AN INTERACTIVE STEERING SYSTEM FOR DIFFERENTIAL OPTICAL ABSORPTION SPECTROSCOPY MEASUREMENTS

Diploma Paper by

Stefan Spännare

Lund Reports on Atomic Physics, LRAP-117

Lund, January 1991

AN INTERACTIVE STEERING SYSTEM FOR DOAS MEASUREMENT

Diploma Paper

by

Stefan Spännare

CONTENTS PAGE

	Abstract	**1**
1.	**Introduction**	**2**
2.	**Principle of DOAS**	**3**
3.	**Interactive DOAS system**	**4**
4.	**Construction of steering system**	**7**
4.1	Light optimization to the spectometer	7
4.2	Stepper motor opto-switches	8
4.3	Adjustment of the PMT gain	9
4.4	Stepper motor interface	9
4.5	Secondary mirror stepper motor drive circuits	9
4.6	Weather data	10
5.	**Computer program description**	**11**
5.1	DOASLAMP	12
5.2	MULTMEAS	14
5.3	DOASASTR	19
5.4	EXTFKN	21
5.5	STEPMOT	21
5.6	DOASINT	24
5.7	SPECFKN	26
5.8	ASTRO	27
6.	**Measurement examples**	**28**
7.	**Acknowledgement**	**35**
8.	**References**	**36**
9.	**Appendices**	**37 -**

ABSTRACT

Atmospheric pollution is today a problem of great importance on regional and well as on a global level. To be able to monitor species in the atmosphere by measurement is therefore of great interest. A system for differential optical absorption spectroscopy (DOAS) has been developed at the Lund Institute of Technology. DOAS is a remote sensing technique, where the average concentrations of species (for example NO_2, SO_2, O_3 and NH_3) can be measured over distances up to 10 km. The system has four main parts, a broad band xenon-lamp as a light source, a telescope to receive light from the lamp, a spectrometer and a computer to evaluate measurement data. The telescope has two stepper motors to be able to turn the telescope towards an arbitrarily located lamp.

This diploma work has increased the possibilities of the DOAS system in particularly three ways:

1. The telescope stepper motors can be turned from the measurement computer to obtain measurement on many lamps in a sequence.

2. Fine adjustment of the secondary mirror of the telescope (by stepper motors) can be made to optimize the light intensity to the spectrometer despite backlash and imperfections in the main stepper motor driving.

3. Tracking of celestial objects (at least the sun and the moon) by the telescope to utilize natural light sources.

The steering program is written in Turbo Pascal and is now integrated with the earlier version of the DOAS-program.

1. INTRODUCTION

Atmospheric pollution is today a great problem depending on the damage they cause on nature and on human beings. Improved production technology and different methods of cleaning are of course the only real way to reduce the pollutants, but measurement and monitoring are important to get knowledge about the state of the atmosphere and to control that steps taken are effective. Different optical remote-sensing methods based on molecular spectroscopy are well adapted for such measurements [1-3]. Usually, the fact that the molecules are absorbing optical radiation at certain characteristic wavelengths is utilized. Thus the molecules spectral characteristics must be known from laboratory experiments. Further, a distinction can be made between active and passive remote-sensing techniques. Passive techniques use natural sources of radiation, usually scattered or reflected sunlight, while active techniques use artificial light sources such as lamps or lasers. Numerous different instruments measuring path averaged concentrations of pollutants have been developed, for example DOAS (Differential Optical Absorption Spectroscopy), Fourier transform spectroscopy, optical filter systems and dispersive correlation techniques.

The DOAS system described here briefly consists of a high pressure xenon lamp at the end of the measurement path, a turnable telescope to receive the light, a spectrometer and a computer to evaluate measured data. The goals of this diploma work was to improve the system in three ways: 1. The telescope should be turnable from the measurement computer to make it possible to measure on many lamps in different directions. 2. It should be able to optimize automatically the light intensity on the entrance slit of the spectrometer by turning the secondary planar mirror of the telescope. 3. Recording of spectra from celestial objects should be possible, requiring a tracking function for the telescope.

