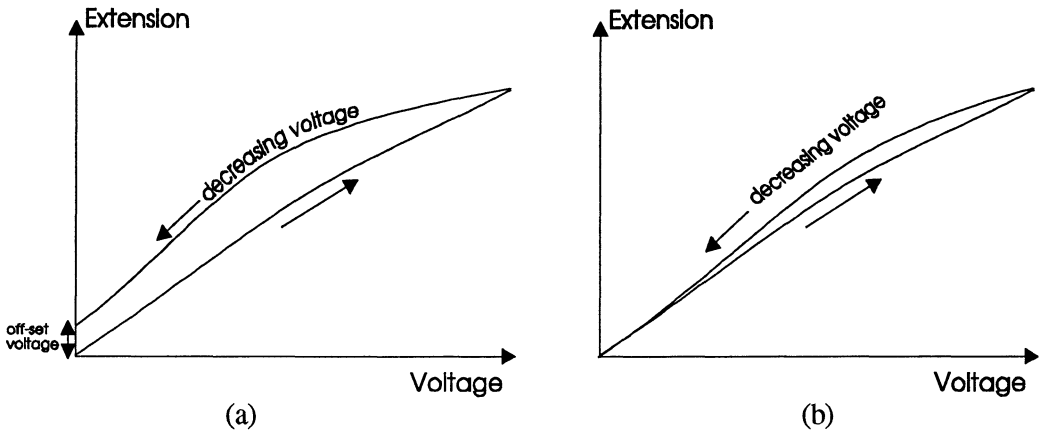


Fig 3.3 (a) shows a typical hysteresis behaviour of a piezomaterial. Note the off-set voltage that remains at zero.
(b) shows the effect of preloading the material with a force. The hysteresis is reduces and the off-set truncation disappears.

Another problem is that after changing the operating voltage of the device there will be further drift in the same direction, following the immediate movement. This creep can be several percent of the displacement but decreases with time making it negligible after a few seconds. In some application these effects can be accepted while in others they must be investigated and compensated for, e.g. with computer software. The most effective method, though, is to use a closed loop with position feedback sensors that will compensate for any incorrections in movement and making hysteresis and creep of no importance.

Several different geometry's has been developed for optimising resolution and/or travel range. Here the three-dimensional single tube is worth mentioning[7]. It uses a small dimension tube (12,7 mm long 6,35 mm in diameter and 0.51 mm thick) constructed with a piezoelectric material like the PZT and covered with an outside electrode sectioned into four equal areas and a single inside electrode. By applying a voltage to a single outside electrode, that segment of the tube is made to expand perpendicular to the electric field. This causes the whole tube to bend perpendicular to its axis and enables x-y movement. The z-motion is created by applying a voltage to the inner electrode which causes a uniform expansion of the tube. Typical response is 5 nm/V in each direction. Other more classical geometry's has been shown to reach resolutions of ≈ 4 Å/V.[8]

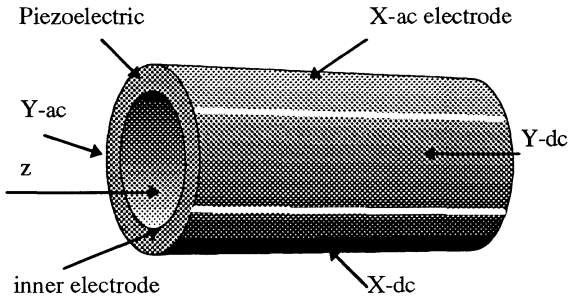


Fig 3.4 Tube scanner. The outside is sectioned in four equal areas. A small ac-signal and a large dc-signal is separated 180° apart.

4 Computer controlled scanning stage.

For making a controlled TPOM-scan a software program has been developed using the GPIB/IEEE-interface that is implemented in the piezocontroller and the lock-in amplifier. It not only controls the scanning movement but also the collection of data from the photomultiplier and presents it as an image.

4.1 Description of the software.

The basic idea of the program is :

1) Set all parameters of all electronic devices used in the scan
2) Set parameters that controls the behaviour of the scan.
3) Control the positioning of the piezo transducer.
4) Aquire information from the photomultiplier.
5) Present data on screen as a picture.
6) Save scanned data in a graphical or data format.

The program has been developed using the language Visual Basic, that has a somewhat object-oriented approach and has support for communicating with the Windows environment.

It is built with 7 major forms (windows) and 4 modules (*.bas -files). The modules contains the global declarations for variables used throughout the program and also procedures and functions that can be called from every form. The structure of the program is not using the ordinary MVC (model-view-controller) - model which is common in languages as Simula and Smalltalk but rather a totally event-driven model. For communicating with the Windows environment the program is using so called *.VBX files that serves as a buffer between the complex Windows routines and the programmer. However in some cases these have been insufficient and calls directly to the Windows so called DLL's (direct linking libraries) has been made.

The program is built around one main form that serves as a collector of scandata and as a launcher of all other forms in the program.

The following major forms has been implemented :

*** fMain:** Handles the input from the user regarding the scan parameters and also establishes connections with the piezocontroller and the lock-in amplifier and launches other forms.

*** fScanning:** Contains the actual scanning routines and also the image processing and the visual presentation of the scanned data.

*** fPiezo:** Makes it possible to control the piezoparameters and save them as default.

*** fLock-In:** Controls the lock-in amplifier and has possibilities to make different automatic initialisations. The parameters can be saved as default.

*** fGPIB:** Makes it possible to investigate the status of the IEEE-interface and make several tests of the instruments. It also contains links to the GPIB-spy that can monitor the traffic on the bus and to the National Instruments configuration program for the card and software.

*** fSaveAs:** Makes it possible to save the scandata in three different formats. (data, raw or as a bitmap)

*** fOpen:** Contains functions for a fast visual scan of previously saved images and makes it possible to open these including the data which they had when they were created.

The program also contains the following minor forms:

* **fSystem:** Checks the system resources for memory and computer hardware.
* **fErrorAnalys:** Presents an analyse of a GPIB-error if one occurs.
* **fErrorInfo:** Notifies that an error has occurred.
* **fResource:** Shows the system settings.
* **fInformation:** Presents the on-line help.
* **fLoading:** Contains a loading presentation.
* **fScanType:** Makes it possible to choose between three different types of scan routines.

The following modules are included:

* **INIT.bas:** Contains important functions in the program and also the declaration of variables, global constants and Windows DLL's.
* **Help.bas:** Contains the on-line help text.
* **NiGlobal.bas:** Contains global constants that are used in the GPIB-communication.
* **Vbib.bas:** Contains the declaration of the functions needed for communication with the supplied National Instruments "GPIB.DLL".
* **Declare.bas** : Contains the declarations for the system resources form.

The program also uses two *.INI files for storage of the default parameters that are loaded at program start-up. They are placed in the Windows system directory and are called TPOM.INI and DATE.INI.

4.2 GPIB communication

The program is communicating with the instrument via the GPIB (General Purpose Interface Bus) standard. It uses the new IEEE-488.2 standard that was defined 1987 which should insure compatibility between instruments from different manufacturers. However the Piezo controller is defined for the old standard. The limitations in the connection are[9]:

* Maximal cable length 20 m all-in-all, 2 m between instruments.
* 31 addresses for talking and 31 for listening.
* Maximum number of devices = 15 (31 if special extenders and electronics are used)
* Maximum transmission speed = 1MB/s under optimized conditions.
  (Normal speed is probably <<250 kB/s)

The communications is implemented in Visual Basic and from there going to a DLL that contains the functions for making low-level commands direct to the card (Fig 4.1). The National Instrument software contains three different possibilities to communicate: Via routines, functions or an universal language interface. The universal language interface is behaving as a I/O file thus making it possible to write and read to it from every programming language that has support for this. However the communication speed is reduced significantly. The functions should be used when there is one (or a few) instruments connected and the complexity is low. The routines gives more powerful commands making it possible to communicate with many instruments in one command. The program is using the routines as they have been shown to give the best results.

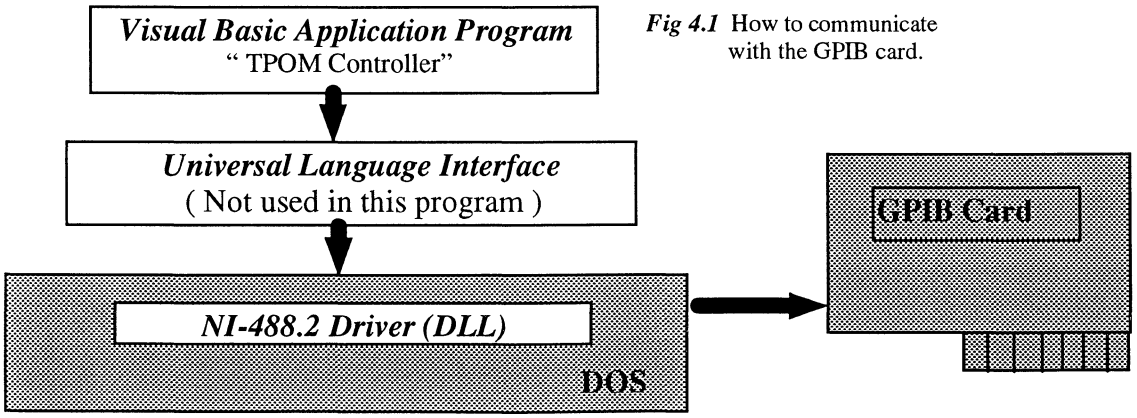
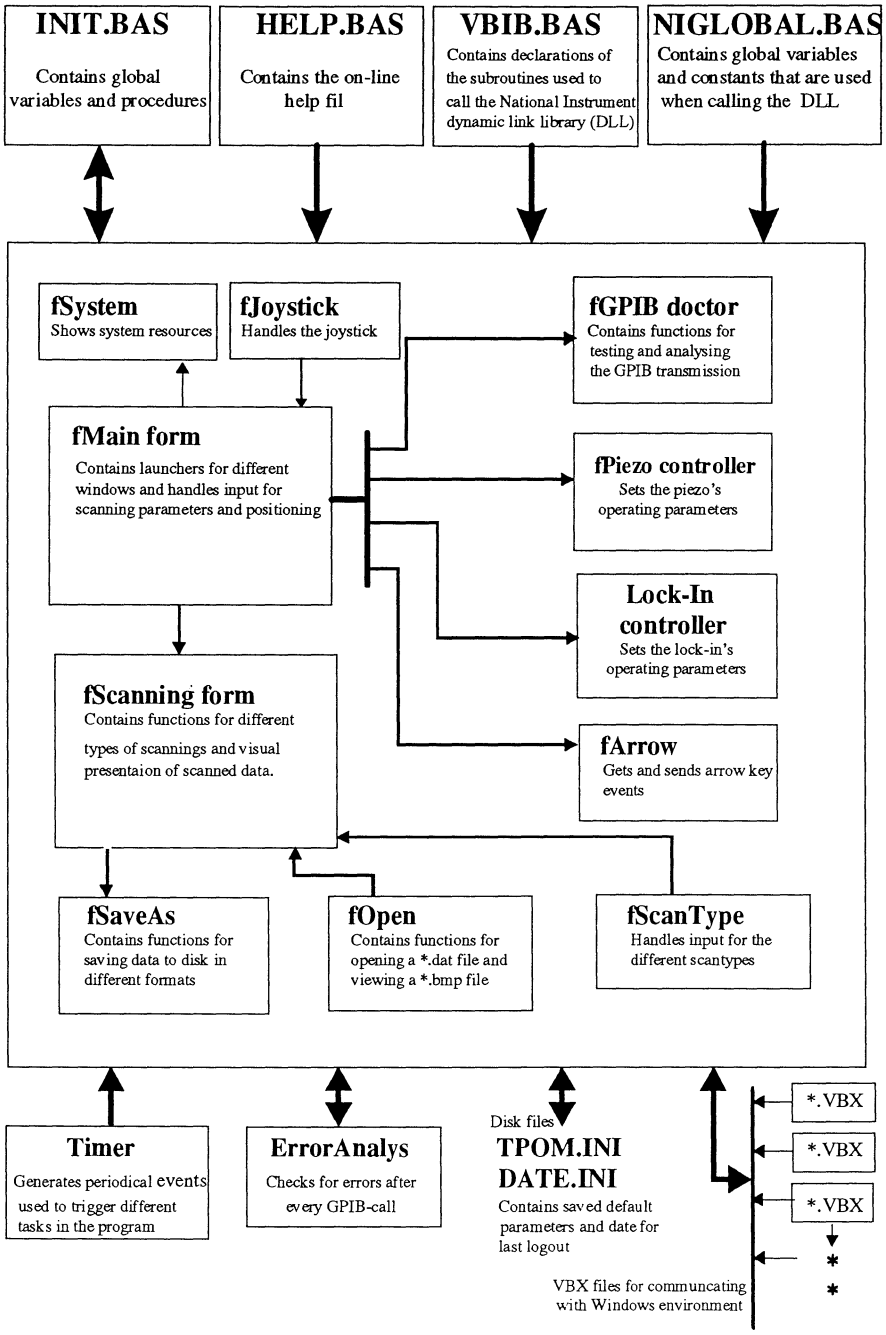


Fig 4.1 How to communicate with the GPIB card.

For correct communication it is necessary to configure the card/software. This is done using the ibconf program that can be found in the windows control panel. The correct parameters can be found in appendix F.

Fig 4.1 Overview of the program



Further information of the program can be obtained in appendix A-E.

4.3 Description of the hardware/electronics.

Besides the computer (a standard 386) and the stage which will be described in the next section the used equipment for the scanning control is a piezoelectric controller and a lock-in amplifier.

The lock-in amplifier that the software support is the model 5209 from EG&G Princeton Applied Research. The amplifier is equipped with several internal high quality filters for noise reduction and with a variable time constant to compensate for fluctuations in the signal. The preamplifier can give a response up to 10^8 V/A that is ideal for our SNOM arrangement where the signals can be very small. The piezocontroller used is the model 17 PCZ 013 from Melles Griot in Cambridge with three independent channels. Both the lock-in and the piezocontroller are equipped with IEEE/GPIB-interface for communicating with a computer. The used GPIB-interface is a National Instrument PCII/A card and software configuration programs to enable correct communication between talkers and listeners. As the program is built with separate modules it is possible to change these instruments while simultaneously creating a new software module and link it to the program. However some calls in the other modules must also be changed if the new equipment is not using the same commands in the GPIB-calls.

5 Experiments

To be able to verify the performance of the piezoelectric equipment and to investigate the computer software reliability several tests have been performed. Experiments has also been done to make sure that the scanning speed or possible vibrations by the transducer is not affecting the trapping. Finally testobjects has been constructed and the general ability of the microscope has been investigated.

5.1 Resolution and linearity of response of the piezoelectric stage

The used equipment was a NanoFlex™ Integral X-Y Flexure Stage (Fig 5.1) with long extension piezos. This stage provides 5 mm of fine position adjustment without friction or stiction[6]. It also features differential micrometers that provides 300 μm of precision adjustment with 100 nm resolution. In addition internal piezoelectric actuators provides 200 μm travel with a resolution of 50 nm. It is designed of steel and aluminium which gives it a temperature invariant performance. To drive this stage the piezoelectric controller 17PCZ003 from Melles Griot was chosen. It is a three channel controller with active closed loop feedback. The feedback gives an automatic compensation for any small positional perturbations and eliminates creep and hysteresis. The linearity of response which describes the ability of following a linear movement is ± 0.5 % of full scale (200 μm) meaning 1 μm in this case. The feedback loop can perform closed-loop operation at frequencies up to 300 Hz with a step response of 2 to 3 ms. In our case we are using steps around 100 ms making the speed performance very satisfying. The controller is also equipped with an IEEE / GPIB interface with a 16-bit digital-to-analog converter.To check the manufacturers data, experiments have been performed using both a digital length gauge and by building a Michelson interferometer.

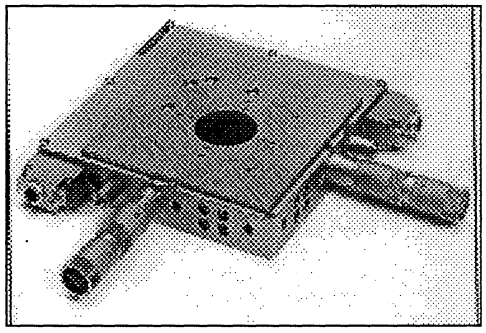


Fig 5.1 The NanoFlex integral flexure stage

.5.2 Resolution measurments

To measure the resolution a Michelson interferometer was built which had the piezoelectrical stage as one of it's arms (Fig 5.2). As a beamsplitter a 50 % reflective grayfilter was used. To make both arms have a more similar intensity a 50 % grayfilter was used in one arm. An aperture was also put in the arrangement as it resulted in much higher visibility at the photodiode. As the HeNe laser had a wavelength of 632.8 nm a movement by the piezo of $\lambda/2$ = 317 nm should create one period at the interference pattern. Several measurements were made indicating that the piezo had a performance much worse than the manufacturers data. The equipment showed both hysteresis and bad repeatability so it was sent back to England for calibration. The tests are shown in Fig 5.3.

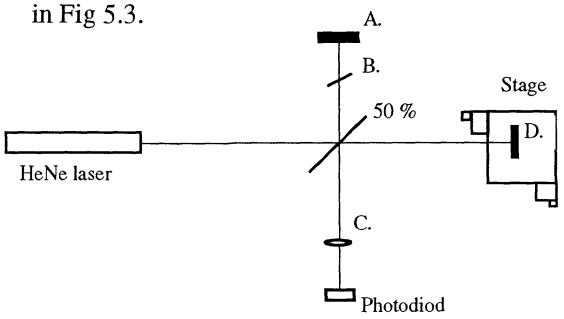


Fig 5.2 The used Michelson interferometer. (A) and (D) are mirrors, (B) is a compensation 50% filter and (C) an aperture.

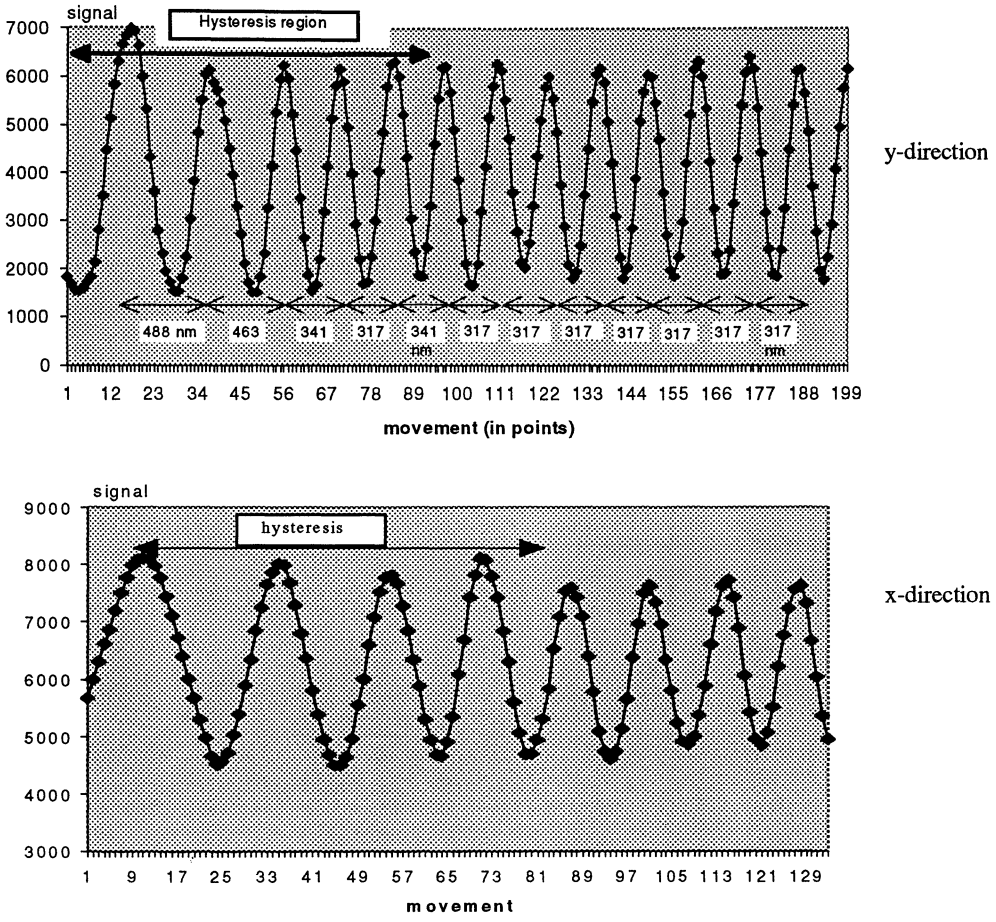


Fig 5.3 The interference signal at the photodiod. The distance between every point is 24,4 nm. Due to hysteresis it takes some periods before the stage is responding in a linear way. This is unacceptable for our scanning design as we are scanning right in the hysteresis region.

When the equipment returned 4 weeks later new tests were made using the interferometer but also with a digital length gauge (Heidenhain MT12B) (Fig 5.5) with a resolution of 50 nm. The piezo was driven by the computer and a curve was plotted from the readings of the Heidenhain (Fig 5.4).

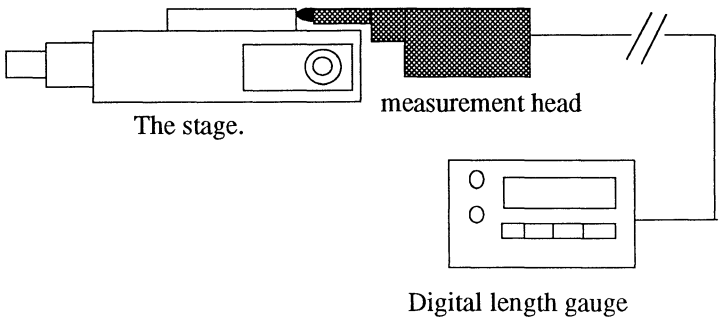


Fig 5.5 The arrangement for the Heidenhahn measurement

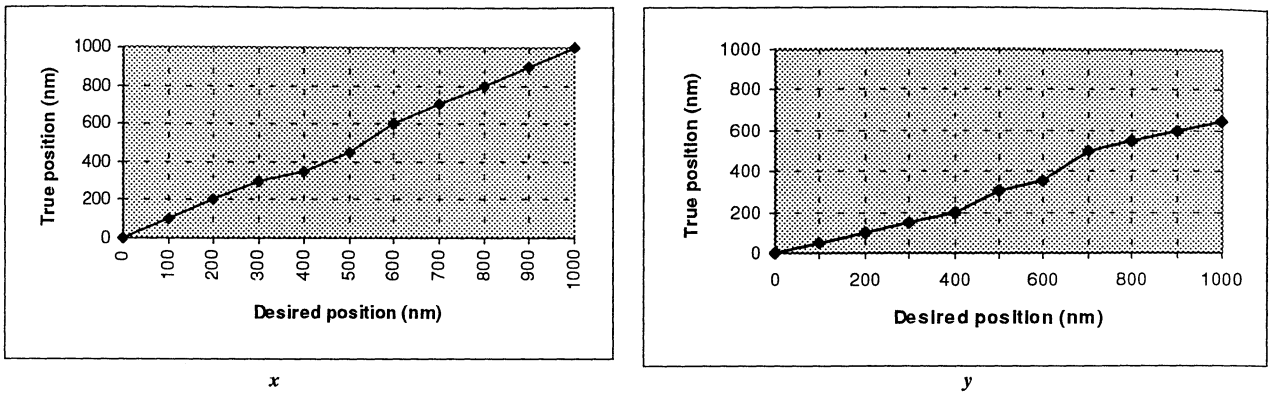


Fig 5.4 Measurements with the Heidenhain shows that the piezo is behaving acceptable in the x-direction but not in the y-direction

The new interferometer readings shows that the x-direction was significantely improved. However, the y-direction still exibits significant hysteresis. See Fig 5.4 and 5.5.