On the following pages a brief description of the principle of the DOAS technique and a discussion of the changes in the system are given. After that a more detailed description of the technical solutions follows and the steering program and some examples of measurement of air pollutants in Lund and spectra recorded from the sun and the moon are printed. Some more detailed information are given in the appendices.

2. THE PRINCIPLE OF DOAS

As mentioned above DOAS is an active technique, and as a light source a high-pressure xenon lamp is used. The light passes a given distance through the atmosphere, is received by a telescope, and its spectrum is recorded. Characteristic frequencies of the spectrum are absorbed with a strength depending on the concentration of the species and the length of the path. Accordingly, it is only possible to determine the path averaged concentrations of the pollutant. One problem is that the turbulence in the atmosphere causes fast variations in the atmospheric index of refraction and therefore also variations in the light intensity to the detector. To solve this problem the spectrum is recorded so fast (about 1/100 s) that the atmosphere can be regarded as frozen during this time. To get a good signal-to-noise ratio thousands of spectra are averaged during the measurement. DOAS is well adapted for measurement of several common atmospheric pollutants, for example sulphur dioxide (SO_2), nitric oxide (NO), nitrogen dioxide (NO_2), ozone (O_3), mercury (Hg), ammonia (NH_3) and also some short lived radicals such as NO_3, HNO_3 and OH [4-8]. The principles for evaluation of atmospheric spectra have been described by Platt and Perner [6]. First the background caused by the photo multiplier dark current is subtracted from the spectrum. To get the characteristic features of the species and to eliminate instrument-dependent influence a polynomial of degree 5 or 6 is fitted to the selected part of the spectrum. The spectrum is then divided by the polynomial and normalized. The evaluation relies on the Beer-Lambertian law and thus a logarithmic transformation of the spectrum is made. The concentrations of the species can then be evaluated by correlating with laboratory recorded reference spectra of the species.

3. INTERACTIVE DOAS SYSTEM

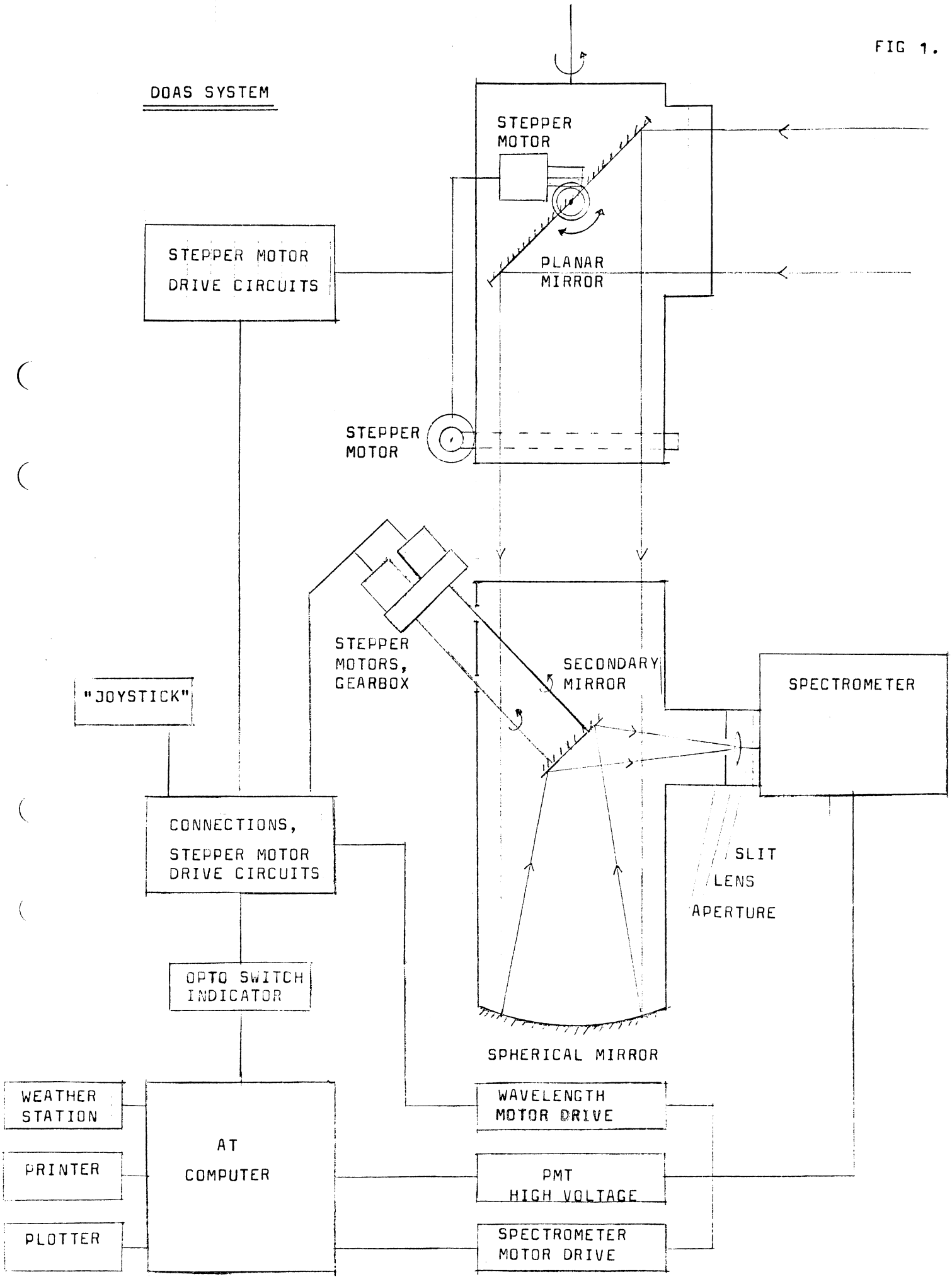
A schematic description of the DOAS system is found in Fig. 1. The light from a 500 W high pressure xenon lamp is received by a 30 cm diameter telescope via a planar mirror situated on the roof of the Physics Departments. The mirror is turnable horizontally and vertically by stepper motors with a resolution of 0.47 mrad/step and 0.63 mrad/step respectively. In the telescope the light is focused and brought via mirrors, aperture, lens and slit, into the Spex 500 M (f/4) spectrometer where it is spectrally divided with a 1200 grooves/mm grating blazed at 300 nm. The grating can be adjusted to a certain wavelength region between 0 nm and 1500 nm and each spectrum comprises a wavelength interval of approximately 40 nm. The rapid wavelength scanning is obtained by rotating a disk, with twenty 100 μm slits, through the spectrally dispersed light. The disk rotates with 5 r/s; thus the measurement time for each scan is 10 ms. The scan starts when the slit passes an infrared light barrier and then the light intensity as a function of time (or the position of the moving slit) is recorded with an EMI 9558 QA photo multiplier. The computer is of the IBM-compatible AT type and is used to collect and evaluate measurement data. A special multi-channel analyzer plug-in card developed for DOAS measurements is used, which stores each spectra in 1024 channels store each spectra digitally. The signal from the photo multiplier has first been amplified and analog-to-digital converted (12-bit). As mentioned before thousand of spectra are averaged on the card to obtain a good signal to noise ratio. A special interactive computer program to evaluate measurement data is used. The program is written in Turbo Pascal and gives among other things, possibilities to make measurement, to evaluate the concentrations of the species, to store spectra on hard disc and floppy and to transfer data to printer and plotter.

The DOAS system was originally made to measure over a certain path towards one lamp. The desire to measure towards many lamps required the telescope stepper motors to be connected to and steered from the measurement computer. The lamps can be of different brightness and have different distances to the telescope why it is important to receive as much light as possible to the spectrometer and that the range of the

AD-converters are effectively used. When the telescope was turned towards a lamp we found that the steps were so crude that the light not necessarily came through the slit of the spectrometer. The solution of this problem was to adjust the angel of the secondary planar mirror of the telescope to optimize the light intensity. To adapt the signal to the AD-converter the PMT voltage also was regulated from the computer.