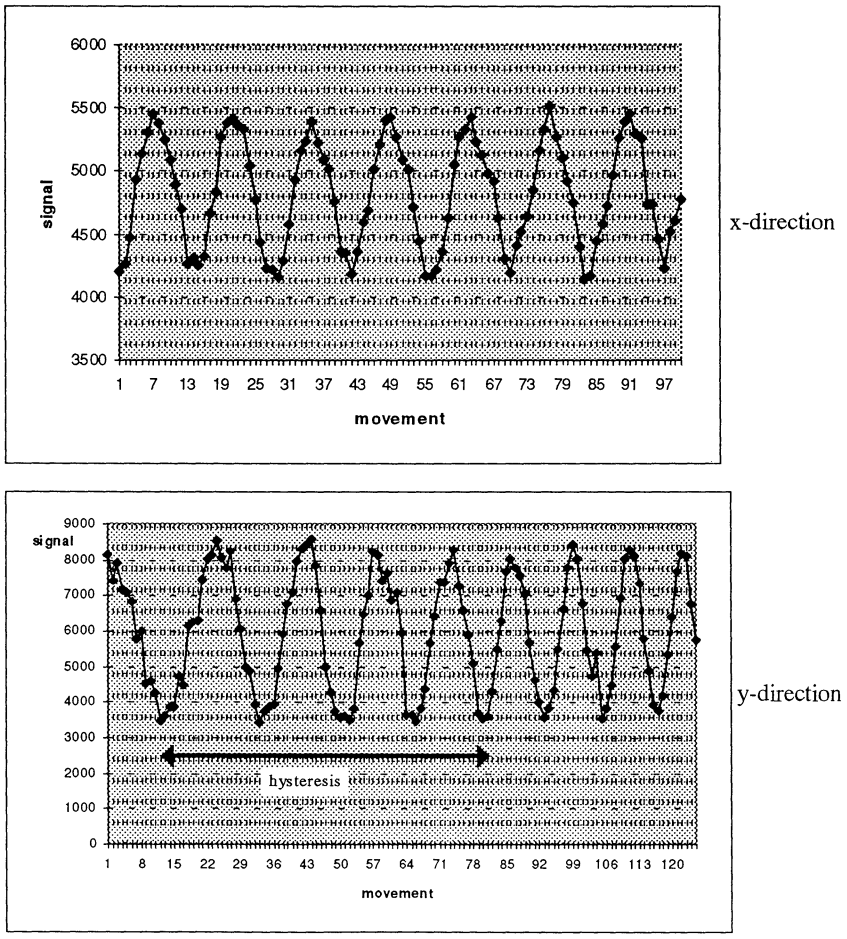


Fig 5.5 The new measurements shows that the stage is moving well in the x-direction but that hysteresis is still present in the y-direction. (Distance between points = 24,4 nm)

5.3 Experimental results

Even though the performance of the piezo controller has been somewhat disappointing it should be possible to test the capability of the microscope. As the x-direction is working well, at least a line scan can be made. A square scan could also be performed. It will however be difficult to make any conclusions of the resolution in the y-direction. Using the interferometer and scanning the same linescan while moving in the orthogonal direction shows that the piezo is returning to almost the same startpoint every time with an error of approx. ± 50 nm (Fig 5.6). However, a stage/controller with better resolution should be discussed. A factor 10 better performance should make it easier to work with and should also make possible hysteresis and non-linearity less critical.

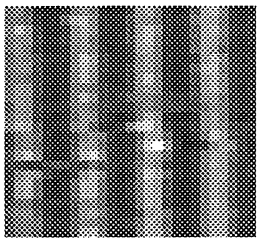


Fig 5.6
Repeated linescan using the interferometer. Each pixel is 50 nm. Dark shows maximum and light minimum.

5.4 Construction of test object

To be able to investigate the resolution of the microscope a testobject with known properties must be used. For that purpose a testobject consisting of 205 nm diameter microspheres has been constructed[13]. These particles are made of dyed polystyrene with a fluorescent dye and supplied by Duke Scientific Corporation. The particles, which were dissolved in water were dispersed in PMMA (Polymethyl Methacrylate) with chlorobenzene. As water and chlorobenzene do not mix several other solvents were tried and also centrifugation of the particles to reduce the water content. Finally a solution of 0.08g ethanol, 0.67g PMMA and 0.02g of the particle solution were shown to give a satisfactory mixture. This sample was spin-coated at the spinning speed of 4000 rpm for 45 seconds after 5-second prespinning at 500 rpm. The result is a slide glass with particles baked in the PMMA (Fig 5.7). The thickness of the PMMA layer is difficult to predict as it is mixed with ethanol. Used by itself it should spin out to 60 nm but a guess is that mixed with ethanol gives less thickness. The important thing is, however, that the thickness is not exceeding the particle diameter (205 nm) as the trapped laser probe must be able to get very close to the particle. Measurements using the TPOM on this object were however difficult to perform due to bad stability in the trap.

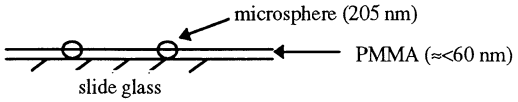


Fig 5.7 The test object.

6 Conclusions

A computer controlled scanning stage has been constructed. The system is able to control a piezoelectrical stage with 25 nm resolution and collect data from a photomultiplier with a maximum speed of 30 readings(pixels)/sec. The developed computer software is able to manipulate the collected data, present them as a picture and save the data in different graphical formats. The accuracy of the scanning stage has been tested and documented. The performance is not yet totally satisfying due to the manufacturers problem to calibrate the piezocontroller. A better calibration will make it possible to use the scanning system for nonintrusive SNOM measurements.

7 ACKNOWLEDGEMENTS

I am very grateful for all the help I received from the people working at the Division of Atomic Physic, Lund. Especially I would like to mention the following people:

Hans Hertz, my instructor who supported me from day one and who has a great part on this diploma paper.

Lars Malmqvist, my co-instructor who has helped my with so many things that I can't remember them all. Every time I was standing in front of a big concrete wall he somehow managed to drive around it.

Anders Persson, the computer expert who is a source of information larger than the Internet.

Göran Werner, for giving me education on the grinding and drilling equipment.

Microsoft™, for helping me with the drivers for the joystick handling.

I would also like to thank **Magnus Berglund**, **Peter Bårmann** and **Lars Rymell** for valuable and/or amusing discussions.

References

[1] L. Malmqvist: " Nonintrusive probes for scanned near-field optical microscopy " , Lund Reports On Atomic Physics LRAP-160 (1994).

[2] L. Malmqvist and H.M. Hertz: " Trapped particle optical microscopy ", Opt commun. **94**,19 (1992).

[3] L. Malmqvist and H.M. Hertz: " Second harmonic generation in optically trapped nonlinear particles using pulsed lasers", Appl. Opt. **34**, 3392 (1995).

[4] L. Malmqvist and H.M. Hertz: " Two-color trapped particle optical microscopy ", Opt. Lett. **19**, 853 (1994).

[5] L. Rymell and L. Malmqvist: " Optical Trapping ", Lund Report On Atomic Physics LRAP-123 (1991).

[6] Nanopositioning Guide by Melles Griot® (1993).

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

[8] B. Drake, R. Sonnenfeld, J. Schneir and P.K. Hansma: " Tunneling microscope for operation in fluids", Rev. Sci. Instrum **57**, 441 (1986).

[9] L. Grahm, H.G Jubrink, A. Lauber: " Modern Elektrisk Mätteknik, del 2 ", (Bokförlaget Teknikinformation, 1994) p.126.

[10] E. Hecht and A. Zajac: "Optics",(Addison-Wesly publishing company,1974).

[11] S. Sato and H. Inaba, Electron. Lett, **28**, 283(1992).

[12] D.W. Pohl: " Advances in Optical and Electron Microscopy ", C.J.R Shepard and T.Mulvey, eds., Vol 12, pp. 243-312, (Academic Press, London,1991).

[13] S. Kawata, Y. Inouye and T. Sigiura: " Near-Field Scanning Optical Microscope with a Laser Trapped Probe", Appl. Phys, **33,** pp. L1725-L1727, (1994).

Appendix A Instructions for general use of the program.

When using the program the following steps should be followed in an ordinary start-up.

During start-up the files "tpom.ini" and "date.ini" will be read and the variables there will be used to set the parameters for the piezo and the lock-in. The date when the program last was used will be checked so the program can give a correct recommended filename when saving a scan.
* When the program is started it will be in "basic" mode. That means it's not connected to any instruments and as a result no parameters are set yet.
* The first step is therefore to establish a connection to the instruments by clicking the "Connect to piezo/ lock-in" images at the far left.
* The second step is now to send the default parameters to the instruments. That is done by clicking on the buttons that says "piezo" and "lock-in" thus launching that controller and at the same time sending the parameters. Usually the piezo's parameters are not changed from time to time but the lock-in parameters probably has to be changed from experiment to experiment depending on signal strength, timeconstant etc... A fast way to do this is to make an "Auto initialisation" or an "Auto measure" where the first choice is making an automatic sensitivity, tuning, phasing and offset but leaving all parameters as filters etc.. unchanged while the second choice makes a auto measure as it can be done at the front panel in manual mode, thus changing all parameters to a "normal" setting.
* The instrument's controllers can now be closed and the scanning properties may be set to wanted values.
* When this is done the scanning form is loaded by clicking the "scan" button which launches the scanner.
* The program is now ready to make a scan . (If clicking "Type" different scantypes can be selected)

Open scan:
The open scan window contains a possibility to take a quick look at saved bitmaps and to open a *.dat file. Just double-click a bitmap and it will show to the left. When opening a data file all the scan-parameters in the program will be altered to the values they had at the time they were saved.

Save scan:
It is wise to save the scan in both bitmap and data format as you then will have both an image as well as the data that was used when the image was created.

LIMITATIONS:

The graphics will limit the dimension of the image to be no greater than 400x400 pixels. The graphics can be disconnected just by disabling the marked rows in the loading procedure of the scanning form. Bigger pictures can then be scanned and saved as a bitmap for further processing.

The other limitation is that the array that are storing the picture can have a maximum of 32 768 data points thus limiting the size to be 181x181 pixels. However this is not a final limit, it's possible to double the capacity if using index's running from -32 768 to 32 768 or you can make the array multidimensional which makes it possible to have an image only limited by the computer's memory. These implementations has not been done as the scantime is going by the square of the size and scans >181 pixels will take hours or even days to accomplish and is impossible to do with the current TPOM trap.

Appendix B Hints

The following problems has been noticed to appear.

* *Sometimes the lock-in does not connect in the first try !*
  It is a usual problem if you are clicking to fast but it's solved just by trying once again clicking more slowly.
* *The program "hangs" when launching the lock-in controller !*
  It is probably waiting for the lock-in to respond to a serial poll and it can't because you didn't connect or did not turn it on. The polling can be stopped by clicking anywhere on the form and answering yes at the message box. Then fix the problem and try again.
* *The piezo is "drifting" !*
  You have an uncalibrated joystick. Calibrate the zeropoint with help of the calibration program or if you have just started Windows calibrate it at the control panel. This is done by clicking the "advance" button at the cal. program and the clicking the joystick icon followed by Ok!
* *Nothing happens when I press "stop" while scanning !*
  Give it some time. The scanning routines listens for outside events rather seldom to improve their speed.
* *The program gives an error when opening a file* !
  That there is a bitmap file that you can see is no guarantee that there are also a data file that the program can open.
* *The scanning form don't appear when called upon* !
  If you will be making a scan with many pixels the calculations for the graphics can take up to 1 min to get finished (with a 386 processor).
* *The "auto-size" is not working when looking at saved files !*
  This happens for some bitmap files. There is no other solution than to use the "normal-size" option.

Appendix C Installing the program on another computer.

Installing the executable program on another computer is very simple. Just use the installation disk and all necessary files will automatically be installed. When starting the program for the first time at a new computer an installation program will create the following directories.
c:\scanning\1995\1jan, 2feb, 3mar, 4apr, 5may, 6june, 7july, 8aug , 90sep, 91oct, 92nov,93dec
1996\ ---------- " ----------
as the program needs them to save it's scanned data.(The months begins with numbers to get correct sorting).

If you want to install not only the executable but the Visual Basic code to be able to make code changes use disk B which contains the different files that is necessary to make the program work. Start a new project and add the following files from the disk.

Erranal.frm	**Lock_3d.frm**	**fArrows.frm**	**fPiezo.frm**
Errifo.frm	**Openfile.frm**	**fGPIB.frm**	**fSaveAs.frm**
Joystick.frm	**Grid1.frm**	**fInforma.frm**	**fScanTyp.frm**
Loadinf.frm	**Resource.frm**	**fMain.frm**	

DECLARES.BAS	**HELP.BAS**	**INIT.BAS**
NIGLOBAL.BAS	**VBIB.BAS**	

Also the following files must be placed under the c:\windows\system\ directory :
tpom.ini, date.ini, vb.lic, grid.vbx, picclip.vbx, spin.vbx, threed.vbx
In both cases the joystick driver must be installed at the Windows control panel. Choose the option drivers and add **ibmjoy.drv** from Disk B.

Appendix D The on-line help file

This is a copy off the on-line help file that is used in the program.

"ON-LINE HELP"

****

Case "MAIN Window"

---------- CONNECTIONS ----------"
By pressing either picture or text is it possible to connect or disconnect"
to Piezo/Lock-In . The connection is putting both instruments in remote "
mode and makes frontpanel controls inoperative"
Note: If an error occurs when connecting is it most likely that you have"
trying to connect the instruments with to small a time apart."
Trying once more slower usually solves this problem."

---------- SCANNING PROPERTIES ----------"
SIZE : Input scanningsize in pixels , only square size is accepted."
DELAY: Decides the delaytime between to input values(pixels) during"
scanning in milliseconds."
X/Y-STEP: Controls the steps in x/y-directions while making a scan or"
when moving around with arrowclicking. If Same Step is "
selected both x and y steps takes on the same value"
TYPE OF SCAN: Selects square / line scanning"
SCAN TIME: Gives an estimated time for chosen scan properties"
COMPENSATE: Makes it possible to compensate for linearly faults in the "
piezo"
Example: If you write in 50 nm and compensate 100 % then"
the program will act as the input was 100 nm"

---------- POSITIONING ---------- "
SET x/y:Input the desired position in um and send that position to "
piezo either by hitting return (sends x OR y pos. depending "
on cursor placement) or clicking the SEND button (sends both "
x AND y position) "
POSITION SENT: Shows last sent position to piezo"
POS. RECEIVED: Reads and displays current position given by the "
piezo then START RECEIVING is selected"
NOTE: Unfortunately is this reading fluctuating and"
is not a good indicator for the exact position"
Instead the sent position should be taken as"
reference "
ARROW'S: By clicking the arrows positioning up/down, left/right can be"
made with steps chosen with X- and Y-STEP"
JOYSTICK: Enables joystick handling (When starting windows "
calibrate the joystick by clicking on ""advance"")"
ESCAPE : If the piezo for some reason hangs clicking this button resets it"
LAUNCHERS: Clicking these buttons launches selected controller"

---------- MENU ----------"
EXIT: Exit TPOM from menu should always be done as this insures a "
correct closing of file's and saving of important parameters."
IF exiting in other way BAD performance can occur in next login !!"
ERRORS: Selecting HARD error checking makes an error message "
with analysing tools occur every time GPIB doctor notice "
something strange on the bus. Selecting SOFT updates only"
the global variable ErrorChecking that keeps track on the "
number of errors detected. That variable can be checked if"

launching the GPIB doctor"
SYSTEM: Gives information of the system settings and available resources"
CALIBRATE: Makes it possible to calibrate the joystick so the piezo "
won't drift"
HELP: Get this help"

Case "LOCK-IN"

---------- **SENSITIVITY** ----------"
SENSITIVITY: Decides the input sensitivity on the lock-in ."
Range= 100nV-3 V"
AUTOSENSITIVITY: Makes an automatic choice to set output between"
30-100% of full scale"
FILTERS: Decide the type of filtration for the input signal before "
locking on signal (BP,LP,Notch and Flat)."
TRACK : If track is selected the filter tunes to the reference frequency"
/MANUAL If manual is selected it tunes to the freq set by the <- and ->"
keys. Best operation is performed by using track to tune to"
ref freq and then switch to manual to take advantage of the"
higher stability in that mode."
AUTOTUNE : If the lock in is in manual mode making an autotune will"
set the tuning freq to ref. freq. In track mode autotuning"
has no effect "
LINE REJECT : Setting Line reject to F puts an extra notch filter on the"
line frequency. 2F puts the filter in double line freq"
2F+F gives you two filters and selecting NONE gives"
no filtering.These line filters are completely independent"
of the tuned filters settings."

---------- **TUNING** ----------"
By clicking <- and -> keys changing the tuning freq is possible. If "
Ref F is selected the filters tune to the ref freq. If other range is "
selected Manual mode will be chosen automatically and changing "
tuning freq will be possible"
AUTOPHASE : Causes the ref. channel's phase shift to be adjusted"
for maximum output"

---------- **REFERENCE** ---------- "
INTERNAL : If Internal is checked the lock-in locks to the internal "
freq. generated by the lock-in.(say when you trigger"
the experiment with the lock-in OSC OUT connector)."
If it is unchecked it locks to the signal applied to the "
REF IN connector or the TTL REF IN connector."
2xF: If this choice is selected the ref.freq operates on twice the "
freq of the applied signal. If not selected it operates in the same "
frequency"

---------- **OUTPUT** ----------"
TIME CONSTANT: The longer timeconstant the narrower the lock-in"
amplifiers noise bandwidth will be and the better "
signal-to-noise ratio.The price is an increased "
respond time"
RESERV : Selects the dynamic reserv that gives 20,40 or 60 dB "
respectively. Reserve and output stability are tradeoff"
parameters. HI STAB gives an output stability of 5 ppm/C"
NORMAL 50 ppm/Celsius and HI RES 500 ppm/C"
EXPAND : Expands the output 10 times after offset"
OFFSET : Select the offset value"
AUTO OFFSET : Offsets the output to zero automatically"
SLOPE : Selects the timeconstants filter's rolloff rate 6 or 12 dB/octave"
12 dB is better but can't be used in all experiments (i.e. feed-"

back loops)"
OUTPUTS: Six different outputs is possible:"
% FS : Shows the lock-in output in % of full scale for all "
sensitivities. This is default"
SIGNAL: Shows the actual output in volts"
OFFSET : Shows the selected offset value. Range:+-1.5 FS"
NOISE : Shows the rectified output noise in % of full scale"
RATIO : Indicates the ratio between the lock-in's OUTPUT"
to the level applied to the rear-panel CH ADC AUX"
INPUT"
LOG RATIO : Shows the log of the ratio"
READ OUTPUT: Reads the selected display output and display's it "
every 250 ms"

---------- **MENU** ----------"
AUTOINIT : Trigger's all auto function's but leaves all other parameters "
unaffected"
AUTOMESURE : Makes an automesure on the lock-in. This autofunction"
changes some parameters like the filters and "
timeconstant for example. It is the same procedure "
that can be done from the front panel."
*Automesure doesn't do an auto offset as AutoInit "
SAVE PARAMETERS : Saves the current parameters as default which"
will be loaded next time the program starts"
NOTE: No autofunctions will be saved, instead will the "
last value before the auto function was performed"
be saved"

Case "PIEZO"

---------- **PIEZO CONTROLLER** ----------"
INPUTS : If these options are checked the Piezo goes into closed "
loop operation. Unchecked they put it in open loop operation"
NOTE: If the stage is not connected these inputs don't work"
SCROLL BARS : Moving these scroll bars changes the um/volt on the"
piezo"
UM/VOLT : Sets wanted operation on the piezo"
START READING : Reads all three channels with a refresh rate of 1/3 s"
SAVE : Saves current parameters to file as default"

Case "GPIB"

---------- **GPIB DOCTOR** ----------"
FIND DEVICES : Gets out on the bus and makes an investigation"
of the currently connected instruments. The result"
is presented as an array (1-32) of addresses to the"
instruments. The first address is 0 which is pointing"
at the GPIB card"
ADDRESSES : Makes it possible to change the addresses that the "
program uses to call for the instruments."
NOTE : If you change this address and you don't have the "
same address configured on the instrument you "
are asking for trouble"

---------- **CONTROLS** ----------"
CLEAR INSTR : Sends a device clear to ALL instruments on the bus"
CLEAR GPIB : Send an interface clear signal to the GPIB card"
SERIAL POLL TEST : Makes an serial poll to the lock-in and analyse"
how it handles it"
ERRORS DETECTED : Shows numbers of errors detected since the "
program started. If soft error checking is "
selected one should look at this variable from"

time to time to ensure the program is running"
smoothly"
ADVANCED : Connects to the Control Panel there you can start the "
GPIB (IBCONF) program to make detailed configuration"
of the software"
SPY : Starts the GPIB spy program that shows every call made over the "
GPIB card"

Case "OPEN"

---------- OPEN SCAN ----------"
Makes it possible to open a saved *.dat file containing info of the scan"
and the scanned picture itself. If you have previously saved the scan "
in both data and bitmap format you can take a quick look at the scan"
by choosing the corresponding bitmap file (Double click the file name"
in the file list, list box)."
If you are looking at the bitmapfile, pressing OPEN will open the *.dat "
file with the same name"

Case "SAVE"

---------- SAVE SCAN ----------"
Before using this program you should create the directory c:\scannings"
You should then create sub directories with names that represents"
different month's(years) . Example: c:\scannings "
\1994\ jan, feb..... dec"
\1995\ jan, feb.....dec"
FILE FORMAT : Three different saving format's is possible :"
RAW : This format contains the information of the picture"
saved as binary bytes and can be read by i.e. Pub "
PaintBrush"
BITMAP : Saves the information as a device independent"
bitmap (DIB) , defined by the windows environment"
Can be read by all design program's that can "
handle bitmaps. The format is uncompressed "
DATA : Saves the info as a *.dat file containing the pixels"
as ASCII characters plus information of the scan "
data. This file can be opened by the program or"
a text editor"
IMPORTANT !!: The five first data values are used by the"
program and should not be considered"
as picture information (say if you are"
using EXCEL for plotting the scan)"
SAVE AS *.BMP and *.DAT : This is the default setting "
and enables both saving of scan data "
and graphic file saving"
RECOMMENDED : The program gives you a recommended name of "
FILENAME the file which contains of the present day, month"
and a nbr that shows how many files you have "
been saving. This number is updated even if you "
close and start the program on the same day."
Starting the program for the first time a new day"
resets the number"

NOTE : If you start changing the recommended file name"
you can't expect the program to give good rec. "
filenames that day"

Case "SCANNING"

---------- SCANNING ----------"
GRIDLINES : Switches gridlines on/off"

ZOOM : Switches the form into zoom mode. By moving to the picture "
and clicking the LEFT mousebutton makes a Zoom In on the "
picture with the zoomdegree chosen at the menu."
Pressing the RIGHT mousebutton makes a Zoom Out."
Clicking the zoombutton once more puts the form back to "
normal mode"
TEST : This gives you four options:"
GRAYSCALE TEST : Makes a grayscale (1-64) test "
on the grid."
RANDOM TEST : Makes a random test on the grid."
CLEAR GRID : Clears the grid."
COLORS : Makes it possible to change colors on the grid"
OPTIMIZE : Sets the maxvalue in pixelpicture to be white and paints "
the picture down from that"
CONTRAST : Makes it possible to change the contrast. The picture"
should have a normal spread for best result "
DATA : Shows the scandata of the picture. "
SCAN : Starts the scan. The scan can be either stopped by pressing"
STOP or paused by pressing PAUSE. Clicking start (pause) "
button again resumes scanning."
COL/ROW : By clicking a pixel these numbers give the position of "
that pixel"

---------- **MENU** ----------"
SAVE : Calls for saving the scan."
OPEN : Calls for open a scan."
ZOOM : Selects the zoomdegree used by the zoombutton . Choosing"
NORMAL gives the default picture zoomdegree"
HELP : Get this help."