The distribution of air pollutants are strongly dependent of the wind direction and speed. Therefore a small weather station has been connected to the computer to record wind data. A technical description of the changes in the system is given later. A steering computer program for the new functions also has been written and integrated with the original program. To make astro tracking possible a program has been written also to follow different celestial objects with the telescope. The program comprises astronomical algorithms and a steering program for the telescope. The programs are described under "computer program description" and the algorithms are listed in program form in Appendix 1.

FIG 1.


DOAS SYSTEM
STEPPER MOTOR
PLANAR MIRROR
STEPPER MOTOR DRIVE CIRCUITS
STEPPER MOTOR
STEPPER MOTORS, GEARBOX
SECONDARY MIRROR
SPECTROMETER
"JOYSTICK"
CONNECTIONS, STEPPER MOTOR DRIVE CIRCUITS
SLIT
LENS
APERTURE
OPTO SWITCH INDICATOR
SPHERICAL MIRROR
WEATHER STATION
PRINTER
PLOTTER
AT COMPUTER
WAVELENGTH MOTOR DRIVE
PMT HIGH VOLTAGE
SPECTROMETER MOTOR DRIVE

4. CONSTRUCTION OF THE STEERING SYSTEM

In this section a more detailed description of the secondary mirror steering, necessary electronics and connections for stepper motor steering, regulation of PMT voltage, spectrometer wavelength region and the connection of the weather station (4.1 to 4.6) is given.

4.1 Light optimization to the spectrometer

As mentioned above the telescope direction can be turned with a resolution of 0.47 mrad/step horizontally and 0.63 mrad/step vertically. The distance from the secondary mirror to the spectrometer slit is 0.30 m (see Fig. 1) which corresponds to a maximal movement of 2*0.63 mrad/step * 0.3 m = 380 μm/step over the slit. The width of the spectrometer slit is about 100 μm and the image of the light source over the slit has a diameter of about 300 μm. Thus the resolution of the main motors is not good enough to ensure that light passes the slit. The position of the image can also change due to mechanical and temperature drift. It was found that a fine positioning of the image over the slit by turning the secondary planar mirror with an accuracy of 5 μm would be suitable. Different methods to steer the mirror were discussed, for example piezo crystals and DC motors. To be able to optimize the position of the light source image over the slit a feed-back system must give information about the light intensity to the steering computer. One way to obtain this was to (via a beam splitter near the slit) illuminate a quadrant detector, another to use the already available measurement signal from the spectrometer. After some consideration I found it suitable to optimize on the spectrometer signal because then the level for spectra recording automatically is maximal and there is no loss in light due to the beam splitter. To steer the mirror stepper motors were used because they require simple electronics and they can easily be steered from the computer. The mounting of the secondary mirror was changed so that the mirror could be turned in two directions by micrometer-screws. The screws were placed in the opposite corners of a square with a side of 46 mm. An accuracy of 5 μm by the slit requires a turning of the mirror by 0.017/2 mrad =0.0085 mrad, which corresponds to a rising of $8.5*10^{-6}*$ 46 mm = 0.39 μm or $0.5/0.39*10^{-6}$ = 1280 steps per rewind for micrometer screws with a rising of 0.5 mm per rewind.

The stepper motors (KP 6M2-001 from Elfa) have 200 steps per rewind and a gear mechanism with a ratio of 6.4 is required. With a distance of √2*46 mm between the axes, two cog wheels with 60 cogs per wheel, modulo 1, could be used. The small cog wheels could not have less than 14 cogs because of the stepper motor axis diameter (6.35 mm). This gives a ratio of 4.29 or 857 steps per rewind which corresponds to an angular resolution of 0.025 mrad/step for the mirror or 7.6 μm/step at the slit. This is however good enough because during the optimization it is possible to let the motors take many steps at the time. To minimize backlash in the gear mechanism the small cog-wheels are made of steel and the big ones of volcanic fiber. The stepper motors and the gear box compose a rather bulky unit, and must be placed outside the telescope tube (to not block the incoming light) and drive the micrometer-screws by axes. Drawings are found in Appendix 4.