Appendix E TPOM controller source code

This sourcecode is a transcript of the following forms:

The following forms and modules are left out: Joystick.frm, Erranal.frm, Errinfo.frm, Loading.frm Resource.frm, fScanTyp.frm, VBIB.bas and Declare.bas as they contains either very little routine code or code which is unimportant for the behaviour of the program.

****

A) MAIN FORM Contains the launching of controllers and the input of scandata.

(fMain.frm)

****

```
Option Explicit
Dim OldX As Single
Dim OldY As Single

'***************** Connects (index=0) or disconnects (index=1) the lock-in amplifer
Sub c_ConnectLockIn_Click (index As Integer)
 If index = 0 Then
   c_ConnectLockIn(0).Visible = False
   c_ConnectLockIn(1).Visible = True
   Call InitGPIB
   Call Connect("LockIn")
 Else
   Call Disconnect("LockIn")
   c_ConnectLockIn(0).Visible = True
   c_ConnectLockIn(1).Visible = False
End If
End Sub

'***************** Connects (index=0) or disconnects (index=1) the piezocontroller
Sub c_ConnectPiezo_click (index As Integer)
 If index = 0 Then
   c_ConnectPiezo(0).Visible = False
   c_ConnectPiezo(1).Visible = True
   Call InitGPIB
   Call Connect("Piezo")
 Else
   Call Disconnect("Piezo")
   c_ConnectPiezo(0).Visible = True
   c_ConnectPiezo(1).Visible = False
 End If
End Sub


'************************ Command button that is used to communicate with the fArrow form
Sub c_Down_Click ()
   Call ChangePosition("DOWN", YStep)
   text_SetY.Text = label_SentY.Caption
End Sub
```

```
'******************************** Launches the different windows
Sub c_Launchers_Click (index As Integer)
  Select Case index
    Case 0: fLock.Show
    Case 1: fPiezo.Show
    Case 2: fScanning.Show
    Case 3: fGPIB.Show
  End Select
End Sub

'************************* Command button that is used to communicate with the fArrow form
Sub c_Left_Click ()
    Call ChangePosition("LEFT", XStep)
    text_SetX.Text = label_SentX.Caption
End Sub

'************************* Command button that is used to communicate with the fArrow form
Sub c_Right_Click ()
   Call ChangePosition("RIGHT", XStep)
   text_SetX.Text = label_SentX.Caption
End Sub

'*****************************Decides the scantype , square ot line
Sub c_ScanType_Click (index As Integer, Value As Integer)
  If index = 1 Then
     SquareScan = True
    Else
     SquareScan = False
  End If
End Sub

'*************************** Send's the current wanted position to the piezo
Sub c_Send_Click ()
Dim Finish As Single
Dim Start As Single
Dim Direction As String

    Finish = Val(text_SetX.Text)                '********** Get wanted position
    If Finish > 199 Then GoTo ending
    Start = OldX                                '********** Get startposition
    If (Start - Finish) > 0 Then Direction = "LEFT" Else Direction = "RIGHT"
    Call MoveTo(Start, Finish, Direction)       '********** Move in small steps
    XAsBin = UmToBin(Finish) + 62
    Call Send(0, Piezo_Number, "O1+" & Str$(XAsBin) & Chr$(13), DABend)  '****** Fine tune
    label_SentX.Caption = text_SetX.Text
    OldX = Finish
    '********** Save current position
    Finish = Val(text_SetY.Text)
    If Finish > 199 Then GoTo ending
    Start = OldY
    If (Start - Finish) > 0 Then Direction = "DOWN" Else Direction = "UP"
    Call MoveTo(Start, Finish, Direction)
    YAsBin = UmToBin(Finish) + 29
    Call Send(0, Piezo_Number, "O2+" & Str$(YAsBin) & Chr$(13), DABend)
    label_SentY.Caption = text_SetY.Text
    OldY = Finish
    Exit Sub
ending:
MsgBox " Maximum input is 199 um", , "Info"
Exit Sub
End Sub

'************************* Command button that is used to communicate with the fArrow form
Sub c_Up_Click ()
   Call ChangePosition("UP", YStep)
   text_SetY.Text = label_SentY.Caption
End Sub
```

```
'*********************************- Changes position (Stepp in nanometer)
Sub ChangePosition (Direction As String, Stepp As Single)
Dim xCurrent As Single
Dim YCurrent As Single
Dim NewPosition As Long
  '****************** Get current position
  xCurrent = Val(label_SentX.Caption)
  YCurrent = Val(label_SentY.Caption)
  If OutOfRange(xCurrent, YCurrent, Stepp, Direction) = True Then
    MsgBox "You are going out the maximum range ! ", 16, "Overload"
    Exit Sub
  End If
  '****************** Move with chosen steps in input direction
 Select Case Direction
   Case "UP"
    NewPosition = UmToBin(YCurrent + Stepp / 1000) + 29
    Call Send(0, Piezo_Number, "O2+" & Str$(NewPosition) & Chr$(13), DABend)
    label_SentY.Caption = Format(YCurrent + Stepp / 1000, "###.##") & " um"
   Case "DOWN"
    NewPosition = UmToBin(YCurrent - Stepp / 1000) + 29
    Call Send(0, Piezo_Number, "O2+" & Str$(NewPosition) & Chr$(13), DABend)
    label_SentY.Caption = Format(YCurrent - Stepp / 1000, "###.##") & " um"
   Case "RIGHT"
    NewPosition = UmToBin(xCurrent + Stepp / 1000) + 62'
    Call Send(0, Piezo_Number, "O1+" & Str$(NewPosition) & Chr$(13), DABend)
    label_SentX.Caption = Format(xCurrent + Stepp / 1000, "###.##") & " um"
   Case "LEFT"
    NewPosition = UmToBin(xCurrent - Stepp / 1000) + 62
    Call Send(0, Piezo_Number, "O1+" & Str$(NewPosition) & Chr$(13), DABend)
    label_SentX.Caption = Format(xCurrent - Stepp / 1000, "###.##") & " um"
 End Select
 DoEvents
 'Delay (PD)
End Sub

'************************** Enabables/Disables joystick handling
Sub check_Joystick_Click (Value As Integer)
  If Value = True Then
    timer_joyX.Enabled = True
    timer_joyY.Enabled = True
     Else
    timer_joyX.Enabled = False
    timer_joyY.Enabled = False
  End If
End Sub

'********************** Decides if x-step should equal y-step
Sub check_SameStep_Click (Value As Integer)
  If Value = True Then
      combo_Step(1).Text = combo_Step(0).Text
  End If
End Sub

'************************** Controls the timer that reads output from the piezo
Sub check_Start_Click (Value As Integer)
 If Value = True Then
   timer_Read.Enabled = True
  Else
   timer_Read.Enabled = False
 End If
End Sub

'********************** Enables/disables arrow handling
Sub check_UseArrow_Click (Value As Integer)
 If Value = True Then
   fArrows.Show
  Else
```

```
    fArrows.Hide
  End If
End Sub

'*************** Controls the positioning with the keys
Sub check_UseArrow_KeyDown (KeyCode As Integer, Shift As Integer)
  If KeyCode = KEY_LEFT Then spin2_SpinDown'ChangePosition ("LEFT")
  If KeyCode = KEY_UP Then spin1_SpinUp
  If KeyCode = KEY_RIGHT Then spin2_SpinUp
  If KeyCode = KEY_DOWN Then spin1_SpinDown
End Sub

'************************** If the program logs in a new day than last login, reset rec. FileNumber
Sub CheckDate ()
  If gSavedDate <> Date$ Then gNbrOfSavings = 0
End Sub

'********************* Get's the delaytime (between two readings when scanning)
Sub combo_Delay_Change ()
  DelayTime = Val(combo_Delay.Text)
  UpdateScanTime
End Sub

'********************** Records the chosen delaytime and update calculated scantime
Sub combo_Delay_Click ()
  DelayTime = Val(combo_Delay.Text)
  UpdateScanTime
End Sub


'********************* Get's the dimension of the scan
Sub combo_Pixel_Change ()
  NbrOfPixels = Val(combo_Pixel.Text)
  UpdateScanTime
End Sub

'*************** Records chosen dimensions of the scan and updates calculated scantime
Sub combo_Pixel_Click ()
  NbrOfPixels = Val(combo_Pixel.Text)
  UpdateScanTime
End Sub

'************************** If samestep is set then set steps to equal values
Sub combo_Step_Change (index As Integer)
  If check_SameStep.Value = True Then
    If index = 0 Then
      combo_Step(1).Text = combo_Step(0).Text
       Else
      combo_Step(0).Text = combo_Step(1).Text
    End If
  End If
  XStep = CInt(Val(combo_Step(0).Text) * CompX)
  YStep = CInt(Val(combo_Step(1).Text) * CompY)
End Sub

'***************If samestep is set then set steps to equal values
Sub combo_Step_Click (index As Integer)
  If check_SameStep.Value = True Then
    If index = 0 Then
      combo_Step(1).Text = combo_Step(0).Text
       Else
      combo_Step(0).Text = combo_Step(1)
    End If
  End If
  XStep = CInt(Val(combo_Step(0).Text) * CompX)
  YStep = CInt(Val(combo_Step(1).Text) * CompY)
End Sub
```

```
'****************************** Put wanted device into remote operation
Sub Connect (Dev As String)
Dim X As Integer
Dim channel As String
 Select Case Dev
   Case "Piezo"  '************** Leave channel 3 in local mode
     For X = 1 To 2
         channel = Str$(X)
         Call Send(0, Piezo_Number, "R" & channel & "=1" & Chr$(13), DABend)
         Call ErrorTest(4)
     Next X
   Case "LockIn"
     Call Send(0, Lock_Number, "REMOTE 1" & Chr$(13), DABend)
     Call ErrorTest(5)
     WaitForDevice
 End Select
End Sub


'******************************* Put wanted device into local operation
Sub Disconnect (Dev As String)
Dim X As Integer
Dim channel As String
 Select Case Dev
   Case "Piezo"
     For X = 1 To 3
         channel = Str$(X)
         Call Send(0, Piezo_Number, "R" & channel & "=0" & Chr$(13), DABend)
         Call ErrorTest(4)
     Next X
   Case "LockIn"
     Call Send(0, Lock_Number, "REMOTE 0" & Chr$(13), DABend)
     Call ErrorTest(4)
 End Select
End Sub

'******************** Get reading from piezo and display it (Index=wanted channel)
Sub DisplayReadings (index As Integer)
Dim range, buffer As String
Dim spaces As String
Dim display As Single
  buffer = Space(10)
  Call Send(0, Piezo_Number, "I" & Str$(index) & Chr$(13), DABend)
  delay (PD)
  Call Receive(0, Piezo_Number, buffer, STOPend)
  range = Mid$(buffer, 2, 1)
  If Mid$(buffer, 3, 1) = "-" Then spaces = "" Else spaces = " "
  buffer = Right$(buffer, 8)
  If range = "H" Then
     display = (Val(buffer) / 32768) * 200
    Else
     display = (Val(buffer) / 32768) * 20
  End If
  Select Case index
   Case 1: label_XDisplay.Caption = spaces & Format(display, "000.00") & " um"
   Case 2: label_YDisplay.Caption = spaces & Format(display, "000.00") & " um"
  End Select
End Sub

'************ Initialise the program and form when the window is loading into memory
Sub Form_Load ()
Dim X, x2 As Long
Dim DelayTime As String
'******************************** Initialize combolists
 combo_Pixel.AddItem "1x1": combo_Pixel.AddItem "2x2": combo_Pixel.AddItem "4x4"
 combo_Pixel.AddItem "8x8": combo_Pixel.AddItem "12x12": combo_Pixel.AddItem "16x16"
```

```
 combo_Pixel.AddItem "20x20": combo_Pixel.AddItem "24x24": combo_Pixel.AddItem "32x32"
 combo_Pixel.AddItem "40x40": combo_Pixel.AddItem "60x60": combo_Pixel.AddItem "80x80"
 combo_Pixel.AddItem "100x100": combo_Pixel.AddItem "140x140": combo_Pixel.AddItem "200x200"
 combo_Pixel.ListIndex = 5
  For X = 0 To 1
   combo_Step(X).AddItem "3 nm": combo_Step(X).AddItem "6 nm": combo_Step(X).AddItem "9 nm"
   combo_Step(X).AddItem "25 nm": combo_Step(X).AddItem "50 nm": combo_Step(X).AddItem "75 nm"
   combo_Step(X).AddItem "100 nm": combo_Step(X).AddItem "150 nm": combo_Step(X).AddItem "250 nm"
   combo_Step(X).AddItem "500 nm": combo_Step(X).AddItem "1000 nm"
  Next X
 combo_Step(0).ListIndex = 4: combo_Step(1).ListIndex = 4
 combo_Delay.AddItem " 0 ms": combo_Delay.ListIndex = 0
 x2 = 1
  For X = 1 To 15
     DelayTime = Str(x2) & " ms"
     combo_Delay.AddItem DelayTime
     x2 = x2 * 2
  Next X
'************************** Initialize global variables
  Piezo_Number = 12
  Lock_Number = 10
  ErrorCounter = 0
  HardChecking = True
  SquareScan = True
  Scantype = 1
  XStep = 50
  YStep = 50
  CompX = 1
  CompY = 1
  OldX = 0
  OldY = 0

'************************** Get and check default parameters
  Call ReadDefaults
  Call ReadDate
  Call CheckDate
End Sub

'************* Saves the current date when the program is quitting
Sub Form_Unload (Cancel As Integer)
  Call SaveDate
End Sub

'Transforms the joystick position into the speed the positioner will move
Function GetInterval (TheForce As Long) As Integer
  If TheForce > 10000 And TheForce < 15000 Then GetInterval = 500
  If TheForce > 15000 And TheForce < 20000 Then GetInterval = 250
  If TheForce > 20000 And TheForce < 25000 Then GetInterval = 150
  If TheForce > 25000 And TheForce < 30000 Then GetInterval = 50
  If TheForce > 30000 Then GetInterval = 10
End Function

'*************** Reads the piezo and returns the position in um (as type single)
Function GetUm (index As Integer) As Single
Dim range, buffer As String
Dim spaces As String
Dim display As Single
  buffer = Space(10)
  Call Send(0, Piezo_Number, "I" & Str$(index) & Chr$(13), DABend)
  delay (PD)
  Call Receive(0, Piezo_Number, buffer, STOPend)
  range = Mid$(buffer, 2, 1)
  If Mid$(buffer, 3, 1) = "-" Then spaces = "" Else spaces = " "
  buffer = Right$(buffer, 8)
  If range = "H" Then
     GetUm = (Val(buffer) / 32768) * 200
    Else
```

```
    GetUm = (Val(buffer) / 32768) * 20
  End If
End Function

'********************** Initialize the GPIB-communication
Sub InitGPIB ()
  '********** Set's interface clear on the GPIB-card
  Call SendIFC(0)
    Call ErrorTest(6)
  '********** Sends device clear to ALL instruments
  Call DevClear(0, NOADDR)
    Call ErrorTest(7)
End Sub

'******************** Choose to connect or disconnect lock-in
Sub label_ConnectLockIn_Click ()
 If c_ConnectLockIn(0).Visible = True Then
   c_ConnectLockIn_Click (0)
    Else
   c_ConnectLockIn_Click (1)
End If
End Sub

'******************** Choose to connect or disconnect piezo
Sub label_ConnectPiezo_Click ()
 If c_ConnectPiezo(0).Visible = True Then
   c_ConnectPiezo_click (0)
    Else
   c_ConnectPiezo_click (1)
 End If
End Sub

'********************************* Calls for the calibration form
Sub menu_CalibrateJoystick_Click ()
  fCalibrate.Show
End Sub

'************************ Set's errormode to hard
Sub menu_ErrorHard_Click ()
  If menu_ErrorHard.Checked = False Then
    HardChecking = True
    menu_ErrorSoft.Checked = False
    menu_ErrorHard.Checked = True
  End If
End Sub

'********************** Set's errormode to soft
Sub menu_ErrorSoft_Click ()
  If menu_ErrorSoft.Checked = False Then
    HardChecking = False
    menu_ErrorSoft.Checked = True
    menu_ErrorHard.Checked = False
  End If
End Sub

'******************** Exit's program
Sub menu_Exit_Click ()
  Unload fMain
  End
End Sub

'************************ Call for helpfile on MAIN
Sub menu_Help_Click ()
  Call HELP("MAIN")
End Sub
```

```
'********************** Call for systeminfo form
Sub menu_SystemInfo_Click ()
  f_Resource.Show
End Sub

'********************** Move's from start to finish (Input in um) in 1 um steps
Sub MoveTo (Start As Single, Finish As Single, Direction As String)
Dim Position, TheStep As Single
  Position = Start
  If Direction = "DOWN" Or Direction = "LEFT" Then TheStep = -1000 Else TheStep = 1000
                                                          '********** Go with 1 um step
  Do While Abs(Position - Finish) > 1
    Position = Position + TheStep / 1000
    Call ChangePosition(Direction, Abs(TheStep))
  Loop
End Sub

'**********************' Controls if the piezo is going out of it's range
Function OutOfRange (xCurrent As Single, YCurrent As Single, Stepp As Single, Direction As String) As
Integer
Dim X
   OutOfRange = False
   If (((xCurrent + Stepp / 1000) > 199) And Direction = "RIGHT") Or (((YCurrent + Stepp / 1000) > 199) And
Direction = "UP") Or (((xCurrent - Stepp / 1000) < 0) And Direction = "LEFT") Or (((YCurrent - Stepp / 1000)
< 0) And Direction = "DOWN") Then
     OutOfRange = True
     Beep
     If (xCurrent + Stepp / 1000 > 199) Then
        Call Send(0, Piezo_Number, "O1+" & 65000 & Chr$(13), DABend)
        label_SentX.Caption = "199 um"

       ElseIf (xCurrent - Stepp / 1000) < 0 Then
        Call Send(0, Piezo_Number, "O1+" & 62 & Chr$(13), DABend)
        label_SentX.Caption = "Stop that !"

       ElseIf (YCurrent + Stepp / 1000) > 199 Then
        Call Send(0, Piezo_Number, "O2+" & 65000 & Chr$(13), DABend)
        label_SentY.Caption = "199 um"

       ElseIf (YCurrent - Stepp / 1000) < 0 Then
        Call Send(0, Piezo_Number, "O2+" & 29 & Chr$(13), DABend)
        label_SentY.Caption = "Stop that!"
     End If
   End If
End Function

'******************** Position down
Sub spin1_SpinDown ()
  Call ChangePosition("DOWN", YStep)
  text_SetY.Text = label_SentY.Caption
End Sub

'********************** Position up
Sub spin1_SpinUp ()
  Call ChangePosition("UP", YStep)
  text_SetY.Text = label_SentY.Caption
End Sub

'************************ Position left
Sub spin2_SpinDown ()
  Call ChangePosition("LEFT", XStep)
  text_SetX.Text = label_SentX.Caption
End Sub
```

```
'****************** Position right
Sub spin2_SpinUp ()
  Call ChangePosition("RIGHT", XStep)
  text_SetX.Text = label_SentX.Caption
End Sub

'******************** Set wanted compensation if <CR> is pressed in list box
Sub text_CompX_KeyPress (KeyAscii As Integer)
  If KeyAscii = 13 Then
    CompX = Val(text_CompX.Text) / 100
    XStep = CInt(Val(combo_Step(0).Text) * CompX)
    KeyAscii = 0
  End If
End Sub

'******************** Set wanted compensation if <CR> is pressed in list box
Sub text_CompY_KeyPress (KeyAscii As Integer)
  If KeyAscii = 13 Then
    CompY = Val(text_CompY.Text) / 100
    YStep = CInt(Val(combo_Step(1).Text) * CompY)
    KeyAscii = 0
  End If
End Sub

'************************** Get's and sends wanted x-position
Sub text_SetX_KeyPress (KeyAscii As Integer)
Dim Finish As Single
Dim Start As Single
Dim Direction As String
If KeyAscii = 13 Then
    Finish = Val(text_SetX.Text)          '************ Get wanted position
    If Finish > 199 Then GoTo ending2
    Start = OldX                          '************ Get start position
    If (Start - Finish) > 0 Then Direction = "LEFT" Else Direction = "RIGHT"
    Call MoveTo(Start, Finish, Direction)  '************ Move form start to finish in small steps
    XAsBin = UmToBin(Finish) + 62
    Call Send(0, Piezo_Number, "O1+" & Str$(XAsBin) & Chr$(13), DABend) '******* Finetune the position
    label_SentX.Caption = text_SetX.Text
    OldX = Finish
    KeyAscii = 0
  End If
  Exit Sub
ending2:
MsgBox " Maximum input is 199 um", , "Info"
Exit Sub
End Sub

'************************** Get's and sends wanted y-position
Sub text_SetY_KeyPress (KeyAscii As Integer)
Dim Finish As Single
Dim Start As Single
Dim Direction As String
If KeyAscii = 13 Then
    Finish = Val(text_SetY.Text)          '************ Get wanted position
    If Finish > 199 Then GoTo ending3
    Start = OldY                          '************ Get start position
    If (Start - Finish) > 0 Then Direction = "DOWN" Else Direction = "UP"
    Call MoveTo(Start, Finish, Direction)  '************ Move from start to finish in small steps
    YAsBin = UmToBin(Finish) + 29
    Call Send(0, Piezo_Number, "O2+" & Str$(YAsBin) & Chr$(13), DABend)
    label_SentY.Caption = text_SetY.Text
    OldY = Finish
    KeyAscii = 0
  End If
  Exit Sub
ending3:
MsgBox " Maximum input is 199 um", , "Info"
```

```
Exit Sub
End Sub

'********** Reads current X-position of the joystick and sends to piezo
Sub timer_JoyX_Timer ()
Dim dummy, X As Integer
Dim joystick As JoyInfo
Dim xpos As Long
Dim xforce As Long
Dim TheXStep As Single
   dummy = JoyGetPos(0, joystick)
   If joystick.button = 1 Then TheXStep = 2000 Else TheXStep = 100
   xpos = joystick.xpos
   xforce = 32768 - Abs(xpos)
   If xforce > 10000 Then
    timer_joyX.Interval = GetInterval(xforce)
     If joystick.xpos < 0 Then
          Call ChangePosition("RIGHT", TheXStep)
           Else
          Call ChangePosition("LEFT", TheXStep)
     End If
   End If
   If joystick.button = 2 Or joystick.button = 2 Then
    For X = 1 To 5: Beep: delay (100): Next X
    fScanning.Show
    fScanning.c_Scan.Value = True
   End If
   text_SetX.Text = label_SentX.Caption
   OldX = Val(text_SetX.Text)
End Sub

'********** Reads current Y-position of the joystick and sends to piezo
Sub timer_JoyY_Timer ()
Dim dummy, X As Integer
Dim joystick As JoyInfo
Dim ypos As Long
Dim yForce As Long
Dim TheYStep As Single
   dummy = JoyGetPos(0, joystick)
   If joystick.button = 1 Then TheYStep = 2000 Else TheYStep = 100
   ypos = joystick.ypos
   yForce = 32768 - Abs(ypos)
   If yForce > 10000 Then
   timer_joyY.Interval = GetInterval(yForce)
    If joystick.ypos < 0 Then
          Call ChangePosition("DOWN", TheYStep)
           Else
          Call ChangePosition("UP", TheYStep)
     End If
   End If
   text_SetY.Text = label_SentY.Caption
   OldY = Val(text_SetY.Text)
End Sub

'********** Reads current position every 1/2 second
Sub timer_Read_Timer ()
Dim X As Integer
 For X = 1 To 2
  Call DisplayReadings(X)
 Next X
End Sub

'********** Calculates and displays the approximative scantime
Sub UpdateScanTime ()
Dim Timescan As Single
Dim min, sec As Integer
  If Scantype = 1 Then
```

```
  Timescan = ((.131 + DelayTime / 1000) * NbrOfPixels * NbrOfPixels)
 Else
  Timescan = ((.075 + DelayTime / 1000) * NbrOfPixels * NbrOfPixels)
 End If
 min = Timescan \ 60
 sec = Format(Timescan Mod 60, "00")
 label_ScanTime.Caption = "ScanTime :" & min & "min " & sec & "s   (approx)"
End Sub
```

****

B) THE SCANNING FORM Contains the code for the scanning form (GRID1.frm)

****

```
Option Explicit
Dim Zoom As Integer
Dim ZoomDegree As Single
Dim ZoomChoice As Single
Dim OldZoomChoice As Integer
Dim Halt As Integer
Dim Pause As Integer
Dim MaxValue As Integer

'------------------ Set gridlines on/off
Sub c_GridLines_Click ()
Static toggle As Integer
 toggle = Not (toggle)
 If toggle = True Then
   grid1.GridLines = False
    Else
   grid1.GridLines = True
 End If
End Sub

'------------------ Make pause in scanning
Sub c_Pause_Click ()
Static toggle As Integer
  If toggle = True Then
     toggle = False
     Pause = True
     c_Pause.Caption = "Start"
    Else
     toggle = True
     Pause = False
     c_Pause.Caption = "pause"
  End If
End Sub

'----------------- Draw picture with range 255=Maxvalue
Sub c_PubPaint_Click ()
'Dim dummy
'  On Error GoTo errorfunc
'  dummy = Shell("c:\windows\pubpb\pubpb.exe", 4)
'  Exit Sub
'errorfunc:
  'MsgBox "Already open", , "Pub Paint"
  'Resume Next
  DrawPixelPicture
End Sub

'----------------- Start an appropriate scanning
Sub c_Scan_Click ()
Dim retur As String
Dim Time1 As Long
```

```
  Raise
  fScanning.MousePointer = 11
  If SquareScan = True Then
     Time1 = Timer
     If ScanType = 3 Then Call MoveToStartPosition
     If ScanType = 1 Then
        Call SquareScanTyp1
        Else
        Call SquareScanTyp2
     End If
      Else
     retur = InputBox$("Which direction for Line Scan ?" & Chr$(10) & "  1: x-direction" & Chr$(10) & "  2: y-
direction", "Line Scan")
     Time1 = Timer
     If retur = "1" Then
       Call LineScanning("x")
       ElseIf retur = "2" Then
       Call LineScanning("y")
       ElseIf retur <> "" Then
         MsgBox "Wrong input ,try again"
     End If
  End If
  fScanning.MousePointer = 0
  TimePassed = Timer - Time1
  label_Time.Caption = "Time: " & TimePassed & " s"
  Low
End Sub

'---------- Set Halt for stopping scanning
Sub c_Stop_Click ()
  Halt = True
End Sub

'---------- Call for TestMenu
Sub c_Test_Click ()
  PopupMenu menu_Testmenu
End Sub

'---------- Open scanType form
Sub c_TypeOfScan_Click ()
   fScanType.Show
End Sub

'---------- Set form to Zoom mode=on/off
Sub c_Zoom_Click ()
Static toggle As Integer
 toggle = Not (toggle)
 If toggle = True Then
    Zoom = True
    fScanning.MousePointer = 5
  Else
    Zoom = False
    fScanning.MousePointer = 0
  End If
End Sub

'---------- Clear the grid
Sub ClearGrid ()
Dim x, y As Integer
  For x = 1 To NbrOfPixels
    grid1.Row = x
    For y = 1 To NbrOfPixels
      grid1.Col = y
      grid1.Picture = LoadPicture()
    Next y
  Next x
End Sub
```

```
'---------- Dummybutton for communicatin with the open file form
Sub Command1_Click ()
  Call ScaleGrid(NbrOfPixels, 1)
  DrawPixelPicture
End Sub

'---------- Draws the array PixelPicture (0-255) on the screen
Sub DrawPixelPicture ()
Dim rad, kol, x As Integer
On Error GoTo ErrorHandling3
  For rad = 1 To NbrOfPixels
  grid1.Row = rad
    For kol = 1 To NbrOfPixels
      x = x + 1
      grid1.Col = kol
      grid1.Picture = picclip1.GraphicCell(CInt(63 * PixelPicture(x) / 255))
    Next kol
  Next rad
  Exit Sub
ErrorHandling3:
Resume Next
End Sub

'---------- Refresh grid parameters
Sub Form_Activate ()
On Error GoTo ErrorHandling4
    ReDim PixelPicture(1 To NbrOfPixels * NbrOfPixels)  '---- Dimension array that contains scandata
    ZoomDegree = 1
    Call ScaleGrid(NbrOfPixels, ZoomDegree) '---------- Remove these 2 lines if you want to disable the graphics
    Call UpDateData
    Exit Sub
ErrorHandling4:
MsgBox "Pixelpicture is out of range"
Exit Sub
End Sub

'---------- Init variables and call for drawing grid and fresh data
Sub Form_Load ()
  ZoomDegree = 1
  ZoomChoice = 1.25
  OldZoomChoice = 1
  Call ScaleGrid(NbrOfPixels, ZoomDegree)      '-------- Remove these 2 lines if you want to disable thegraphics
  Call UpDateData
End Sub

'------ Makes a movement from start -> finish with "NbrOFPixels" steps
Sub GoBackSlowly (Start As Long, Finish As Long)
Dim x  As Integer
Dim Position As Long
   Position = Start
   For x = 1 To NbrOfPixels
     Position = Position - XStep
     Call Send(0, Piezo_Number, "O1+" & Position & Chr$(13), DABend) '----- Go back one XStep
   delay (25)                                          '----- Wait 25 ms between steps
   Next x
   Call Send(0, Piezo_Number, "O1+" & Finish & Chr$(13), DABend)     '----- Move to exact startposition
   delay (100)                                         '----- Wait 100ms at start to reduce vibrations when reversing
direction
End Sub

'---------- Display chosen pixel
Sub grid1_click ()
Dim x, y As Integer
 x = grid1.Col
 y = grid1.Row
 text_Position.Text = "Col:" & x & " ,Row: " & y
End Sub
```

```
'_____ When mousebutton is pressed over grid and zoom is chosen
Sub Grid1_MouseDown (button As Integer, Shift As Integer, x As Single, y As Single)
Dim size, cellSize As Integer
  If Zoom = True Then
    If button = 1 Then
      ZoomDegree = ZoomDegree * ZoomChoice
       ElseIf button = 2 Then
      ZoomDegree = ZoomDegree * (1 / ZoomChoice)
    End If
    If ZoomChoice = 1 Then ZoomDegree = 1
    size = grid1.Width - 400                    ' Compensate for drawbar
    cellSize = Int(ZoomDegree * (size / (NbrOfPixels + 1)))
    If NbrOfPixels < 12 Then cellSize = Int(cellSize / 2)
    If (cellSize < (picclip1.Height * 11.4)) And (cellSize > 2) Then ' Kontrollera så att bitmapen räckerv
       Call ScaleGrid(NbrOfPixels, ZoomDegree)
      Else
      MsgBox "Sorry can't zoom to that degree", 48, "Zooming"
    End If
  End If 'Zoom=true
End Sub

'---------- Make a line scan in input direction      NOTE: Important sub !!!!!
Sub LineScanning (direction As String)
Dim x, y, k As Long
Dim xCurrent, yCurrent, xStepBin, yStepBin As Long
Dim buffer As String
Dim Nbr, convert As Integer
  ReDim PixelPicture(1 To NbrOfPixels)
  buffer = Space(10)
  '---------- Get current position and calculate scanning step
  xCurrent = XasBin
  yCurrent = YAsBin
  xStepBin = UmToBin(XStep / 1000)                    '____ Range: 0-65535
  yStepBin = UmToBin(yStep / 1000)                    '____ Range: 0-65535
  grid1.Row = 1: grid1.Col = 1
  '---------- Scan in "y"-direction
  If direction = "y" Then
     For y = 1 To NbrOfPixels
        grid1.Row = y
        Nbr = Nbr + 1
        yCurrent = yCurrent + yStepBin                '------ Update position
        Call Send(0, Piezo_Number, "O2+" & yCurrent & Chr$(13), DABend)  '------ Move to position
        Call delay(DelayTime)                         '------ DelayTime
        Call Send(0, lock_number, "*" & Chr$(13), DABend)  '------ High-Speed mode
        Call Receive(0, lock_number, buffer, STOPend)  '------ Get lock-in output
        PixelPicture(Nbr) = Val(buffer)               '------ Store in global array
        convert = Abs(CInt(Val(buffer) * 63 / 15000)) + 1  '------ Convert to range: 0-64
        If convert < 64 Then grid1.Picture = picclip1.GraphicCell(convert)  '------ Draw pixel on form if
                                                          '-not overload
        DoEvents
        If y = 600 Then
           Call Send(0, Piezo_Number, "O2+" & (yCurrent + 2000) & Chr$(13), DABend)
                                                      '------ Move to position
           delay (60000)
           Call Send(0, Piezo_Number, "O2+" & yCurrent & Chr$(13), DABend) '------ Move to position
        End If
     Next y
  Else
    '---------- Scan in "x"-direction
     For x = 1 To NbrOfPixels
        grid1.Col = x
        Nbr = Nbr + 1
        xCurrent = xCurrent + xStepBin                '------ Update position
        Call Send(0, Piezo_Number, "O1+" & xCurrent & Chr$(13), DABend)  '------ Move to position
        Call delay(DelayTime)                         '------ DelayTime
```

```
        Call Send(0, lock_number, "*" & Chr$(13), DABend)          '------ High-Speed mode
        Call Receive(0, lock_number, buffer, STOPend)              '------ Get lock-in output
        PixelPicture(Nbr) = Val(buffer)                            '------ Store in global array
        convert = Abs(CInt(Val(buffer) * 64 / 15000))              '------ Convert to range: 0-64
        If convert < 64 Then grid1.Picture = picclip1.GraphicCell(convert)   '------ Draw pixel on form if not
                                                                   '_____ overload
        DoEvents
      Next x
    End If
    Call Send(0, lock_number, "OUT" & Chr$(13), DABend)            '--- Disconnect High-Speed mode
End Sub

'---------- Low the grid
Sub Low ()
  grid1.Top = grid1.Top - 20
  grid1.Left = grid1.Left - 10
  panel_Picture.BevelInner = 2
  panel_Picture.BevelOuter = 2
End Sub

'-------------------- Make a grayscale test on grid
Sub MakeGrayScaleTest ()
Dim kol, rad As Integer
Dim GrayNbr As Integer
Dim x As Integer
  GrayNbr = 0
  For x = 1 To (NbrOfPixels * NbrOfPixels)
    GrayNbr = GrayNbr + 1: If GrayNbr = 63 Then GrayNbr = 0
    PixelPicture(x) = CInt(GrayNbr * 255 / 63)
  Next x
  DrawPixelPicture
End Sub


'---------------------- Make a randomtest on grid
Sub MakeRandomTest ()
Dim GrayNbr, x As Integer
  Randomize
  For x = 1 To NbrOfPixels * NbrOfPixels
    GrayNbr = 255 * Rnd
    PixelPicture(x) = GrayNbr
  Next x
  DrawPixelPicture
End Sub

'---------- Set grid color
Sub menu_Backcolor_Click (index As Integer)
Dim colorValue As Integer
Select Case index
  Case 1: colorValue = 7
  Case 2: colorValue = 8
  Case 3: colorValue = 1
  Case 4: colorValue = 0
End Select
  panel_Picture.BackColor = QBColor(colorValue)
End Sub

'---------- Make call for clearing grid
Sub menu_ClearGrid_Click ()
  Call ClearGrid
End Sub

'---------- Set grid back color
Sub menu_GridColor_Click (index As Integer)
Dim colorValue As Integer
Select Case index
  Case 1: colorValue = 7
```

```
  Case 2: colorValue = 8
  Case 3: colorValue = 1
  Case 4: colorValue = 0
  Case 5: colorValue = 15
End Select
 grid1.BackColor = QBColor(colorValue)
End Sub

'---------- Call for help on subject "scanning"
Sub menu_Help_Click ()
  Call HELP("SCANNING")
End Sub

'--------------- Open OpenFile form
Sub menu_OpenScan_Click ()
  fOpenFile.Show
End Sub

'--------- Open SaveAs form
Sub menu_SaveAs_Click ()
   fSaveAs.Show
End Sub

'---------- Call for testing grayscale
Sub menu_TestGrayScale_Click ()
 Call MakeGrayScaleTest
End Sub

'--------------- Call for randomtesting
Sub menu_TestRandom_Click ()
   Call MakeRandomTest
End Sub

'------------ Select ZoomDegree
Sub menu_Zoom_Click (index As Integer)
 menu_Zoom(index).Checked = True
 Select Case index
    Case 0: ZoomChoice = 1.1
    Case 1: ZoomChoice = 1.25
    Case 2: ZoomChoice = 1.5
    Case 3: ZoomChoice = 1.75
    Case 4: ZoomChoice = 2
    Case 5: ZoomChoice = 1
 End Select
 If OldZoomChoice <> index Then menu_Zoom(OldZoomChoice).Checked = False
 OldZoomChoice = index
End Sub

'---------------- Move the starting point for the scan to upper-left
Sub MoveToStartPosition ()
 XasBin = XasBin - (UmToBin(XStep / 1000) * Int(NbrOfPixels / 2))
 YAsBin = YAsBin - (UmToBin(XStep / 1000) * Int(NbrOfPixels / 2))
 If XasBin < 0 Or YAsBin < 0 Or XasBin > 65535 Or YAsBin > 65535 Then
   MsgBox "You will scan out of the range of the piezo ! I will try a type 2 scan instead."
   Exit Sub
 End If
 Call Send(0, Piezo_Number, "O1+" & XasBin & Chr$(13), DABend)
 Call Send(0, Piezo_Number, "O2+" & YAsBin & Chr$(13), DABend)
End Sub

'------------------ Raise grid
Sub Raise ()
  panel_Picture.BevelInner = 1
  panel_Picture.BevelOuter = 1
  grid1.Left = grid1.Left + 10
  grid1.Top = grid1.Top + 20
End Sub
```

```
'------------------------ Scale numbers and PixelSize on grid
Sub ScaleGrid (NbrOfPixels As Integer, ZoomGrade As Single)
Dim size, cellSize, x, y As Integer
On Error GoTo ErrorHandler3
 '---------- Calculate the properties of the grid
 grid1.Rows = NbrOfPixels + 1                          '---- Set number of rows
 grid1.Cols = NbrOfPixels + 1                          '---- Set number of columns
 size = grid1.Width - 400                              '---- Compensate for the drawbar
 cellSize = Int(ZoomGrade * (size / (NbrOfPixels + 1))) '---- Calculate cell(pixel) size
 If NbrOfPixels < 12 Then cellSize = Int(cellSize / 2) '---- Make smaller grid for small scannings
 If cellSize > 140 Then grid1.FontSize = 6             '---- Scale font
 If cellSize < 140 Then grid1.FontSize = 3
 '---------- Create and draw the grid on screen
 grid1.Col = 0
 For x = 0 To NbrOfPixels
    grid1.Row = x
    grid1.Text = x                                     '---- Number rows
    grid1.RowHeight(x) = cellSize                      '---- Scale rows
 Next x
 grid1.Row = 0
 For y = 0 To NbrOfPixels
    grid1.Col = y
    grid1.Text = y                                     '---- Number columns
    grid1.ColWidth(y) = cellSize                       '---- Scale columns
 Next y
TheEnd:
Exit Sub
ErrorHandler3:
MsgBox "Bad values when scaling grid,closing scaling", , "Graphic problem"
GoTo TheEnd
End Sub

'-------------------- Change picture contrast (Note: Affects only screen, not actual scandata)
Sub scroll_Contrast_Change ()
Dim kol, x, rad, GrayNbr, ContrastIndex As Integer
Static slaskPixel() As Integer
 ReDim slaskPixel(1 To NbrOfPixels * NbrOfPixels)
 fScanning.MousePointer = 11
 '---------------------- Display selected contrast
 ContrastIndex = scroll_Contrast.Value
 label_ContrastPercent.Caption = Int((ContrastIndex + 32) / .635) & "%"
 '---------------------- Make copy of inscanned data
 For x = 1 To (NbrOfPixels * NbrOfPixels)
  slaskPixel(x) = CInt(PixelPicture(x) * 63 / 255)
 Next x
 '---------------------- Evaluate every pixel
 For x = 1 To (NbrOfPixels * NbrOfPixels)
 If ContrastIndex > 0 Then '---------------------- For increasing contrast
     If (slaskPixel(x) <= 32) And (slaskPixel(x) >= ContrastIndex) Then
        slaskPixel(x) = slaskPixel(x) - ContrastIndex
       ElseIf slaskPixel(x) < ContrastIndex Then
        slaskPixel(x) = 0
      End If
     If (slaskPixel(x) > 32) And ((63 - slaskPixel(x)) >= ContrastIndex) Then
        slaskPixel(x) = slaskPixel(x) + ContrastIndex
       ElseIf ((63 - slaskPixel(x)) < ContrastIndex) Then
        slaskPixel(x) = 63
     End If
 End If
 If ContrastIndex < 0 Then '---------------------- For decreasing contrast
     If (slaskPixel(x) < 32) And (Abs(32 - slaskPixel(x)) >= ContrastIndex) Then
        slaskPixel(x) = slaskPixel(x) - ContrastIndex
       ElseIf (slaskPixel(x) < 32) And (Abs(32 - slaskPixel(x)) < Abs(ContrastIndex)) Then
        slaskPixel(x) = 32
     End If
     If (slaskPixel(x) > 32) And (Abs(32 - slaskPixel(x)) >= Abs(ContrastIndex)) Then
```

```
        slaskPixel(x) = slaskPixel(x) + ContrastIndex
       ElseIf (slaskPixel(x) > 32) And (Abs(32 - slaskPixel(x)) < Abs(ContrastIndex)) Then
        slaskPixel(x) = 32
     End If
  End If
  Next x
  '---------------------- Draw picture with new contrast
  x = 0
  For rad = 1 To NbrOfPixels
  grid1.Row = rad
   For kol = 1 To NbrOfPixels
     x = x + 1
     grid1.Col = kol
     grid1.Picture = picclip1.GraphicCell(slaskPixel(x))
   Next kol
  Next rad
  fScanning.MousePointer = 0
End Sub

'------------------- Make a SquareScanning of type 1          NOTE: Important sub !!!!!
Sub SquareScanTyp1 ()
Dim x, y, k As Long
Dim xStepBin, yStepBin As Long
Dim xCurrent As Long
Dim yCurrent As Long
Dim buffer As String
Dim Nbr, convert As Integer
Dim ERRO As Integer
  ERRO = False
  On Error GoTo ErrorHandling
  ReDim PixelPicture(1 To NbrOfPixels * NbrOfPixels)          '---- Dimension array that contains
scandata
  Halt = False: Pause = False                                 '---- Init startparameters for scan
  '------------------ Get current position and calculate scanning step
  buffer = Space(10)
  xCurrent = XasBin
  yCurrent = YAsBin
  xStepBin = UmToBin(XStep / 1000)                            '----- Range:0-65535
  yStepBin = UmToBin(yStep / 1000)                            '----- Range:0-65535
  Call Send(0, Piezo_Number, "O1+" & xCurrent & Chr$(13), DABend)
  Call Send(0, Piezo_Number, "O2+" & yCurrent & Chr$(13), DABend)
  '----------------- Begin scanning
  For y = 1 To NbrOfPixels
    grid1.Row = y
    Call Send(0, Piezo_Number, "O2+" & yCurrent & Chr$(13), DABend)
    yCurrent = yCurrent + yStepBin                            '----- Update y-position
    For x = 1 To NbrOfPixels
      Nbr = Nbr + 1
      grid1.Col = x
      xCurrent = xCurrent + xStepBin                          '----- Update x-position
      Call Send(0, Piezo_Number, "O1+" & xCurrent & Chr$(13), DABend) '----- Move to new position
      delay (DelayTime)                                       '----- DelayTime
      Call Send(0, lock_number, "*" & Chr$(13), DABend)        '----- Set High-Speed mode
      Call Receive(0, lock_number, buffer, STOPend)           '----- Get lock-in output
      text_Position.Text = Nbr
      PixelPicture(Nbr) = Val(buffer)                  '----- Store in global array PixelPicture()
      convert = CInt(Val(buffer) * 63 / 15000)         '----- Convert to range: 0-64 for drawing
      grid1.Picture = picclip1.GraphicCell(convert)    '----- Draw pixel on form
    Next x
    DoEvents                                           '----- Get events in queue if any
    If Halt = True Then Exit For                       '----- Halt scan if stopbutton pressed
    If Pause = True Then                               '----- Pause scan if pausebutton pressed
     Do: DoEvents: Loop While Pause = True
    End If
    Call GoBackSlowly(xCurrent, XasBin)          '----- Go from current x to start (XAsBin) slowly
    xCurrent = XasBin
```

```
  Next y
  TransformTo255                                    '----- Transform PixelPicture to 0-255
  Call Send(0, lock_number, "OUT" & Chr$(13), DABend)   '----- Disconnect High-Speed mode
  If ERRO = True Then MsgBox "There was an overload during scanning"
  Exit Sub
ErrorHandling:
ERRO = True
Resume Next
End Sub

'-------------------- Make a SquareScanning of type 2
Sub SquareScanTyp2 ()
Dim x, y, k As Long
Dim xCurrent, yCurrent, xStepBin, yStepBin As Long
Dim buffer As String
Dim Nbr, convert, change As Integer
Dim ERRO As Integer
  ERRO = False
  On Error GoTo ErrorHandling2
  ReDim PixelPicture(1 To (NbrOfPixels * NbrOfPixels))   '---- Dimension array that contains scandata
  Halt = False: Pause = False                            '---- Init startparameters for scan
  '------------------ Get current position and calculate scanning step
  buffer = Space(10)
  xCurrent = XasBin
  yCurrent = YAsBin
  xStepBin = UmToBin(XStep / 1000)                       '----- Range:0-65535
  yStepBin = UmToBin(yStep / 1000)                       '----- Range:0-65535
  Call Send(0, Piezo_Number, "O1+" & xCurrent & Chr$(13), DABend)
  Call Send(0, Piezo_Number, "O2+" & yCurrent & Chr$(13), DABend)
  '----------------- Begin scanning
  change = 1
  For y = 1 To NbrOfPixels
    grid1.Row = y
    yCurrent = yCurrent + yStepBin                       '----- Update y-position
    For x = 1 To NbrOfPixels
      Nbr = Nbr + c                                      '----- Increase or decrease arrayindex depending
                                                         '____on direction
      If change = 1 Then
        grid1.Col = x                                    '----- If direction = left->right
        Else
        grid1.Col = NbrOfPixels - x + 1                  '----- If direction = right->left
      End If
      xCurrent = xCurrent + (xStepBin * change)          '----- Update x-position
      Call Send(0, Piezo_Number, "O1+" & xCurrent & Chr$(13), DABend) '----- Move to new position
      delay (DelayTime)                                  '----- DelayTime
      Call Send(0, lock_number, "*" & Chr$(13), DABend)  '----- Set High-Speed mode
      Call Receive(0, lock_number, buffer, STOPend)      '----- Get lock-in output
      text_Position.Text = Nbr
      PixelPicture(Nbr) = Val(buffer)                    '----- Store in global array PixelPicture()
      convert = CInt(Val(buffer) * 63 / 15000)           '----- Convert to range: 0-64 for drawing
      grid1.Picture = picclip1.GraphicCell(convert)      '----- Draw pixel on form
    Next x
    If change = 1 Then
      change = -1:                                       '----- Reverse direction
      Nbr = Nbr + NbrOfPixels + 1                        '----- Increase arrayindex with one row
     Else
      change = 1
      Nbr = Nbr + NbrOfPixels - 1
    End If
    DoEvents                                             '----- Get events in queue if any
    If Halt = True Then Exit For                         '----- Halt scan if stopbutton pressed
    If Pause = True Then                                 '----- Pause scan if pausebutton pressed
     Do: DoEvents: Loop While Pause = True
    End If
    Call Send(0, Piezo_Number, "O2+" & yCurrent & Chr$(13), DABend)
  Next y
```