Specifications and manufacturers:

Axes connections:	BK 150306	MEKANEX
Bearings:	626-2Z/QE6	SKF
Cog wheels:	10014S, 10060V	MEKANEX
Micrometer screws:	148-207	MITUTOYO
Opto switches:	GP-411	Elfa
Stepper motors:	KP6M2-001	Elfa

4.2 Stepper motor opto-switches

Each stepper motor has been supplied with an opto switch indicating a fix zero-position (see Appendix 2.1). The signal from the opto witch is connected to the computer I/O-card and also switch on or off one of the LED (light emitting diodes) on the small connection box, to make seen that the opto-switches are working. The telescope motor LED are switched on while the secondary mirror LED are switched off when the opto switches are activated. The spectrometer wavelength motor has no opto switch connection. The opto switch of the vertical stepper motor on top of the telescope does not work if a bright lamp or the sun is shining into the telescope.

4.3 Adjustment of the PMT gain

To be able to measure towards lamps of various brightness situated at different distances from the telescope the signal from the PMT must be adapted to the right level for the computer AD-converters. This can easily be done by regulating the PMT voltage. Thus the computer has been equipped with an AD-DA card (PC ADDA-14 CARD FPC-011 from Josty Kit), the DA output is connected to the external input of the Oltronix A2.5K - 10 HR PMT power supply via an adjustable voltage divider (see Appendix 2.5). A switch has been connected so that external or internal regulation can be chosen. **NOTE:** This connection is "home made" and admits the power supply ONLY to be used with NEGATIVE OUTPUT VOLTAGE. Positive output causes the switch voltage to be many times overstepped.

4.4 Stepper motor interface

To obtain steering signals from the computer to the stepper motor drive circuits a PC-36 card from Elfa is used. The card has three programmable 8-bit ports. To protect the card from damage, drive circuits (ULN 2803) with open collector outputs, have been used as interface. A problem was that disturbances on the electric supply network (for example on and off switching of the room light) caused the stepper motors to take a few steps randomly. To remedy this, simple filters (a resistor and a capacitor, see appendix 2.3) were connected to the outputs of the PC-36 card and to the inputs of the stepper motor drive circuits. On and off switching of the stepper motor power supply still can cause randomly stepping, why the stepper motors should be turned to their opto switches to reach the right start positions.

4.5 Secondary mirror stepper motor drive circuits

The drive circuits (and power supply) of the secondary mirror stepper motors are placed in the main connection box. In the box is also a 5 V, 1 A power supply. Cable connections are found in Appendix 3.

4.6 Weather data

In measurements of atmospheric pollution the wind direction and wind speed are of great interest. Therefore the signal from a small weather station has been connected to the computer via the AD-card. The wind speed meter was damaged during the time of this diploma work, and because of that only the wind direction could be measured.

5. COMPUTER PROGRAM DESCRIPTION

When I started to write the computer program I did not suspect its extent. To comment every detail in the program is therefore rather useless, instead I have decided to give my view of what is important, for example the meaning of certain procedures, the record variables and describe how to use the program. The algorithms that I have found out myself are not claimed to be the best or the fastest and should of course be changed if better ideas turn up. One demand that was put on the program was that it could be composed with and make up a part of the old version of the DOAS program. The program was written in Turbo Pascal 3.0 but must be upgraded to version 5.5 before it was extended because of compiler limitations. The upgrading gave rise to numerous advantages for example the "include-files" could be changed to precompiled "units". Other pleasant experiences was the "longint variables" and the compiler itself. By the upgrading the head program was divided in two parts, one called INITIAL with all variable declarations and the new head program ROOT which contains the executive part of the previous head program and which calls the new units. ROOT also allocates all variables and variable records their potential default values and reads and writes the files which are necessary for some of the new units (i.e, STEPMOT, DOASLAMP and DOASASTR) on the hard disc. Certain parts of what is written below requires good knowledge about programming in Turbo Pascal and how the old version of the DOAS program (and the whole system) was working.