NOTE: Important sub !!!!!

Warning! This sub is somewhat complicated.
Dont give up, you will get it !!!

```
  TransformTo255                                    '----- Transform PixelPicture to 0-255
  Call Send(0, lock_number, "OUT" & Chr$(13), DABend)   '------ Disconnect High-Speed mode
  If ERRO = True Then MsgBox "There was an overload during scanning"
  Exit Sub
ErrorHandling2:
ERRO = True
Resume Next
End Sub

'---------- Transform PixelPictures data to range 0-255
'---------- Use only the range were the data actually was collected
Sub TransformTo255 ()
Dim x As Integer
Dim MaxValue As Integer
 MaxValue = 1
 For x = 1 To (NbrOfPixels * NbrOfPixels)
  If PixelPicture(x) > MaxValue Then MaxValue = PixelPicture(x)
 Next x
 For x = 1 To (NbrOfPixels * NbrOfPixels)
   PixelPicture(x) = CInt(Abs((255 / MaxValue) * (PixelPicture(x))))
 Next x
 text_Position.Text = "Max:" & MaxValue
End Sub


'__________ Calculate and show new scandata
Sub UpDateData ()
Dim sizeX, sizeY, TrueXStep, TrueYStep As Single
  label_NbrOfPixels.Caption = "NbrOfPixels:" & Str$(NbrOfPixels)
  sizeX = BinToUm(UmToBin(XStep / 1000) * NbrOfPixels)
  TrueXStep = BinToUm(UmToBin(XStep / 1000)) * 1000
  TrueYStep = BinToUm(UmToBin(yStep / 1000)) * 1000
  sizeY = BinToUm(UmToBin(yStep / 1000) * NbrOfPixels)
  If (sizeX < 10) And (sizeY < 10) Then
    label_Size.Caption = "True size:" & Chr$(10) & Format(1000 * sizeX, "####") & " x " & Format(1000 *
sizeY, "####") & " nm"
    Else
    label_Size.Caption = "True size:" & Chr$(10) & Format(sizeX, "####.#") & " x " & Format(sizeY, "###.#")
& " um"
  End If
  label_Xstep.Caption = "X-Step:" & Format(TrueXStep, "####.#") & " nm"
  label_Ystep.Caption = "Y-Step:" & Format(TrueYStep, "####.#") & " nm"
  label_ScanType.Caption = "ScanType : " & Str$(ScanType)
  label_Delay = "Delay: " & DelayTime & " ms"
End Sub
```

****

C) THE LOCK-IN CONTROLLER (Lock_3d.frm) Controls the lock-in

****

```
Option Explicit
Dim Card As Integer
Dim DeviceID As Integer
Dim TuningHz As Single
Dim Range As Integer
Dim Stopp As Integer
Dim Stepp As Single
Dim Min As Single
Dim Max As Single
Dim Autoscale As Integer
```

```
'------------------- Send's AutoOffset command to lock-in
Sub c_AutoOutputOffset_Click ()
  label_Offset.Caption = "On"
  Call Send(0, lock_number, "AXO" & Chr$(13), DABend)
  Call ErrorTest(1)
  WaitForDevice
  MsgBox ("Finished with auto offset")
End Sub

'---------------- Send's Autophasing command to lock-in
Sub c_AutoPhasing_Click ()
Dim PollByte As Integer
  text_Information.Visible = True
  text_Information.Text = "  Doing automatic phasing to maximum output , please wait....."
  Call Send(0, lock_number, "AQN" & Chr$(13), DABend)
  Call ErrorTest(1)
  WaitForDevice
  text_Information.Visible = False
End Sub

'----------------- Send's Autosensitivity command to lock-in
Sub c_AutoSensitivity_Click ()
  Dim PollByte As Integer
  combo_Sensitivity.ListIndex = 0
  text_Information.Visible = True
  text_Information.Text = "    Doing automatic search for sensitivity , please wait..... "
  Call combo_Sensitivity_Click
  WaitForDevice
  text_Information.Visible = False
End Sub

'------------ Sends Autotuning command to lock-in
Sub c_AutoTune_Click ()
  text_Tuning.Text = "Auto Tune"
  Call Send(0, lock_number, "ATS" & Chr$(13), DABend)
  ErrorTest (1)
  WaitForDevice
  MsgBox ("Finished auto tuning")
End Sub

'------------- Calls for setting offset value (false=set it manually)
Sub c_OutputOffset_click ()
  SetOffset (False)
End Sub

'---------------- Set's the slope on the lock-in
Sub c_OutputSlope_click ()
    If label_Slope.Caption = "=6 dB" Then
      Call Send(0, lock_number, "XDB 1" & Chr$(13), DABend)
      label_Slope.Caption = "=12 dB"
      gSlope = "=6 dB"  ' ---------- Set As Default
     Else
      Call Send(0, lock_number, "XDB 0" & Chr$(13), DABend)
      label_Slope.Caption = "=6 dB"
      gSlope = "=12 dB"  '---------- Set As Default
    End If
    Call ErrorTest(1)
End Sub

'----------------- Set the tuning frequency
Sub c_TuningUpDown_MouseDown (Index As Integer, Button As Integer, Shift As Integer, X As Single,
Y As Single)
   Dim potens As String
   '------------------ Disconnect reference freq
   Call Send(0, lock_number, "D1 3" & Chr$(13), DABend)
   Call ErrorTest(1)
   '------------------ Change tuning freq by selected Stepp while mousebutton is down or until exceeding range
```

```
  Stopp = False
  If Range > 2 Then potens = " kHz" Else potens = " Hz"
  Do
   If Index = 0 Then
    TuningHz = TuningHz - Stepp
   Else
    TuningHz = TuningHz + Stepp
   End If
   text_Tuning.Text = "    " & Format(TuningHz, "fixed") & potens
   Call Send(0, lock_number, "FF " & (TuningHz * Autoscale) & " " & Range & Chr$(13), DABend)
   Call ErrorTest(1)
   gTuneValue = TuningHz  'Set value
   DoEvents
  Loop Until (Stopp = True) Or (TuningHz >= Max) Or (TuningHz <= Min)
End Sub

'----------------- Stop changing tuning frequency when mouseup
Sub c_TuningUpDown_MouseUp (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y
As Single)
 Stopp = True
End Sub

'------------ Set's double freq on/off
Sub check_2F_click (Value As Integer)
  If Value <> 0 Then
   Call Send(0, lock_number, "F2F 1" & Chr$(13), DABend)
   gFx2 = True   '---------- Set As Default
   Else
   Call Send(0, lock_number, "F2F 0" & Chr$(13), DABend)
   gFx2 = False  '---------- Set As Default
  End If
  Call ErrorTest(1)
End Sub

'------------- Set's expand x 10 on/off
Sub check_Expand_click (Value As Integer)
  If Value <> 0 Then
   Call Send(0, lock_number, "EX 1 " & Chr$(13), DABend)
   gExpandx10 = True   ' ---------- Set As Default
   Else
   Call Send(0, lock_number, "EX 0 " & Chr$(13), DABend)
   gExpandx10 = False  ' ---------- Set As Default
  End If
Call ErrorTest(1)
End Sub

'-------------- Set's internal on/off
Sub check_Internal_click (Value As Integer)
  If Value <> 0 Then
    Call Send(0, lock_number, "IE 1 " & Chr$(13), DABend)
    gInternal = True ' ---------- Set As Default
    Else
    Call Send(0, lock_number, "IE 0 " & Chr$(13), DABend)
    gInternal = False  '---------- Set As Default
  End If
  Call ErrorTest(1)
End Sub
'-------------- Set's reading the lock-in on/off
Sub check_ReadOutput_Click (Value As Integer)
  If Value <> 0 Then
   timer_Display.Enabled = True
   gReadOutput = True  '---------- Set As Default
   Else
   timer_Display.Enabled = False
   gReadOutput = False '---------- Set As Default
  End If
End Sub
```

```
'---------- Select's rejection filter if changing
Sub combo_Filters_Change ()
  Dim choice As String
  Select Case combo_Filters.Text
    Case " OFF": choice = "0"
    Case "  2F": choice = "1"
    Case "   F": choice = "2"
    Case "2F+F": choice = "3"
    Case "": choice = " null "
  End Select
  If choice <> " null " Then
     Call Send(0, lock_number, "LF " & choice & Chr$(13), DABend)
     gReject = Val(choice) '---------- Set As Default
  End If
  Call ErrorTest(1)
End Sub

'---------- Select's rejection filter if clicking
Sub combo_Filters_Click ()
Dim choice As String
  Select Case combo_Filters.Text
    Case " OFF": choice = "0"
    Case "  2F": choice = "1"
    Case "   F": choice = "2"
    Case "2F+F": choice = "3"
    Case "": choice = "null"
  End Select
  If choice <> "null" Then
    Call Send(0, lock_number, "LF" & choice & Chr$(13), DABend)
    gReject = Val(choice) '---------- Set As Default
  End If
End Sub

'---------- Select's output display contents
Sub combo_OutputDisplay_Click ()
Dim choice As String
   Select Case combo_OutputDisplay.Text
     Case " % FS  ": choice = "0"
     Case " Signal": choice = "1"
     Case "Offset%": choice = "2"
     Case " Noise%": choice = "3"
     Case " Ratio ": choice = "4"
     Case " Log R ": choice = "5"
     Case "": choice = " null "
   End Select
   If choice <> " null " Then
     Call Send(0, lock_number, "D2 " & choice & Chr$(13), DABend)
     gOutDisplay = Val(choice)  ' ---------- Set As Default
   End If
   Call ErrorTest(1)
End Sub

'---------- Select's the sensitivity
Sub combo_Sensitivity_Click ()
Dim choice As String
Dim auto As Integer
   auto = False
   Select Case combo_Sensitivity.Text
     Case " AUTO ": auto = True
     Case "100 nV": choice = "0"
     Case "300 nV": choice = "1"
     Case "  1 uV": choice = "2"
     Case "  3 uV": choice = "3"
     Case " 10 uV": choice = "4"
     Case " 30 uV": choice = "5"
     Case "100 uV": choice = "6"
```

```
      Case "300 uV": choice = "7"
      Case "  1 mV": choice = "8"
      Case "  3 mV": choice = "9"
      Case " 10 mV": choice = "10"
      Case " 30 mV": choice = "11"
      Case "100 mV": choice = "12"
      Case "300 mV": choice = "13"
      Case "   1 V": choice = "14"
      Case "   3 V": choice = "15"
      Case "": choice = " null "
   End Select
   If choice <> " null " Then
      If auto Then ' Put's the lock-in sensitivity in auto-mode
         Call Send(0, lock_number, "AS" & Chr$(13), DABend)
        Else
         Call Send(0, lock_number, "AA" & Chr$(13), DABend)
         Call Send(0, lock_number, "SEN " & choice & Chr$(13), DABend)
         If combo_Sensitivity <> "AUTO" Then gSensitivity = Val(choice) + 1' ---------- Set As Default
      End If
   End If
   Call ErrorTest(1)
End Sub


'----------------- Select's the time constant
Sub combo_TimeConst_Click ()
Dim choice As String
   Select Case combo_TimeConst.Text
      Case "  1 ms": choice = "0"
      Case "  3 ms": choice = "1"
      Case " 10 ms": choice = "2"
      Case " 30 ms": choice = "3"
      Case "100 ms": choice = "4"
      Case "300 ms": choice = "5"
      Case "   1 s": choice = "6"
      Case "   3 s": choice = "7"
      Case "  10 s": choice = "8"
      Case "  30 s": choice = "9"
      Case " 100 s": choice = "10"
      Case " 300 s": choice = "11"
      Case "1000 s": choice = "12"
      Case "3000 s": choice = "13"
      Case "": choice = " null "
   End Select
   If choice <> " null " Then
     Call Send(0, lock_number, "XTC " & choice & Chr$(13), DABend)
     gTimeConst = Val(choice)  '---------- Set As Default
   End If
   Call ErrorTest(1)
End Sub

'_____________________ IMPORTANT sub when the window loads into memory
Sub Form_Load ()
   '------------------- Initialize combo lists
   combo_Sensitivity.AddItem " AUTO "
   combo_Sensitivity.AddItem "100 nV": combo_Sensitivity.AddItem "300 nV"
   combo_Sensitivity.AddItem "  1 uV": combo_Sensitivity.AddItem "  3 uV"
   combo_Sensitivity.AddItem " 10 uV": combo_Sensitivity.AddItem " 30 uV"
   combo_Sensitivity.AddItem "100 uV": combo_Sensitivity.AddItem "300 uV"
   combo_Sensitivity.AddItem "  1 mV": combo_Sensitivity.AddItem "  3 mV"
   combo_Sensitivity.AddItem " 10 mV": combo_Sensitivity.AddItem " 30 mV"
   combo_Sensitivity.AddItem "100 mV": combo_Sensitivity.AddItem "300 mV"
   combo_Sensitivity.AddItem "   1 V": combo_Sensitivity.AddItem "   3 V"

   combo_Filters.AddItem " OFF": combo_Filters.AddItem "   F"
   combo_Filters.AddItem "  2F": combo_Filters.AddItem "2F+F"
```

```
  combo_TimeConst.AddItem "  1 ms": combo_TimeConst.AddItem "  3 ms":
  combo_TimeConst.AddItem " 10 ms": combo_TimeConst.AddItem " 30 ms":
  combo_TimeConst.AddItem "100 ms": combo_TimeConst.AddItem "300 ms":
  combo_TimeConst.AddItem "  1 s": combo_TimeConst.AddItem "  3 s":
  combo_TimeConst.AddItem "  10 s": combo_TimeConst.AddItem "  30 s":
  combo_TimeConst.AddItem " 100 s": combo_TimeConst.AddItem " 300 s":
  combo_TimeConst.AddItem "1000 s": combo_TimeConst.AddItem "3000 s"

  combo_OutputDisplay.AddItem " % FS  ": combo_OutputDisplay.AddItem " Signal"
  combo_OutputDisplay.AddItem "Offset%": combo_OutputDisplay.AddItem " Noise%"
  combo_OutputDisplay.AddItem " Ratio ": combo_OutputDisplay.AddItem " Log R "

  '------------------------ Wait 1/2 sec before setting default parameters
  timer_Init.Enabled = True

End Sub

'---------------------- Set all parameters to saved default parameters
Sub InitDevice ()
 '----------- Set and send sensitivity
 combo_Sensitivity.ListIndex = gSensitivity
 combo_Sensitivity_Click
 Call WaitForDevice

 '----------- Set and send filter
 Select Case gFilter
  Case "BP":  opt_BandPass.Value = True
  Case "LP":  opt_LowPass.Value = True
  Case "NOTCH":  opt_Notch.Value = True
  Case "FLAT":  opt_Flat.Value = True
 End Select
 text_Tuning.Text = gFilter
 Call WaitForDevice

 '---------- Set freq tuning to track or manual
 If gSetFrq = "TRACK" Then opt_Track.Value = True Else opt_Manual.Value = True
 Call WaitForDevice

 '---------- Set LineReject on/off
 combo_Filters.ListIndex = gReject
 combo_Filters_Click
 Call WaitForDevice

 '---------- Set Hertz and TuneValue
  Select Case gHertz
    Case "HZ1":  opt_Hz1.Value = True
    Case "HZ2":  opt_Hz2.Value = True
    Case "HZ3":  opt_Hz3.Value = True
    Case "HZ4":  opt_Hz4.Value = True
    Case "HZ5":  opt_Hz5.Value = True
    Case "REF F": opt_FilterRef.Value = True
  End Select
  Call WaitForDevice
  If gHertz <> "REF F" Then text_Tuning.Text = "    " & Format(gTuneValue, "fixed")

 '----------- Set internal and double freq
 If gInternal = True Then check_Internal.Value = True Else check_Internal.Value = False
 Call WaitForDevice
 If gFx2 = True Then check_2F.Value = True Else check_2F.Value = False
 Call WaitForDevice

 '------------ Set TimeConstant
 combo_TimeConst.ListIndex = gTimeConst
 combo_TimeConst_Click
 Call WaitForDevice

 '------------ Set Slope
```

```
label_Slope.Caption = gSlope
c_OutputSlope_click
Call WaitForDevice

'---------- Set resolution
Select Case gResolution
  Case "HI": opt_HiRes.Value = True
  Case "NORM": opt_NormRes.Value = True
  Case "STAB": opt_HiStab.Value = True
End Select
Call WaitForDevice

'---------- Set OutDisplay
combo_OutputDisplay.ListIndex = gOutDisplay
combo_OutputDisplay_Click
Call WaitForDevice

'---------- Set off-set
If gOffsetOn = True Then
  label_Offset.Caption = "Off"
  SetOffset (True)
 Else
  label_Offset.Caption = "On"
  SetOffset (True)
End If
Call WaitForDevice

'---------- Set expand on/off
If gExpandx10 = True Then check_Expand.Value = True Else check_Expand.Value = False
Call WaitForDevice

'---------- Set continues reading
If gReadOutput = True Then
  check_ReadOutput.Value = True
 Else
  check_ReadOutput.Value = False
End If
Call WaitForDevice
End Sub

'---------- Makes auto initialize
Sub menu_AutoInit_Click ()
   Dim X As Variant
   '---------- Do automatic sensitivity
   text_Information.Visible = True
   text_Information.Text = "    Doing automatic search for sensitivity , please wait....."
   combo_Sensitivity.ListIndex = 0
   Call Send(0, lock_number, "AS" & Chr$(13), DABend)
   Call ErrorTest(1)
   WaitForDevice
   '---------- Do automatic tuning
   text_Information.Text = "  Doing automatic tuning to internal or external frequency , please wait....."
   For X = 1 To 200000: Next X
   Call Send(0, lock_number, "ATS" & Chr$(13), DABend)
   Call ErrorTest(1)
   WaitForDevice
   '---------- Do automatic phasing
   text_Information.Text = "  Doing automatic phasing to maximum output , please wait....."
   Call Send(0, lock_number, "AQN" & Chr$(13), DABend)
   Call ErrorTest(1)
   WaitForDevice
   '---------- Do automatic offset
   text_Information.Text = "     Doing automatic offset , please wait....."
   label_Offset.Caption = "On"
   For X = 1 To 200000: Next X
   Call Send(0, lock_number, "AXO" & Chr$(13), DABend)
   Call ErrorTest(1)
```

```
   WaitForDevice
   text_Information.Visible = False
   MsgBox "Finished !", 0, "AutoInit"
End Sub

'------------------------ Make an auto mesure
Sub menu_AutoMesure_Click ()
Dim buffer As String
  text_Information.Visible = True
  text_Information.Text = "Doing an automesure, please wait 1 min....."
  buffer = Space(10)
  Call Send(0, lock_number, "ASM" & Chr$(13), DABend)
  Call ErrorTest(1)
  WaitForDevice
  '--------------- Set's the screenparameters to automesure standard
  combo_Sensitivity.ListIndex = 0     '--- Set sensitivity to auto
  opt_BandPass.Value = True           '--- Set filter to BandPass
  opt_Manual.Value = True             '--- Set frequency to manual
  combo_Filters.ListIndex = 0         '--- Disconnect line reject
  check_2F.Value = False              '--- Disconnect double frequency
  label_Slope.Caption = "=12 dB"      '--- Set slope to 12 dB
  opt_NormRes.Value = True            '--- Set resolution to normal
  label_Offset.Caption = "Off"        '--- Shut-off offset
  check_Expand.Value = 0              '--- Shut-off expand x 10
  combo_OutputDisplay.ListIndex = 0  '--- Set Output display to %FS Signal
  WaitForDevice
  '--------------- Read the timeconstant set by automesure and set screenparameter to the same
  Call Send(0, lock_number, "XTC" & Chr$(13), DABend)
  Call ErrorTest(1)
  Call Receive(0, lock_number, buffer, STOPend)
  Call ErrorTest(2)
  combo_TimeConst.ListIndex = Val(buffer)
  text_Information.Visible = False
End Sub

'------------ Call for help on subject Lock-In
Sub menu_Help_Click ()
   Call HELP("LOCK-IN")
End Sub

'------------- Save current parameters as default
Sub menu_Save_Click ()
   Call SaveDefaults
End Sub

'------------------ Select BP-filter
Sub opt_BandPass_click (Value As Integer)
   Call Send(0, lock_number, "FLT 3" & Chr$(13), DABend)
   Call ErrorTest(1)
   gFilter = "BP" ' ---------- Set As Default
End Sub

'------------------ Select tuning freq to be ref. freq
Sub opt_FilterRef_click (Value As Integer)
  Min = 0: Max = 0
  TuningHz = 0
  Stepp = 0
  text_Tuning.Text = "   Ref F"
  gHertz = "REF F" '---------- Set As Default
  Call Send(0, lock_number, "D1 4" & Chr$(13), DABend)
  Call ErrorTest(1)
End Sub
```

```
'---------- Select filter to be flat
Sub opt_Flat_click (Value As Integer)
  Call Send(0, lock_number, "FLT 0" & Chr$(13), DABend)
  Call ErrorTest(1)
  gFilter = "FLAT" ' ---------- Set As Default
End Sub

'---------- Select resolution to be High
Sub opt_HiRes_Click (Value As Integer)
  Call Send(0, lock_number, "DR 2" & Chr$(13), DABend)
  Call ErrorTest(1)
  gResolution = "HI"  '---------- Set As Default
End Sub

'---------- Select resolution to be High Stability
Sub opt_HiStab_Click (Value As Integer)
  Call Send(0, lock_number, "DR 0" & Chr$(13), DABend)
  Call ErrorTest(1)
  gResolution = "STAB"  '---------- Set As Default
End Sub

'---------- Select range 1 in Hertz
Sub opt_Hz1_click (Value As Integer)
  Range = 0
  Min = .5: Max = 12
  TuningHz = 5
  Stepp = .01
  Autoscale = 100
  text_Tuning.Text = "    " & Format(TuningHz, "fixed") & " Hz"
  gHertz = "HZ1" '---------- Set As Default
  Call Send(0, lock_number, "D1 4,FF " & (TuningHz * Autoscale) & " " & Range & Chr$(13), DABend)
  Call ErrorTest(1)
End Sub

'---------- Select range 2 in Hertz
Sub opt_Hz2_click (Value As Integer)
  Range = 1
  Min = 10: Max = 120
  TuningHz = 50
  Stepp = .1
  Autoscale = 10
  text_Tuning.Text = "    " & Format(TuningHz, "fixed") & " Hz"
  gHertz = "HZ2" '---------- Set As Default
  Call Send(0, lock_number, "FF " & (TuningHz * Autoscale) & " " & Range & Chr$(13), DABend)
  Call ErrorTest(1)
End Sub

'---------- Select range 3 in Hertz
Sub opt_Hz3_click (Value As Integer)
  Range = 2
  Min = .1: Max = 1.2
  TuningHz = .5
  Stepp = .001
  Autoscale = 1000
  text_Tuning.Text = "    " & Format(TuningHz, "fixed") & " kHz"
  gHertz = "HZ3" '---------- Set As Default
  Call Send(0, lock_number, "FF " & (TuningHz * Autoscale) & " " & Range & Chr$(13), DABend)
  Call ErrorTest(1)
End Sub

'---------- Select range 4 in Hertz
Sub opt_Hz4_click (Value As Integer)
  Range = 3
  Min = 1: Max = 12
  TuningHz = 5
  Stepp = .01
  Autoscale = 100
```

```
  text_Tuning.Text = "    " & Format(TuningHz, "fixed") & " kHz"
  gHertz = "HZ4" '---------- Set As Default
  Call Send(0, lock_number, "FF " & (TuningHz * Autoscale) & " " & Range & Chr$(13), DABend)
  Call ErrorTest(1)
End Sub

'---------------- Select range 5 in Hertz
Sub opt_Hz5_click (Value As Integer)
  Range = 4
  Min = 10: Max = 120
  TuningHz = 50
  Stepp = .1
  Autoscale = 10
  text_Tuning.Text = "    " & Format(TuningHz, "fixed") & " kHz"
  gHertz = "HZ5" '---------- Set As Default
  Call Send(0, lock_number, "FF " & (TuningHz * Autoscale) & " " & Range & Chr$(13), DABend)
  Call ErrorTest(1)
End Sub

'---------------- Select's filter to be LP
Sub opt_LowPass_click (Value As Integer)
  Call Send(0, lock_number, "FLT 2" & Chr$(13), DABend)
  Call ErrorTest(1)
  gFilter = "LP"  '---------- Set As Default
End Sub

'-------------- Select's tuning freq to be manual
Sub opt_Manual_click (Value As Integer)
  Call Send(0, lock_number, "ATC 0" & Chr$(13), DABend)
  Call ErrorTest(1)
  gSetFrq = "MAN"' ---------- Set As Default
End Sub

'------------ Select's resolution to be Normal
Sub opt_NormRes_Click (Value As Integer)
  Call Send(0, lock_number, "DR 1" & Chr$(13), DABend)
  Call ErrorTest(1)
  gResolution = "NORM"  '---------- Set As Default
End Sub

'----------------- Select's filter to be Notch
Sub opt_Notch_click (Value As Integer)
  Call Send(0, lock_number, "FLT 1" & Chr$(13), DABend)
  Call ErrorTest(1)
  gFilter = "NOTCH"  ' ---------- Set As Default
End Sub

'--------------- Select's tuning freq to be track
Sub opt_Track_click (Value As Integer)
  Call Send(0, lock_number, "ATC 1" & Chr$(13), DABend)
  Call ErrorTest(1)
  gSetFrq = "TRACK"  '---------- Set As Default
End Sub

'------------ Makes it possible to interrupt a serial poll just by clicking the mousebutton
Sub panel_1_Click ()
Dim Response As Integer
  If Waiting = True Then
    Response = MsgBox("I'm trying to wait for the device to signal ready,do you want to break this ?  (If the
autobutton on the lockin is off and you have been waiting for a long time there could be something wrong.", 68,
"Poll break")
    If Response = 6 Then
      Waiting = False
      Call Send(0, lock_number, "AA" & Chr$(13), DABend)
    End If
  End If
End Sub
```

```
'---------- Set offset value
Sub SetOffset (automatic As Integer)
Dim retur As Variant
If label_Offset.Caption = "Off" Then
  '---------- If not in loading mode get offset value
  If automatic <> True Then
     retur = InputBox$("Input offset voltage:" & Chr(10) & "( Range : -150V - 150V )", "Offset")
     Else
     retur = Str(gOffsetValue)
  End If
  '---------- Send offset value
  If retur <> "" Then
     retur = Val(retur)
     retur = retur * 10
     Call Send(0, lock_number, "XOF 1 " & retur & Chr$(13), DABend)
     label_Offset.Caption = "On"
     gOffsetOn = True              ' ---------- Set As Default
     gOffsetValue = retur          ' ---------- Set As Default
   End If
Else
 '---------- Else disconnect offset
 Call Send(0, lock_number, "XOF 1 0 " & Chr$(13), DABend)
 Call Send(0, lock_number, "XOF 0 " & Chr$(13), DABend)
 label_Offset.Caption = "Off"
 gOffsetOn = False             ' ---------- Set As Default
 gOffsetValue = 0              ' ---------- Set As Default
End If
Call ErrorTest(1)
End Sub

'---------- Read and display lock-in output every 1/2 sec
Sub timer_Display_Timer ()
Dim buffer As String
Dim display As Single
Dim spaces As String
Dim X As Integer
 '--------- Get reading
 buffer = Space(10):
 Call Send(0, lock_number, "OUT" & Chr$(13), DABend)
 'Call ErrorTest(1)
 Call Receive(0, lock_number, buffer, STOPend) 'Call ErrorTest(2)
 '--------- Format reading for fixed pos. for positiv/negative values
 display = Val(buffer)
 display = (display / 100)
 If display < 0 Then spaces = " " Else spaces = "  "
 text_output.Text = spaces & Format(display, "fixed")
End Sub

'---------- Call for initialization when loading form
Sub timer_Init_Timer ()
InitDevice
timer
_Init.Enabled = False
End Sub
```

****

D) THE PIEZO FORM (fPiezo.frm) Piezo controllern

****

```
Option Explicit

'---------- Set's input 1 on channel 1 on/off
Sub check_Input1_Click (Value As Integer)
If Value = 0 Then
  Call Send(0, Piezo_Number, "P1=0" & Chr$(13), DABend)
  gInput1 = False          '_____Set default value
    Else
  Call Send(0, Piezo_Number, "P1=1" & Chr$(13), DABend)
  gInput1 = True           '_____Set default value
End If
delay (PD)
End Sub

'---------- Set's input 2 on channel 2 on/off
Sub check_Input2_Click (Value As Integer)
If Value = 0 Then
  Call Send(0, Piezo_Number, "P2=0" & Chr$(13), DABend)
  gInput2 = False          '_____Set default value
    Else
  Call Send(0, Piezo_Number, "P2=1" & Chr$(13), DABend)
  gInput2 = True           '_____Set default value
End If
delay (PD)
End Sub

'---------- Set's input 3 on channel 3 on/off
Sub check_Input3_Click (Value As Integer)
If Value = 0 Then
  Call Send(0, Piezo_Number, "P3=0" & Chr$(13), DABend)
  gInput3 = False          '_____Set default value
    Else
  Call Send(0, Piezo_Number, "P3=1" & Chr$(13), DABend)
  gInput3 = True           '_____Set default value
End If
delay (PD)
End Sub

'---------- Set's reading the piezo on/off
Sub check_ReadingOn_Click (Value As Integer)
  If Value <> 0 Then
    timer_Read.Enabled = True
  Else
    timer_Read.Enabled = False
  End If
End Sub

'---------- Get reading from channel(index) and display on display(index)
Sub DisplayReadings (index As Integer)
Dim range, buffer As String
Dim selected As SSOption
Dim display As Single
    buffer = Space(10)
    '---------- Get reading and format
    Call Send(0, Piezo_Number, "I" & Str$(index) & Chr$(13), DABend)
    delay (PD)
    Call Receive(0, Piezo_Number, buffer, STOPend)
    range = Left$(buffer, 1)
    buffer = Right$(buffer, 8)
```

```
'---------- Get current status on option button for channel=index
  Select Case index
   Case 1: Set selected = opt_Volt1
   Case 2: Set selected = opt_Volt2
   Case 3: Set selected = opt_Volt3
  End Select
  '---------- Evaluate reading depending on status (volt/um, high/low ?)
  If selected.Value = True Then
     display = (Val(buffer) / 32768) * 100
    ElseIf range = "L" Then
     display = (Val(buffer) / 32768) * 20
    Else
     display = (Val(buffer) / 32768) * 200
  End If
  '---------- Display reading on display=index
  Select Case index
    Case 1: text_Output1.Text = Format(display, "00.00")
    Case 2: text_Output2.Text = Format(display, "00.00")
    Case 3: text_Output3.Text = Format(display, "00.00")
  End Select
End Sub

'---------- Set piezo to saved default's
Sub Form_Load ()
 InitPiezo
End Sub

'---------- Put the piezo in saved default mode
Sub InitPiezo ()
  If gVoltOrUm1 = "VOLT" Then opt_Volt1.Value = True Else opt_um1.Value = True
  If gVoltOrUm2 = "VOLT" Then opt_Volt2.Value = True Else opt_um2.Value = True
  If gVoltOrUm3 = "VOLT" Then opt_Volt3.Value = True Else opt_um3.Value = True
  If gInput1 = True Then check_Input1.Value = True Else check_Input1.Value = False
  If gInput2 = True Then check_Input2.Value = True Else check_Input2.Value = False
  If gInput3 = True Then check_Input3.Value = True Else check_Input3.Value = False
End Sub

'_____ Call for HELP on subject PIEZO
Sub menu_Help_Click ()
  Call HELP("PIEZO")
End Sub

'---------- Save current parameters as default
Sub menu_Save_Click ()
  Call SaveDefaults
End Sub

'---------- Set's piezo channel 1 to um mode
Sub opt_um1_Click (Value As Integer)
  Call Send(0, Piezo_Number, "M1=1" & Chr$(13), DABend)
' scroll_change (0)
  gVoltOrUm1 = "UM"                ' _____Set default value
  delay (PD)
End Sub

'---------- Set's piezo channel 2 to um mode
Sub opt_um2_Click (Value As Integer)
  Call Send(0, Piezo_Number, "M2=1" & Chr$(13), DABend)
' scroll_change (1)
  gVoltOrUm2 = "UM"                ' _____Set default value
  delay (PD)
End Sub

''---------- Set's piezo channel 3 to um mode
Sub opt_um3_Click (Value As Integer)
  Call Send(0, Piezo_Number, "M3=1" & Chr$(13), DABend)
' scroll_change (2)
```

```
 gVoltOrUm3 = "UM"                ' _____Set default value
 delay (PD)
End Sub

'------------- Set's piezo channel 1 to volt mode
Sub opt_Volt1_Click (Value As Integer)
 Call Send(0, Piezo_Number, "M1=0" & Chr$(13), DABend)
' scroll_change (0)
 gVoltOrUm1 = "VOLT"              ' _____Set default value
 delay (PD)
End Sub

'------------- Set's piezo channel 2 to volt mode
Sub opt_Volt2_Click (Value As Integer)
 Call Send(0, Piezo_Number, "M2=0" & Chr$(13), DABend)
' scroll_change (1)
 gVoltOrUm2 = "VOLT"              ' _____Set default value
 delay (PD)
End Sub

'------------- Set's piezo channel 3 to volt mode
Sub opt_Volt3_Click (Value As Integer)
 Call Send(0, Piezo_Number, "M3=0" & Chr$(13), DABend)
' scroll_change (2)
 gVoltOrUm3 = "VOLT"              ' _____Set default value
 delay (PD)
End Sub

'---------------------- Send's new position/volt to piezo
Sub scroll_change (index As Integer)
 Dim Value, opt As String
  opt = Str$(index + 1)
  Value = Str$(2 * scroll(index).Value)
  Call Send(0, Piezo_Number, "O" & opt & "+" & Value & Chr$(13), DABend)
  delay (PD)
  DisplayReadings (index + 1)
End Sub

'-------------------- Get readings from piezo (every 1/2 sec)
Sub timer_Read_Timer ()
Dim x As Integer
 For x = 1 To 3
  Call DisplayReadings(x)
 Next x
End Sub
```

****

E) THE GPIB doctor (fGPIB.frm) Investigates GPIB problems

****

```
Dim Loading As Integer

'---------- Start ibconf-program
Sub c_Advanced_Click ()
Dim dummy
  dummy = Shell("c:\windows\control.exe", 4)
End Sub

'---------- Clear all devices
Sub c_ClearAll_click ()
  Call DevClear(0, NOADDR)          ' _______Sends Device Clear to ALL instruments
    Call ErrorTest(7)
End Sub
```

```
'---------- Set interface clear on GPIB-card
Sub c_ClearGPIB_click ()
  Call SendIFC(0)
    Call ErrorTest(6)
End Sub

'---------- Call for finding instruments
Sub c_Find_Click ()
  Call FindInstruments
End Sub

'---------- Make a SeriellPollTest on the lock-in
Sub c_SeriellPoll_Click ()
Dim SRQByte As Integer
Dim PollByte As Integer
Dim SQRInfo, PollInfo, buffer As String
 Call Send(0, Lock_Number, "OUT" & Chr$(13), DABend)
  Call ErrorTest(1)
 Call ReadStatusByte(0, Lock_Number, PollByte)
  If (PollByte And &H80) = 128 Then PollInfo = "Polling returned a message that there is valid data to read.
So,Ok !" Else PollInfo = "Polling response was bad, the device signaled it didn't have valid data to send or maybe
it's not connected."
 Call Receive(0, Lock_Number, buffer, STOPend)
  Call ErrorTest(2)
 MsgBox "Testing communications found that:" & Chr(10) & "* Sending a request to get a reading..." &
Chr(10) & Chr(10) & PollInfo & Chr(10) & Chr(10) & " * Receiving the data....", 0, "TestResult"
End Sub

'---------- Start GPIB-spy program
Sub c_Spy_Click ()
Dim dummy
  dummy = Shell("c:\GPIB-PCW\gpibspy.exe", 4)
End Sub

'---------- Change adress on lock-in
Sub combo_LockInAdress_Click ()
Dim response As Integer
  If Loading = False Then
   response = MsgBox("If you change this value to a wrong value nothing will work !" & Chr(10) & "(=You
                      should know what you are doing)", 49, "Warning")
     If response = 1 Then
      Lock_Number = combo_LockInAdress.Text
      Else
      Loading = True
      combo_LockInAdress.ListIndex = 10
      Loading = False
     End If
  End If
End Sub

'---------- Change adress on piezo
Sub combo_PiezoAdress_Click ()
Dim response As Integer
  If Loading = False Then
    response = MsgBox("If you change this value to a wrong value nothing will work !" & Chr(10) & "(=You
should know what you are doing)", 49, "Warning")
      If response = 1 Then
        Piezo_Number = combo_PiezoAdress.Text
        Else
        Loading = True
        combo_PiezoAdress.ListIndex = 12
        Loading = False
      End If
  End If
End Sub
```

```
'---------- Search for and display all instruments on the bus
Sub FindInstruments ()
Static Instrument(0 To 31) As Integer
Static result(0 To 31) As Integer
   text_DevFound.Text = "Adresses: "
   fGPIB.MousePointer = 11
   For X = 0 To 30
     Instrument(X) = X
   Next X
   Instrument(31) = NOADDR
   Call Findlstn(0, Instrument(), result(), 30)
   For X = 0 To 30
     text_DevFound.Text = text_DevFound.Text & "   " & result(X)
   Next X
   fGPIB.MousePointer = 0
End Sub

'---------- Initialize combo's and Errorlabel
Sub Form_Load ()
  Loading = True
  timer1.Enabled = True
  For X = 0 To 30
    combo_LockInAdress.AddItem X
    combo_PiezoAdress.AddItem X
  Next X
  combo_LockInAdress.ListIndex = 10
  combo_PiezoAdress.ListIndex = 12
  label_Errors.Caption = "   " & ErrorCounter
  Loading = False
End Sub

'_____ Undo update the ErrorCounter when leaving form
Sub Form_Unload (Cancel As Integer)
  timer1.Enabled = False
End Sub

'_____ Call for help on subject GPIB
Sub menu_GetHelp_Click ()
  Call HELP("GPIB")
End Sub

'---------- Update ErrorCounter every 2 sec
Sub Timer1_Timer ()
  label_Errors.Caption = "   " & ErrorCounter
End Sub
```