The new units are:
DOASLAMP
MULTMEAS
DOASASTR
EXTFKN
STEPMOT
DOASINT
SPECFKN
ASTRO

I describe them in this order (5.1 to 5.8). A program listing of ASTRO is found in appendix 1.

5.1 DOASLAMP

This program includes some procedures to direct the telescope towards different lamps and to type in data about the lamps. In the program data can be stored for up to 26 lamps in a vector. The vector can be read from and stored on the hard disc by the procedures init_doaslamp and exit_doaslamp, which have directory and file name as in parameters. For the present time the file DOASLAMP.DAT if found in the library TEL. To address the lamps they are named to one of the letters a-z. Each lamp record contains the following variables:

lamp_name:	The lamps real name (max 20 letters).
lamp_number:	1 - 26.
vert_position, hor_position:	The positions of the telescope stepper motors (in steps) vertically and horizontally.
mir_x_position, mir_y_position:	The positions of the secondary mirror stepper motors (in steps).
lamp_distance:	The lamps distance from the telescope in meters.
lamp_pmt_voltage:	Suitable PMT voltage (in volts) with respect to the lamps distance and brightness.

Lamp_distance and lamp_pmt_voltage are used in the unit MULTMEAS which is described later. Many lamp functions are accessible from LAMP menu by the function keys:

F1: Choose lamp.
F2: Delete lamp.
F3: Save position.
F4: Set PMT voltage.
F5: Int max (shift param).
F6: Horizon coordinates.
F7: Telescope (shift mirror) steering.
F8: Zero telescope.
F9: Zero mirror.
F10: Quit.

F1 Choose lamp: Lists the lamps in the vector with letter, name, position in steps, PMT voltage and distance. The position is also given as angle relative to north azimuth and altitude (in degrees). One condition is that the coordinates for north and zero altitude are correctly set. This can otherwise be done by the sun (see the ASTRO menu). By typing the letter of a lamp the telescope turns to the right position.

F2 Delete lamp: Deletes a lamp from the vector by typing it's letter.

F3 Save position: Coordinates and data for a lamp can be saved in the vector. The telescope must first be turned towards the lamp manually i.e. by steering from the computer keyboard or the joy stick. The letter, name, PMT voltage and distance are typed from the keyboard.

F4 Set PMT voltage: Sets the spectrometer PMT voltage from the computer keyboard.

F5 Int max (shift param): Light optimization as described for the unit DOASINT. With shift the parameters can be set.

F6 Horizon coordinates: The telescope can be directed to an arbitrary position by typing altitude and azimuth in degrees. The coordinates for north and zero altitude must be correctly set. The highest altitude is about 25 degrees with rain protection on the mirror dome and about 40 degrees without it.

F7 Telescope (shift mirror) steering: The telescope or the secondary mirror stepper motors can be manually turned 1, 10, 100 or 1000 steps at the time from the computer keyboard or the joy stick.

F8 Zero telescope: Zeros the telescope and the planar mirror stepper motors i.e. turns the stepper motors to their opto switches to get well defined zero positions. Should be done every time the program is used since the stepper motors can take a few steps randomly when the power is turned on. Must be done after a program crash since then the positions of the stepper motors have not been stored on the hard disc. This is done if the program is finished normally.

F9 Zero mirror: Zeros only the secondary mirror stepper motors.

F10 Quit: Exits the menu.

5.2 MULTMEAS

With MULTMEAS it is possible to measure concentrations of different species towards many lamps in sequence. The telescope is turning from lamp to lamp and measure on each lamp in a way that is settled beforehand. MULTMEAS has been rather confused and difficult to follow partly because it uses the unit GAUSS from the old version of the DOAS program. Further it is an advantage to be familiar with the old program since it is too extensive to describe here. First the menu and how to type a measurement sequence is described and which out data (to printer, hard disc and floppy) the program produces. Lastly a few comments about the program code.