****

F) The Save As Form (fSaveAs.frm) Handles the saving off the scan to disk

****

```
Option Explicit
Dim RecFileName As String

'---------- Exit SavingForm
Sub c_Cancel_Click ()
  fSaveAs.Hide
End Sub

'_____ Call for help on subject SAVE
Sub c_Help_Click ()
   Call HELP("SAVE")
End Sub
```

```
'---------- Call appropriate saving format
Sub c_Save_Click ()
  fSaveAs.MousePointer = 11
  gSavedDate = Date$
  gNbrOfSavings = gNbrOfSavings + 1
  If opt_RawFormat.Value = True Then Call SaveAsRaw
  If opt_DataFormat.Value = True Then Call SaveAsData
  If opt_Bitmap.Value = True Then Call SaveAsBitmap
  If opt_DataAndBitmap.Value = True Then
    Call SaveAsBitmap
    Call SaveAsData
  End If
  fSaveAs.MousePointer = 0
  fSaveAs.Hide
End Sub

'---------- Set correct pattern for file1
Sub combo_Pattern_Change ()
 file1.Pattern = combo_Pattern.Text
End Sub

'---------- Set path for file1
Sub Dir1_Change ()
 file1.Path = dir1.Path
 Call ShowFileName
End Sub

'---------- Set path for drive1
Sub Drive1_Change ()
 dir1.Path = drive1.Drive
End Sub

'---------- Update recommended filename
Sub Form_Activate ()
Dim currentMonth, currentYear, StartPath As String
 RecFileName = Date$
 Select Case Month(Now)
   Case 1: currentMonth = "1jan": Case 2: currentMonth = "2feb": Case 3: currentMonth = "3mar": Case 4:
currentMonth = "4apr": Case 5: currentMonth = "5may": Case 6: currentMonth = "6jun"
   Case 7: currentMonth = "7jul": Case 8: currentMonth = "8aug": Case 9: currentMonth = "90sep": Case 10:
currentMonth = "91oct": Case 11: currentMonth = "92nov": Case 12: currentMonth = "93dec"
 End Select
 currentYear = Right$(RecFileName, 4)
 StartPath = "c:\scannings\" & currentYear & "\" & currentMonth & "\"
 RecFileName = Left$(RecFileName, 2) & "_" & Mid$(RecFileName, 4, 2) & "_" & gNbrOfSavings
 text_RecFileName.Text = StartPath & RecFileName
End Sub

'---------- Set combo's and RecFileName
Sub Form_Load ()
Dim currentMonth, currentYear, StartPath As String
 combo_Pattern.AddItem "*.bmp"
 combo_Pattern.AddItem "*.dat"
 combo_Pattern.AddItem "*.raw"
 combo_Pattern.AddItem "*.bmp;*.dat;*.raw"
 combo_Pattern.ListIndex = 3
 '---------- Construct recommended filename
 RecFileName = Date$
 Select Case Month(Now)
   Case 1: currentMonth = "1jan": Case 2: currentMonth = "2feb": Case 3: currentMonth = "3mar": Case 4:
currentMonth = "4apr": Case 5: currentMonth = "5may": Case 6: currentMonth = "6jun"
   Case 7: currentMonth = "7jul": Case 8: currentMonth = "8aug": Case 9: currentMonth = "90sep": Case 10:
currentMonth = "91oct": Case 11: currentMonth = "92nov": Case 12: currentMonth = "93dec"
 End Select
 currentYear = Right$(RecFileName, 4)
 StartPath = "c:\scannings\" & currentYear & "\" & currentMonth & "\"
 RecFileName = Left$(RecFileName, 2) & "_" & Mid$(RecFileName, 4, 2) & "_" & gNbrOfSavings
```

```
 text_RecFileName.Text = StartPath & RecFileName
  '---------- Set start path
 drive1.Drive = "c:"
 dir1.Path = "c:\scanning"
End Sub

'---------- Save selected file as bitmap (*.bmp)
Sub SaveAsBitmap ()
   Dim x, Y, rad, kol As Integer
   Dim pos As Long
   Dim BitHead As BitmapFileHeader
   Dim BitInfo As BitmapInfoHeader
   Dim GoodNumber As String * 1
   Dim BadNumber As String * 1
   Dim ActiveFile As String
   Dim Path As String
   Static GrayShade(0 To 255) As RGBQUAD
   Static BitData() As Integer
   On Error GoTo BasicError
   ReDim BitData(NbrOfPixels, NbrOfPixels)

   '---------- Create BitmapFileHead
   BitHead.bfType = 19778
   BitHead.bfSize = (14 + 40 + 256 * 4 + NbrOfPixels * NbrOfPixels)
   BitHead.bfReserved1 = 0
   BitHead.bfReserved2 = 0
   BitHead.bfOffBits = 1078
   '---------- Create BitmapInfoHead
   BitInfo.biSize = 40
   BitInfo.biWidth = NbrOfPixels
   BitInfo.biHeight = NbrOfPixels
   BitInfo.biPlanes = 1
   BitInfo.biBitCount = 8
   BitInfo.biCompression = 0
   BitInfo.biSizeImage = NbrOfPixels * NbrOfPixels
   BitInfo.biXPelsPerMeter = 0
   BitInfo.biYPelsPerMeter = 0
   BitInfo.biClrUsed = 256
   BitInfo.biClrImportant = 256
   '---------- Create Palette
   For x = 0 To 255
     GrayShade(x).rgbBlue = x
     GrayShade(x).rgbGreen = x
     GrayShade(x).rgbRed = x
     GrayShade(x).rgbReserved = 0
   Next x
   '---------- Transform, because a Bitmap begins lower-left and PixelPicture begins upper-Left
   x = 0
   For rad = NbrOfPixels To 1 Step -1
     For kol = 1 To NbrOfPixels
       x = x + 1
       BitData(rad, kol) = PixelPicture(x)
     Next kol
   Next rad
   '---------- Save as file
   ActiveFile = text_RecFileName.Text & ".bmp"
   Open ActiveFile For Binary As 1
     Put #1, , BitHead
     Put #1, , BitInfo
     For x = 0 To 255
       Put #1, , GrayShade(x)
     Next x
     For rad = 1 To NbrOfPixels
       For kol = 1 To NbrOfPixels
         Put #1, , BitData(rad, kol)
       Next kol
     Next rad
```

```
  Close #1
  '---------------- Transform to only 1 Byte/pixel
  Path = dir1.Path
  If Right$(Path, 1) <> "\" Then Path = Path & "\"
  Open ActiveFile For Binary As 1
  Open Path & "slask.bmp" For Binary As 2
    Put #2, , BitHead
    Put #2, , BitInfo
    pos = 55
      For x = 1 To (256 * 4 + NbrOfPixels * NbrOfPixels)
        Get #1, pos, GoodNumber
        Put #2, , GoodNumber
        pos = pos + 2
      Next x
    Close 1
    Close 2
    Kill ActiveFile
    Name Path & "slask.bmp" As ActiveFile
TheEnd:
    Exit Sub
'------------ If an error occurs
BasicError:
Dim dummy As Variant
  dummy = MsgBox(ERRORINF, 48, "Error")
  Resume TheEnd
End Sub

'---------------- Save selected file as DataFile (*.dat)
Sub SaveAsData ()
Dim ActiveFile As String
Dim x, MAX As Integer
 On Error GoTo BasicError2
 If SquareScan = True Then MAX = NbrOfPixels * NbrOfPixels:  Else MAX = NbrOfPixels
 ActiveFile = text_RecFileName.Text & ".dat"
 Open ActiveFile For Output As 1
 If SquareScan = True Then Print #1, ScanType, NbrOfPixels, XStep, YStep, DelayTime, TimePassed
  For x = 1 To MAX
    Print #1, PixelPicture(x)
  Next x
 Close 1
TheEnd2:
 Exit Sub
BasicError2:
Dim dummy As Variant
  dummy = MsgBox(ERRORINF, 48, "Error")
  Resume TheEnd2
End Sub

'----------- Save selected file as RawFormat (*.raw)
Sub SaveAsRaw ()
Dim GoodNumber As String
Dim BadNumber As String
Dim ActiveFile As String
Dim Path As String
Dim x As Integer
 BadNumber = "0": GoodNumber = "0"
 ActiveFile = text_RecFileName.Text & ".raw"
 Open ActiveFile For Binary As 1
  For x = 1 To NbrOfPixels * NbrOfPixels
    Put 1#, , PixelPicture(x)
  Next x
 Close 1
 ' Transformerar filen till endast 1 byte/tal
 Open ActiveFile For Binary As 1
 Path = dir1.Path
 If Right$(Path, 1) <> "\" Then Path = Path & "\"
 Open Path & "slask.raw" For Binary As 2
```

```
    For x = 0 To NbrOfPixels * NbrOfPixels / 2
      Get 1#, , GoodNumber
      Put 2#, , GoodNumber
      Get 1#, , BadNumber
    Next x
  Close 1
  Close 2
  Kill ActiveFile     ' Döda filen med 2 bytes/tal
  Name Path & "slask.raw" As ActiveFile ' Byt dess namn till den aktiva filen
End Sub

'------------------ Display current file
Sub ShowFileName ()
Dim Path As String
  Path = dir1.Path
  If Right$(Path, 1) <> "\" Then Path = Path & "\"
  text_RecFileName.Text = Path & RecFileName
End Sub
```

****

G) THE OPEN FORM (OpenFile.frm) Opens a saved data file

****

```
Option Explicit
Dim RecFileName As String

'------------ Exit savingform
Sub c_Cancel_Click ()
  fOpenFile.Hide
End Sub

'_________ Call for help on subject OPEN
Sub c_Help_Click ()
  Call HELP("OPEN")
End Sub

'--------------- Open selected file
Sub c_Open_Click ()
  OpenScan
  UpdateDataOnMain
  fOpenFile.Hide
End Sub

'--------------- Set filedriver to selected pattern
Sub combo_Pattern_Change ()
  file1.Pattern = combo_Pattern.Text
End Sub

'--------------- Set filedrivers path to dir1.path
Sub Dir1_Change ()
  file1.Path = dir1.Path
  Call ShowFileName
End Sub

'------------------ Set dir1's path to drive1's drive
Sub Drive1_Change ()
  dir1.Path = drive1.Drive
End Sub

'------------- Display current file
Sub File1_Click ()
  ShowFileName
End Sub
```

```
'_____ Load file to preview if file is doubleclicked
Sub File1_DblClick ()
 On Error GoTo BasicError
 c_Screen.Picture = LoadPicture(text_FileName.Text)
 Exit Sub
BasicError:
 Resume Next
End Sub

'---------- Set start path and combo's
Sub Form_Load ()
 combo_Pattern.AddItem "*.bmp"
 combo_Pattern.AddItem "*.bmp;*.dat"
 combo_Pattern.ListIndex = 0
 drive1.Drive = "c:"
 dir1.Path = "c:\scanning"
End Sub

'_____ Opens the datafile and sets all parameters to new values
Sub OpenScan ()
Dim ActiveFile As String
On Error GoTo BasicError3
Dim x As Integer
 x = Len(text_FileName)
 ActiveFile = Left$(text_FileName, (x - 3)) & "dat"
Open ActiveFile For Input As 1
  Input #1, ScanType, NbrOfPixels, XStep, YStep, DelayTime, TimePassed
 ReDim PixelPicture(1 To NbrOfPixels * NbrOfPixels)
  For x = 1 To NbrOfPixels * NbrOfPixels
   Input #1, PixelPicture(x)
  Next x
 Close #1
 fScanning.Show
 fScanning.Command1.Value = True
TheEnd3:
Exit Sub
'------ If an error occurs
BasicError3:
Dim dummy As Variant
  dummy = MsgBox(ERRORINF & "          Tip:You have probably tried to open a file that don't exist ! (Not
                  all *.bmp's has an associated *.dat file)", 48, "Error")
  Resume TheEnd3
End Sub

'---------- Set screen when autosizing
Sub opt_AutoSize_Click (Value As Integer)
 If Value = True Then c_Screen.AutoSize = 1
 c_Screen.Width = 2700
 c_Screen.Height = 2700
End Sub

'---------- Set natural size
Sub opt_NaturalSize_Click (Value As Integer)
 If Value = True Then c_Screen.AutoSize = 0
End Sub

'---------- Display current filename
Sub ShowFileName ()
Dim Path As String
  Path = dir1.Path
  If Right$(Path, 1) <> "\" Then Path = Path & "\"
  text_FileName.Text = Path & file1.FileName
End Sub
```

```
'____________ Writes the new paramters on the Main form
Sub UpdateDataOnMain ()
 fMain.combo_Pixel.Text = NbrOfPixels & "x" & NbrOfPixels
 fMain.combo_Delay.Text = DelayTime & " ms"
 fMain.combo_Step(0).Text = XStep & " nm"
 fMain.combo_Step(1).Text = YStep & " nm"
End Sub
```

****

H) THE GLOBAL MODULE (NIGLOBAL.bas)

****

Here are the declarations of the Constants, Types, Functions and Globals that are needed for the *GPIB communication !*

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This module contains the variable declarations,
' constant definitions, and type information that
' is recognized by the entire application.
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Global ibsta As Integer
Global iberr As Integer
Global ibcnt As Integer
Global ibcntl As Long

Global buf As String

Global iarr(&H100) As Integer
Global Addresses(&H100) As Integer
Global IBresults(&H100) As Integer

' GPIB Commands

Global Const UNL = &H3F   ' GPIB unlisten command
Global Const UNT = &H5F   ' GPIB untalk command
Global Const GTL = &H1    ' GPIB go to local
Global Const SDC = &H4    ' GPIB selected device clear
Global Const PPC = &H5    ' GPIB parallel poll configure
Global Const GGET = &H8   ' GPIB group execute trigger
Global Const TCT = &H9    ' GPIB take control
Global Const LLO = &H11   ' GPIB local lock out
Global Const DCL = &H14   ' GPIB device clear
Global Const PPU = &H15   ' GPIB parallel poll unconfigure
Global Const SPE = &H18   ' GPIB serial poll enable
Global Const SPD = &H19   ' GPIB serial poll disable
Global Const PPE = &H60   ' GPIB parallel poll enable
Global Const PPD = &H70   ' GPIB parallel poll disable

' GPIB status bit vector :
'     status variable ibsta and wait mask

Global Const EERR = &H8000     ' Error detected
Global Const TIMO = &H4000     ' Timeout
Global Const EEND = &H2000     ' EOI or EOS detected
Global Const SRQI = &H1000     ' SRQ detected by CIC
Global Const RQS = &H800       ' Device requesting service
Global Const SPOLL = &H400     ' Board has been serially polled
Global Const EEVENT = &H200    ' An event has occured
Global Const CMPL = &H100      ' I/O completed
Global Const LOK = &H80        ' Local lockout state
Global Const RREM = &H40       ' Remote state
Global Const CIC = &H20        ' Controller-in-Charge
Global Const AATN = &H10       ' Attention asserted
Global Const TACS = &H8        ' Talker active
Global Const LACS = &H4        ' Listener active
```

```
Global Const DTAS = &H2        ' Device trigger state
Global Const DCAS = &H1        ' Device clear state

' Error messages returned in global variable iberr

Global Const EDVR = 0     ' System error
Global Const ECIC = 1     ' Function requires GPIB board to be CIC
Global Const ENOL = 2     ' Write function detected no listeners
Global Const EADR = 3     ' Interface board not addressed correctly
Global Const EARG = 4     ' Invalid argument to function call
Global Const ESAC = 5     ' Function requires GPIB board to be SAC
Global Const EABO = 6     ' I/O operation aborted
Global Const ENEB = 7     ' Non-existent interface board
Global Const EDMA = 8     ' DMA Error
Global Const EOIP = 10    ' I/O operation started before previous
                          ' operation completed
Global Const ECAP = 11    ' No capability for intended operation
Global Const EFSO = 12    ' File system operation error
Global Const EBUS = 14    ' Command error during device call
Global Const ESTB = 15    ' Serial poll status byte lost
Global Const ESRQ = 16    ' SRQ remains asserted
Global Const ETAB = 20    ' The return buffer is full
Global Const ELCK = 21    ' Address or board is locked


' EOS mode bits

Global Const BIN = &H1000   ' Eight bit compare
Global Const XEOS = &H800   ' Send EOI with EOS byte
Global Const REOS = &H400   ' Terminate read on EOS

' Timeout values and meanings

Global Const TNONE = 0     ' Infinite timeout (disabled)
Global Const T10us = 1     ' Timeout of 10 us (ideal)
Global Const T30us = 2     ' Timeout of 30 us (ideal)
Global Const T100us = 3    ' Timeout of 100 us (ideal)
Global Const T300us = 4    ' Timeout of 300 us (ideal)
Global Const T1ms = 5      ' Timeout of 1 ms (ideal)
Global Const T3ms = 6      ' Timeout of 3 ms (ideal)
Global Const T10ms = 7     ' Timeout of 10 ms (ideal)
Global Const T30ms = 8     ' Timeout of 30 ms (ideal)
Global Const T100ms = 9    ' Timeout of 100 ms (ideal)
Global Const T300ms = 10   ' Timeout of 300 ms (ideal)
Global Const T1s = 11      ' Timeout of 1 s (ideal)
Global Const T3s = 12      ' Timeout of 3 s (ideal)
Global Const T10s = 13     ' Timeout of 10 s (ideal)
Global Const T30s = 14     ' Timeout of 30 s (ideal)
Global Const T100s = 15    ' Timeout of 100 s (ideal)
Global Const T300s = 16    ' Timeout of 300 s (ideal)
Global Const T1000s = 17   ' Timeout of 1000 s (maximum)

' IBLN constants

Global Const ALL_SAD = -1
Global Const NO_SAD = 0

' IBEVENT constants

Global Const EventDTAS = 1
Global Const EventDCAS = 2
Global Const EventIFC = 3

' The following constants are used for the second parameter of the
' ibconfig function.  They are the "option" selection codes.

Global Const IbcPAD = &H1          ' Primary Address
```

```
Global Const IbcSAD = &H2          ' Secondary Address
Global Const IbcTMO = &H3          ' Timeout Value
Global Const IbcEOT = &H4          ' Send EOI with last data byte?
Global Const IbcPPC = &H5          ' Parallel Poll Configure
Global Const IbcREADDR = &H6       ' Repeat Addressing
Global Const IbcAUTOPOLL = &H7     ' Disable Auto Serial Polling
Global Const IbcCICPROT = &H8      ' Use the CIC Protocol?
Global Const IbcIRQ = &H9          ' Use PIO for I/O
Global Const IbcSC = &HA           ' Board is System Controller.
Global Const IbcSRE = &HB          ' Assert SRE on device calls?
Global Const IbcEOSrd = &HC        ' Terminate reads on EOS.
Global Const IbcEOSwrt = &HD       ' Send EOI with EOS character.
Global Const IbcEOScmp = &HE       ' Use 7 or 8-bit EOS compare.
Global Const IbcEOSchar = &HF      ' The EOS character.
Global Const IbcPP2 = &H10         ' Use Parallel Poll Mode 2.
Global Const IbcTIMING = &H11      ' NORMAL, HIGH, or VERY_HIGH timing.
Global Const IbcDMA = &H12         ' Use DMA for I/O.
Global Const IbcReadAdjust = &H13  ' Swap bytes during an ibrd.
Global Const IbcWriteAdjust = &H14 ' Swap bytes during an ibwrt.
Global Const IbcEventQueue = &H15  ' Enable/disable the event queue.
Global Const IbcSPollBit = &H16    ' Enable/disable the visibility of SPOLL.
Global Const IbcSendLLO = &H17     ' Enable/disable the sending of LLO.
Global Const IbcSPollTime = &H18   ' Set the timeout value for serial polls.
Global Const IbcPPollTime = &H19   ' Set the parallel poll length period
Global Const IbcEndBitIsNormal = &H1A  ' Remove EOS from END bit of IBSTA.
Global Const IbcUnAddr = &H1B          ' Enable/disable device unaddressing.
Global Const IbcSignalNumber = &H1C    ' Set UNIX signal number - unsupported
Global Const IbcHSCableLength = &H1F   ' Enable/disable high-speed handshaking.
Global Const IbcIst = &H20             ' Set the IST bit
Global Const IbcRsv = &H21             ' Set the RSV bit
Global Const IbcLON = &H22             ' Enable listen only mode.


' Constants that can be used (in addition to the ibconfig constants)
' when calling the IBASK function.

Global Const IbaPAD = &H1          ' Primary Address
Global Const IbaSAD = &H2          ' Secondary Address
Global Const IbaTMO = &H3          ' Timeout Value
Global Const IbaEOT = &H4          ' Send EOI with last data byte?
Global Const IbaPPC = &H5          ' Parallel Poll Configure
Global Const IbaREADDR = &H6       ' Repeat Addressing
Global Const IbaAUTOPOLL = &H7     ' Disable Auto Serial Polling
Global Const IbaCICPROT = &H8      ' Use the CIC Protocol?
Global Const IbaIRQ = &H9          ' Use PIO for I/O
Global Const IbaSC = &HA           ' Board is System Controller.
Global Const IbaSRE = &HB          ' Assert SRE on device calls?
Global Const IbaEOSrd = &HC        ' Terminate reads on EOS.
Global Const IbaEOSwrt = &HD       ' Send EOI with EOS character.
Global Const IbaEOScmp = &HE       ' Use 7 or 8-bit EOS compare.
Global Const IbaEOSchar = &HF      ' The EOS character.
Global Const IbaPP2 = &H10         ' Use Parallel Poll Mode 2.
Global Const IbaTIMING = &H11      ' NORMAL, HIGH, or VERY_HIGH timing.
Global Const IbaDMA = &H12         ' Use DMA for I/O.
Global Const IbaReadAdjust = &H13  ' Swap bytes during an ibrd.
Global Const IbaWriteAdjust = &H14 ' Swap bytes during an ibwrt.
Global Const IbaEventQueue = &H15  ' Enable/disable the event queue.
Global Const IbaSPollBit = &H16    ' Enable/disable the visibility of SPOLL.
Global Const IbaSendLLO = &H17     ' Enable/disable the sending of LLO.
Global Const IbaSPollTime = &H18   ' Set the timeout value for serial polls.
Global Const IbaPPollTime = &H19   ' Set the parallel poll length period
Global Const IbaEndBitIsNormal = &H1A  ' Remove EOS from END bit of IBSTA.
Global Const IbaUnAddr = &H1B          ' Enable/disable device unaddressing.
Global Const IbaSignalNumber = &H1C    ' Set UNIX signal number - unsupported
Global Const IbaHSCableLength = &H1F   ' Enable/disable high-speed handshaking.
Global Const IbaIst = &H20             ' Set the IST bit
Global Const IbaRsv = &H21             ' Set the RSV bit
```

```
Global Const IbaBNA = &H200            ' A device's access board.
Global Const IbaBaseAddr = &H201       ' A GPIB board's base I/O address.
Global Const IbaDmaChannel = &H202     ' A GPIB board's DMA channel.
Global Const IbaIrqLevel = &H203       ' A GPIB board's IRQ level.
Global Const IbaBaud = &H204           ' Baud rate used to communicate to CT box.
Global Const IbaParity = &H205         ' Parity setting for CT box.
Global Const IbaStopBits = &H206       ' Stop bits used for communicating to CT.
Global Const IbaDataBits = &H207       ' Data bits used for communicating to CT.
Global Const IbaComPort = &H208        ' System COM port used for CT box.
Global Const IbaComIrqLevel = &H209    ' System COM port's interrupt level.
Global Const IbaComPortBase = &H20A    ' System COM port's base I/O address.
Global Const IbaSingleCycleDma = &H20B ' Does the board use single cycle DMA?
Global Const IbaSlotNumber = &H20C     ' Board's slot number.
Global Const IbaLPTNumber = &H20D      ' Parallel port number
Global Const IbaLPTType = &H20E        ' Parallel port protocol

' These are the values used by the 488.2 Send command

Global Const NULLend = &H0         ' Do nothing at the end of a transfer
Global Const NLend = &H1           ' Send NL with EOI after a transfer
Global Const DABend = &H2          ' Send EOI with the last DAB

' This value is useds by the 488.2 Receive command

Global Const STOPend = &H100       ' Stop the read on EOI

' The following values are used by the iblines function.  The integer
' returned by iblines contains:
'     The lower byte will contain a "monitor" bit mask.  If a bit
'          is set (1) in this mask, then the corresponding line
'          can be monitored by the driver.  If the bit is clear (0),
'          then the line cannot be monitored.
'     The upper byte will contain the status of the bus lines.
'          Each bit corresponds to a certain bus line, and has
'          a corresponding "monitor" bit in the lower byte.

Global Const ValidEOI = &H80
Global Const ValidATN = &H40
Global Const ValidSRQ = &H20
Global Const ValidREN = &H10
Global Const ValidIFC = &H8
Global Const ValidNRFD = &H4
Global Const ValidNDAC = &H2
Global Const ValidDAV = &H1
Global Const BusEOI = &H8000
Global Const BusATN = &H4000
Global Const BusSRQ = &H2000
Global Const BusREN = &H1000
Global Const BusIFC = &H800
Global Const BusNRFD = &H400
Global Const BusNDAC = &H200
Global Const BusDAV = &H100

' This value is used to terminate an address list.  It should be
' assigned to the last entry.  (488.2)

Global Const NOADDR = &HFFFF

' Miscellaneous

Global Const S = &H8       ' parallel poll sense bit
Global Const LF = &HA      ' ASCII linefeed character
```

****

I) THE INIT MODULE (INIT.BAS)

****

Here are the declarations of the Constants, Types, Functions and Globals that are needed for the *TPOM program !*

```
'---------------------------------- TypeDeclarations for BitmapFileFormat
Type BITMAPFILEHEADER '--- 14 bytes
  bfType As Integer
  bfSize As Long
  bfReserved1 As Integer
  bfReserved2 As Integer
  bfOffBits As Long
End Type
Type BITMAPINFOHEADER '--- 40 bytes
  biSize As Long
  biWidth As Long
  biHeight As Long
  biPlanes As Integer
  biBitCount As Integer
  biCompression As Long
  biSizeImage As Long
  biXPelsPerMeter As Long
  biYPelsPerMeter As Long
  biClrUsed As Long
  biClrImportant As Long
End Type
Type RGBQUAD        '--- 8 bytes
  rgbBlue As Integer
  rgbGreen As Integer
  rgbRed As Integer
  rgbReserved As Integer
End Type

Global Const KEY_LEFT = &H25
Global Const KEY_UP = &H26
Global Const KEY_RIGHT = &H27
Global Const KEY_DOWN = &H28
Global Const ERRORINF = "You have made an error that is generally fatal. The error is intercepted though and
will not cause a crash of the program. Note that the program now may act in a peculiar way depending on what
you made wrong !"