The MULTMEAS (Multi lamp measurement) menu gives a number of choices with the function keys:

F1: Set measurement parameters.
F2: Set automatic evaluation sequence.
F3: Set evaluation parameters.
F4: Set intensity parameters.
F5: Start measurement.
F6: Set measurement sequence.
F7: Read measurement sequence from disc.
F8: Store measurement sequence on disc.
F10: Quit.

F1 - F3: With F1 - F3 a number of important parameters can be set for each measurement, for example number of cycles, spectrometer wavelength, the species to evaluate, the way the evaluation will be done and many others. These parameters must be settled beforehand and stored on the hard disc. The file that you name yourself, can at the present time only be stored when you exit the program. Normally you type and store one parameter file for each combination of species that will be

evaluated. The program then reads the parameters that are necessary for measurement on a certain lamp. This works exactly the same way as for the old version of the DOAS program.

F4: Set intensity parameters: Sets the parameters for optimization of the light intensity as described for the unit DOASINT.

F5: Start measurement: Starts the measurement.

F6: Set measurement sequence: Sets the measurement sequence and the measurement sequence parameters as described below.

First the measurement sequence parameters are set:

Measurements: The number of times the sequence is repeated. Measurements can be highly set since it's possible to break the measurement by typing 0.

Store interval: How often the spectra will be stored on the hard disc or floppy counted in measurements. Spectra from every measurement on a certain lamp are added until they are saved.

Time interval: The time when the lamp is changed (or the species on the same lamp) is a multiple of time interval counted in minutes from the latest integer hour. Time interval must be longer than the measurement time and telescope positioning time taken together. If time interval is set to zero the measurement will be performed without delay.

Wind measurement: The time the wind direction is measured from the small weather station (1 s - 59 s).

Evaluate: Determines whether the concentrations of the species will be evaluated or not (normally yes).

Print status: Determines whether the concentrations will be printed during the measurement or not.

Store spectra on disc: Determines if the spectra will be stored on the hard disk.

Store spectra on floppy: Determines if the spectra will be stored on floppy.

After typing RETURN it is time to type the lamp sequence itself. For each lamp the letter (a-z), number of repetitions (rep) and the name of the parameter file are specified. The name, coordinates, PMT voltage and distance for each lamp must have been settled beforehand (which can be done with LAMP MENU). Repetitions (rep) determines how many times a certain measurement is repeated before it is time to change lamp. The parameter file contains the parameters that are set with F1 - F3. When the sequence is complete you finish by typing 0.

F7: Read measurement sequence from disc: Reads a measurement sequence from the hard disc (At present in the library DOASTEXT).

F8: Store measurement sequence on disc: Stores a measurement sequence on the hard disc (At present in the library DOASTEXT).

F10: Quit: Exits the MULTMEAS menu.

Output data from MULTMEAS

SPECTRA

Each spectrum comprises 4.2 kbyte and it is eligible (see Set measurement sequence) whether it is stored on the hard disc and the floppy or not. During a measurement up to 144 spectra (or 605 kbyte) are produced every 24 hours. To not overfill the hard disc during lengthy measurements it might be better to not store the spectra.

PRINTER

On the printer the letter, time and date, wind direction, PMT voltage, regression coefficient, delta and the species, their concentrations and standard deviations are listed for each lamp. Regression coefficient and delta are evaluation parameters, that tell how well the reference spectra is fitted to the measured spectra. The lamp names and date are printed before the measurement starts.

TEXT FILES

The same data is also written on the text file DOASCONC.DAT in the library DOASTEXT. To be able to manipulate data given by the program (draw diagrams, make elegant transcriptions etc.) different computer programs can be used. QUATRO PRO is one used by the department. These programs are often capable of reading text files of "comma and quote" type, i.e. variables are surrounded by comma, strings also by quotes. Thus the program also writes the "comma and quote" file QUATCONC.DAT in the DOASTEXT library. Except the data mentioned above the time is also expressed in decimal days (24 hours) from the beginning of the measurement (To simply make concentration diagrams with time on the x-axis). When a new measurement starts these text files are filled from the end. To empty them copy the text file EMPTY also found in the DOASTEXT library.