'---------------------------------- Declare windows DLL for DelayTime checking
Declare Function GetTickCount Lib "User" () As Long
Global Const PD = 500 '---- When calling delayprocedure with PD(PiezoDelay) makes a 500 ms delay
'---------------------------------- Declare windows DLL for joystick handling and the structure

Type JoyInfo
 XPos As Integer
 YPos As Integer
 zPos As Integer
 button As Integer
End Type


Declare Function JoyGetPos Lib "MMSYSTEM" (ByVal uJoyID As Integer, lpCaps As JoyInfo) As Integer

'---------- Declaration of the default variables used to set the instrument's (Starting "g" for recognition!)
Global gFilter, gSetFrq, gHertz, gResolution, gSlope  As String
Global gVoltOrUm1, gVoltOrUm2, gVoltOrUm3
Global gOffsetOn, gExpandx10, gOutDisplay As Integer
Global gSensitivity, gInternal, gTimeConst, gTuneValue, gReject As Integer
Global gReadOutput, gFx2, gInput1, gInput2, gInput3 As Integer
Global gOffsetValue As Variant
```

```
'---------- Declaration of variables used for datechecking
Global gSavedDate, gNbrOfSavings
'---------- General variables   IMPORTANT!
Global Piezo_Number As Integer          '---- GPIB-address of piezo-controller
Global Lock_Number As Integer           '---- GPIB-address of lock-in amplifier
Global NbrOfPixels As Integer           '---- Dimension of the picture
Global PixelPicture() As Integer        '---- Array containing the pixels of the picture
Global ErrorCounter As Integer          '---- Stores the numbers of errors in GPIB-transmission
Global HardChecking As Integer          '---- Boolean defining the errorchecking method (hard/soft)
Global DelayTime As Long                '---- Delay between readings while scanning
Global XStep As Single                  '---- Decides the step in x-direction
Global YStep As Single                  '---- Decides the step in y-direction
Global SquareScan As Integer            '---- Decides type of scan (square/line)
Global ScanType As Integer              '---- Decides one of three types of SquareScans
Global GlobalMsg As String              '---- Contains ErrorMessage
Global Waiting As Integer               '---- Flag to show serial-pollning in progress
Global XAsBin As Long                   '---- Contains current x-position in binary value
Global YAsBin As Long                   '---- Contains current y-position in binary value
Global TimePassed As Long               '---- Contains the time for the current scan
Global CompX As Single                  '---- Compensates XStep for unlinearity if any
Global CompY As Single                  '---- Compensates YStep for unlinearity if any
```

*********GLOBAL FUNCTIONS AND SUBS **********

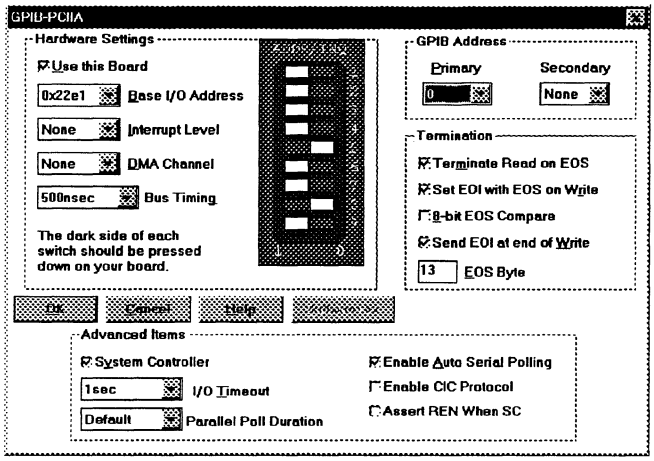
```
'------------ Return number of counts sent in an Errorbreak
Function AddIbcnt () As String
  AddIbcnt = Chr$(13) + Chr$(10) + "ibcnt = 0x" + Hex$(ibcnt%) + Chr$(13) + Chr$(10) + Chr$(13) +
Chr$(10)
End Function

'------------ Return explanation of Errorbit
Function AddIberr () As String
  If (ibsta And EERR) Then
    If (iberr% = EDVR) Then AddIberr = Chr$(13) + Chr$(10) + "iberr = EDVR (DOS Error)"
    If (iberr% = ECIC) Then AddIberr = Chr$(13) + Chr$(10) + "iberr = ECIC (Not CIC)"
    If (iberr% = ENOL) Then AddIberr = Chr$(13) + Chr$(10) + "iberr = ENOL (No Listener)"
    If (iberr% = EADR) Then AddIberr = Chr$(13) + Chr$(10) + "iberr = EADR (Address Error)"
    If (iberr% = EARG) Then AddIberr = Chr$(13) + Chr$(10) + "iberr = EARG (Invalid argument)"
    If (iberr% = ESAC) Then AddIberr = Chr$(13) + Chr$(10) + "iberr = ESAC (Not Sys Ctrlr)"
    If (iberr% = EABO) Then AddIberr = Chr$(13) + Chr$(10) + "iberr = EABO (Op. aborted)"
    If (iberr% = ENEB) Then AddIberr = Chr$(13) + Chr$(10) + "iberr = ENEB (No GPIB board)"
    If (iberr% = EOIP) Then AddIberr = Chr$(13) + Chr$(10) + "iberr = EOIP (Async I/O in prg)"
    If (iberr% = ECAP) Then AddIberr = Chr$(13) + Chr$(10) + "iberr = ECAP (No capability)"
    If (iberr% = EFSO) Then AddIberr = Chr$(13) + Chr$(10) + "iberr = EFSO (File sys.error)"
    If (iberr% = EBUS) Then AddIberr = Chr$(13) + Chr$(10) + "iberr = EBUS (Command error)"
    If (iberr% = ESTB) Then AddIberr = Chr$(13) + Chr$(10) + "iberr = ESTB (Status byte lost)"
    If (iberr% = ESRQ) Then AddIberr = Chr$(13) + Chr$(10) + "iberr = ESRQ (SRQ stuck high)"
    If (iberr% = ETAB) Then AddIberr = Chr$(13) + Chr$(10) + "iberr = ETAB (Table overflow)"
  Else
    AddIberr = Chr$(13) + Chr$(10) + "iberr = " + Str$(iberr%)
  End If
End Function


'------------ Return StatusByte
Function AddIbsta () As String
  sta$ = Chr$(13) + Chr$(10) + "ibsta = &H" + Hex$(ibsta%) + " ("
  If (ibsta% And EERR) Then sta$ = sta$ + " ERR"
  If (ibsta% And TIMO) Then sta$ = sta$ + " TIMO"
  If (ibsta% And EEND) Then sta$ = sta$ + " END"
  If (ibsta% And SRQI) Then sta$ = sta$ + " SRQI"
  If (ibsta% And RQS) Then sta$ = sta$ + " RQS"
  If (ibsta% And CMPL) Then sta$ = sta$ + " CMPL"
  If (ibsta% And LOK) Then sta$ = sta$ + " LOK"
```

```
   If (ibsta% And RREM) Then sta$ = sta$ + " REM"
   If (ibsta% And CIC) Then sta$ = sta$ + " CIC"
   If (ibsta% And AATN) Then sta$ = sta$ + " ATN"
   If (ibsta% And TACS) Then sta$ = sta$ + " TACS"
   If (ibsta% And LACS) Then sta$ = sta$ + " LACS"
   If (ibsta% And DTAS) Then sta$ = sta$ + " DTAS"
   If (ibsta% And DCAS) Then sta$ = sta$ + " DCAS"
   sta$ = sta$ + ")"
   AddIbsta = sta$
End Function

'_________ Transforms a binary value to um
Function BinToUm (bin As Long) As Single
  BinToUm = (bin / 65535) * 200
End Function

'--------- Delay for input WaitTime in milliseconds (using call to Windows runtime dll's)
Sub delay (WaitTime As Long)
 Dim ElapsedTime, Time1, Time2, x
  Time1 = GetTickCount()
  Do
    Time2 = GetTickCount()
    ElapsedTime = Time2 - Time1
  Loop Until ElapsedTime > WaitTime
End Sub

'_________ Draws the array PixelPicture at the scanning grid
Sub DrawPixelPicture ()
Dim rad, kol, x As Integer
  For rad = 1 To NbrOfPixels
  fScanning.Grid1.Row = rad
    For kol = 1 To NbrOfPixels
      x = x + 1
      fScanning.Grid1.Col = kol
      fScanning.Grid1.Picture = fScanning.PicClip1.GraphicCell(CInt(63 * PixelPicture(x) / 255))
    Next kol
  Next rad
End Sub

'---------- Builds the Errormessage and call for Error form
Sub ErrMsg (msg As String)
Dim info As String
   info = "If you don't understand the error codes you should refer to appendix B in the NI-488.2 User Manual"
   GlobalMsg = msg + AddIbsta() + AddIberr() + AddIbcnt() + info
   fErrorInfo.Show
End Sub

'---------- Test for error in transmission and in case update ErrorCounter (index=type of call before
calling ErrorTest)
Sub ErrorTest (value As Integer)
Dim msg As String
  If (ibsta And &H8000) <> 0 Then
     ErrorCounter = ErrorCounter + 1
     If HardChecking = True Then
        Select Case value
          Case 1: msg = "Problems in transmission computer-> instrument"
          Case 2: msg = "Problems in receiving values from instrument"
          Case 3: msg = "Problems in polling the lock-in amplifier": Waiting = False
          Case 4: msg = "Problems in connecting to Piezo:" + Chr$(13) + Chr$(10) + "(Are cables connected ? .
                        power turned on ?)"
          Case 5: msg = "Problems in connecting to Lock-In:" + Chr$(13) + Chr$(10) + "(Are cables connected ? ,
                        power turned on ?)"
          Case 6: msg = "Problems with initializion of the GPIB-card"
          Case 7: msg = "Problems with initializion of the instruments"
        End Select
        msg = msg + Chr$(13) + Chr$(10) + "An investigation of the transmission on the card gives the following
                                          result:" + Chr$(13) + Chr$(10)
```

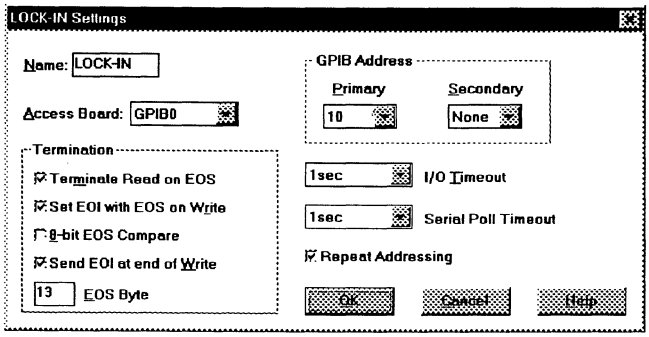
```
        Call ErrMsg(msg)
    End If
  End If
End Sub

'---------- Read piezo and return as binary. NOTE:Assumes using a 200 um positioner
Function GetPiezo_Bin (channel As Integer) As Long
Dim buffer As String
Dim range As String
 buffer = Space(10)
 Call Send(0, Piezo_Number, "I" & Str$(channel) & Chr$(13), DABend)
 delay (PD)
 Call Receive(0, Piezo_Number, buffer, STOPEnd)
 delay (PD)
 buffer = Right$(buffer, 8)
 GetPiezo_Bin = Val(buffer)
End Function

'---------- Read piezo and return as um
Function GetPiezo_um (channel As Integer) As Single
Dim buffer As String
Dim range As String
 buffer = Space(10)
 Call Send(0, Piezo_Number, "I" & Str$(channel) & Chr$(13), DABend)      ' Hämta startposition
 delay (PD)
 Call Receive(0, Piezo_Number, buffer, STOPEnd)
 delay (PD)
  range = Mid$(buffer, 2, 1)
  buffer = Right$(buffer, 8)
  If range = "H" Then
    GetPiezo_um = (Val(buffer) / 32768) * 200
    Else
    GetPiezo_um = (Val(buffer) / 32768) * 20
  End If
End Function

'---------- Get saved date and number of savings from FILE
Sub ReadDate ()
 Open "c:\windows\system\date.ini" For Input As 2
 Input #2, gSavedDate, gNbrOfSavings
 Close #2
End Sub

'---------- Read saved default parameters from FILE
Sub ReadDefaults ()
Open "c:\windows\system\tpom.ini" For Input As 1
Input #1, gSensitivity, gFilter, gSetFrq, gReject
Input #1, gTuneValue, gHertz, gInternal, gFx2
Input #1, gTimeConst, gResolution, gOffsetOn, gOffsetValue
Input #1, gExpandx10, gSlope, gReadOutput, gOutDisplay
Input #1, gVoltOrUm1, gVoltOrUm2, gVoltOrUm3
Input #1, gInput1, gInput2, gInput3
Close #1
End Sub


'---------- Save current date to FILE
Sub SaveDate ()
  Open "c:\windows\system\date.ini" For Output As 2
  Write #2, gSavedDate, gNbrOfSavings
  Close #2
End Sub
```

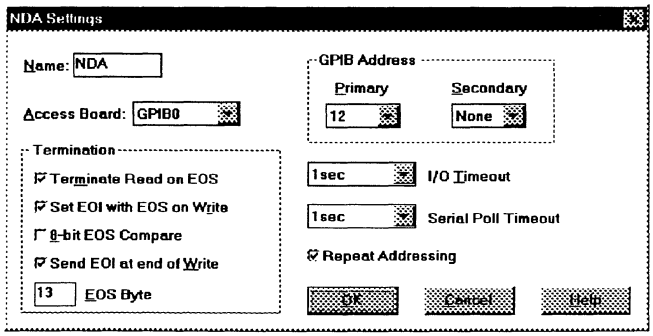
```
'---------- Save default parameters to FILE
Sub SaveDefaults ()
  Open "c:\windows\system\tpom.ini" For Output As 1
  Write #1, gSensitivity, gFilter, gSetFrq, gReject
  Write #1, gTuneValue, gHertz, gInternal, gFx2
  Write #1, gTimeConst, gResolution, gOffsetOn, gOffsetValue
  Write #1, gExpandx10, gSlope, gReadOutput, gOutDisplay
  Write #1, gVoltOrUm1, gVoltOrUm2, gVoltOrUm3
  Write #1, gInput1, gInput2, gInput3
  Close #1
End Sub

'---------- Transform um to binary value (0-65535)
Function UmToBin (um As Single) As Long
  UmToBin = CLng(um * 65535 / 200)
End Function

'---------- Make a serial poll and return when device signal "ready for new command" or breaking manually
Sub WaitForDevice ()
Dim PollByte As Integer
Dim turns As Integer
  Waiting = True
  Do
    turns = turns + 1
    Call ReadStatusByte(0, Lock_Number, PollByte)
    DoEvents
  Loop While ((PollByte And 1) = 0) And (Waiting = True)
  Waiting = False
End Sub
```

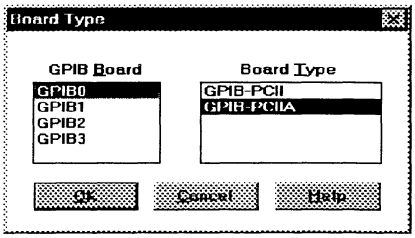
Appendix F The GPIB software configuration



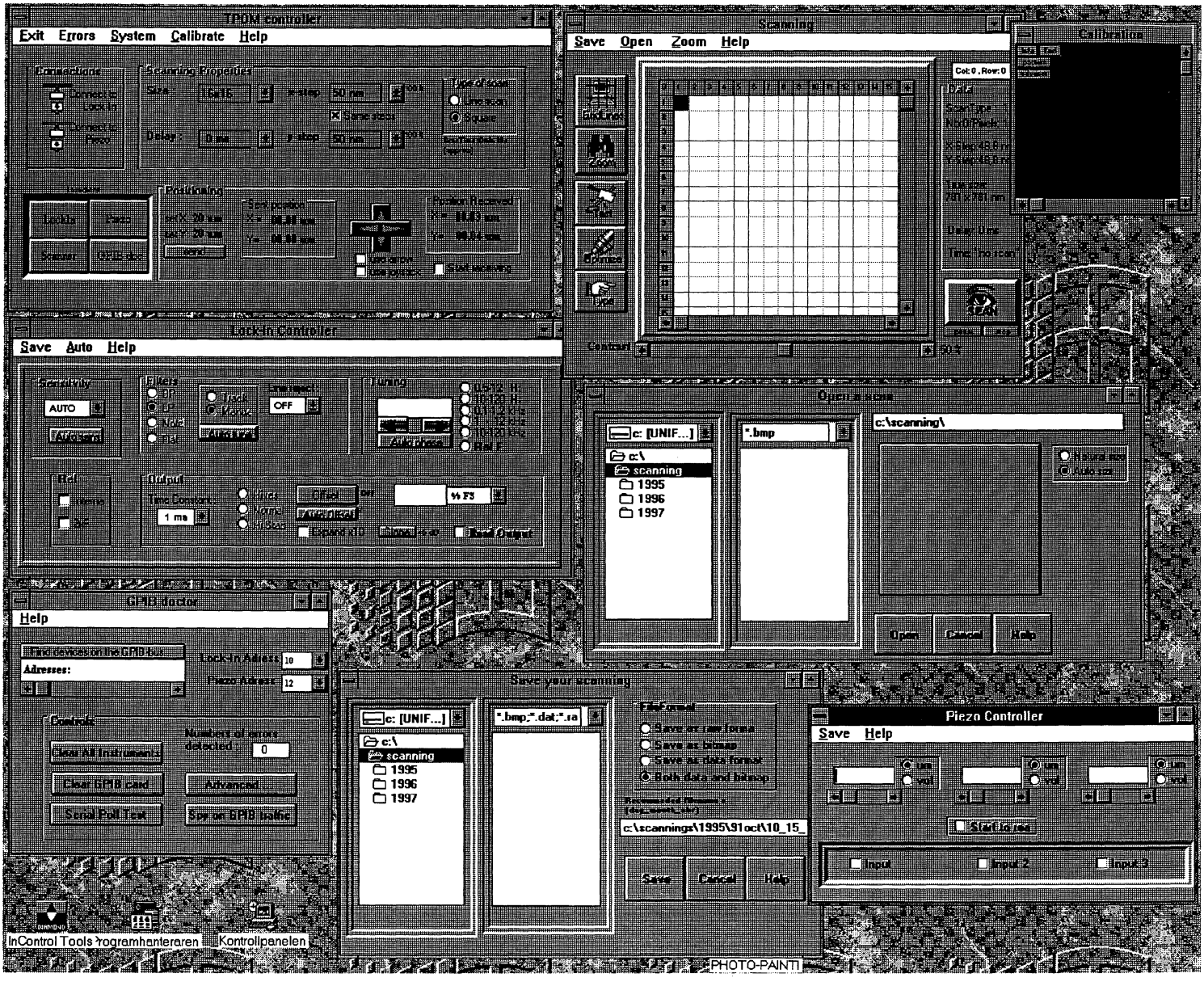
...

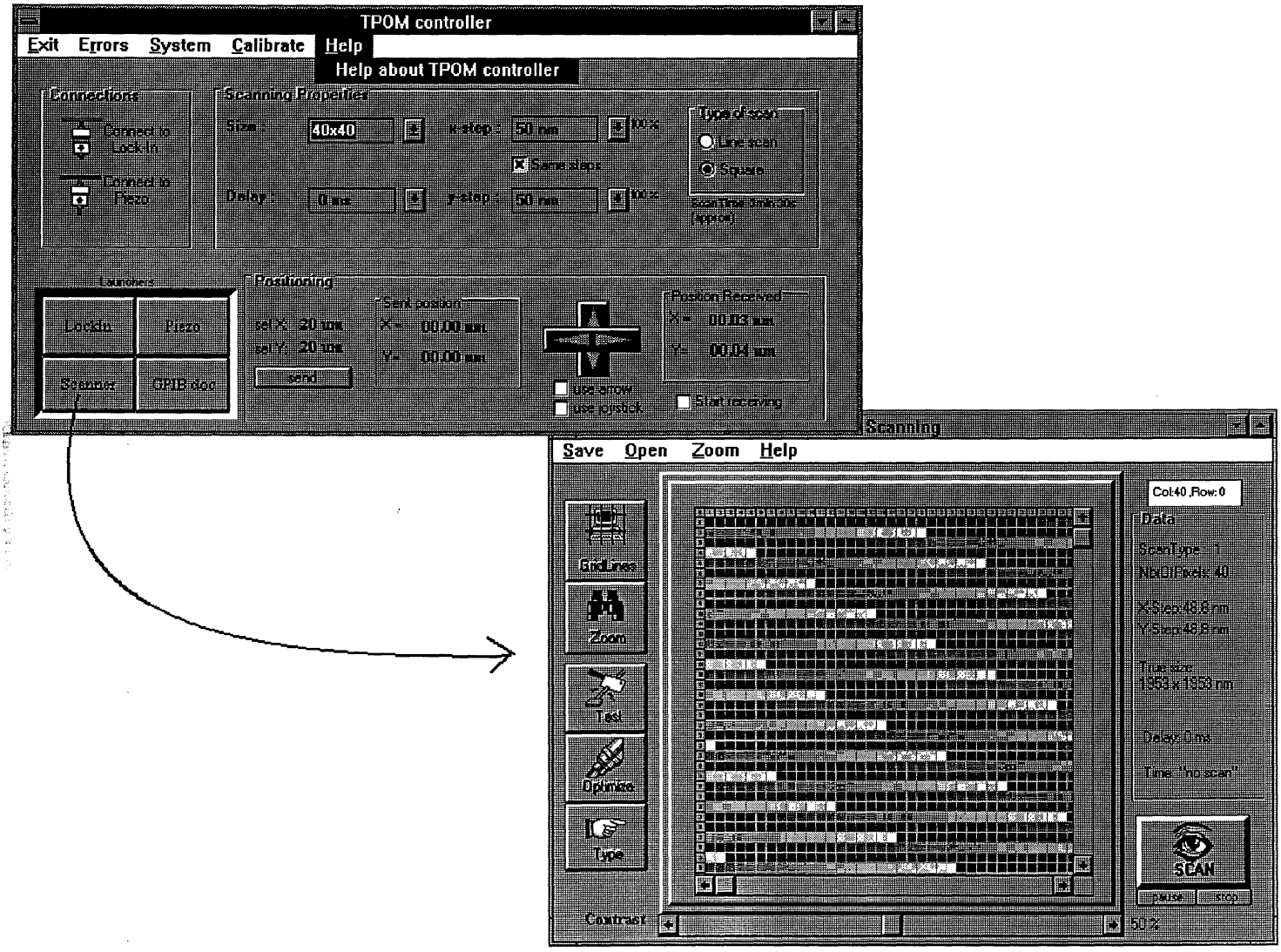






Appendix G The program interface




TPOM controller
Exit
Errors
System
Calibrate
Help
Help about TPOM controller
Connections
Connect to Lock-In
Connect to Piezo
Scanning Properties
Size :
40x40
x-step :
Same steps
Delay :
y-step :
Type of scan
Line scan
Square
Launchers
Lock-In
Piezo
Scanner
GPIB doc
Positioning
Sent position
X=
Y=
send
use arrow
use joystick
Position Received
Start receiving
Scanning
Save
Open
Zoom
Help
Grid Lines
Zoom
Test
Optimize
Type
Contrast
Col:40 ,Row: 0
Data
SCAN
pause
stop
50 %