The program code

Multmeas main procedure is named multi_lamp_measurement. Most of the others are sub procedures to make the variable handling easy. Some procedures for example voltage_maximum and get_wind_direction are independent. MULTMEAS uses some units from the old version of the DOAS program for example DISK (reading and writing spectra on hard disc and floppy) and GAUSS (the procedure evaluate to evaluate concentrations of the species). To collect the data that is necessary for a measurement two variable records are used: multi_meas_data and multi_meas_eval_data both are declared in the unit INITIAL. Multi_meas_eval_data get it's values when the procedure evaluate (from GAUSS) is used. This has been achieved by a small addition in the program since evaluate itself has no out parameters.

Here a few but important procedures are described:

voltage_maximum: Fits automatically the PMT voltage so that the signal from the spectrometer to the AD converters is between 70% and 80% of the maximal level. The fitting time is 10 s - 20 s.

In parameters:	
cycles:	The number of cycles the light intensity is measured during each iteration (100).
start-chnl, stop-chnl:	The channels in the spectrum the fitting comprise. The AD converters can be saturated outside this interval.
Out parameters:	
voltage:	PMT voltage after fitting.
Light_factor:	AD-level after fitting (range 0.0 - 1.0).

get_wind_direction: Gives an average value of the wind direction, counted from north, measured during a specified time. About 500 values are sampled per second.

In parameter:	
meas_seconds:	The time in seconds the wind direction is measured (1 s - 59 s).
Out parameters:	
average_wind_angel:	The wind direction from north in degrees.
wind_string:	The wind direction with letters, for example (N, S, NNW etc.).

julian_time: Is a function that uses the function jd, from the ASTRO unit, to express a certain point of time in julian time as decimal days (24 hours). In parameter is a variable of time_type (declared in INITIAL), i.e. time and date.

set_write_and_print_parameters: A procedure in multi_lamp_measurement. Gives multi_meas_data their values.

The other procedures consider feeding the measurement sequence, storing spectra on disc, output to printer and text files etc. and can hopefully be understood from the program code.Last but one in multi_lamp_measurement is the procedure multi_measurement which executes the measurement sequence. Since evaluation of data for one lamp is not done before measurement on the next, special consideration must be taken when measuring on the first or the last lamp. Among other things some temporary variables, which remember the measurement data of the previous lamp, are necessary. This has hardly not improved the readability of the program. Last in multi_lamp_measurement is the menu from which the procedures are called.

5.3 DOASASTR

With DOASASTR tracking of different celestial objects can be obtained by the telescope. All functions can be reached from the ASTRO menu by the function keys:

F1: Sun tracking.
F2: Moon tracking.
F3: Planet tracking.
F4: Star tracking.
F5: Spectra parameters.
F6: Store spectra.
F7: North with sun.
F8: Test.
F10: Quit.

F1 - F4: With sun, moon, planet and star tracking the telescope automatically follows the object in question. There is no feed back in the system but the coordinates are calculated by the algorithms from the ASTRO unit. In data to the procedures the longitude and latitude for Lund the time zone and time and date. Thus the computer time and date must be correctly set (not day light saving time (!) and with an error less than 15 seconds). The error in the position of the celestial object can be as much as 0.5 degrees for planets and the moon, far less for the sun and stars. The error is however never larger than that the object is seen by the telescope. The position can be adjusted from the computer

keyboard or with the joy stick. On the computer screen the horizontal coordinates (altitude and the angle from north azimuth) of the object are given in degrees. For the sun the refraction in the earths atmosphere also is written (in degrees). When tracking the sun a filter with about 2% transmittance must be used to get an appropriate amount of light to the spectrometer and not to burn the aperture. The coordinates of the sun can also be stepped in time to determine for example the points of time for sunrise and sunset, during the day in question. For a star (or galaxy etc.) the equatorial coordinates from a certain epoch must be feed into the computer.

When tracking is going on some functions can be chosen from the computer keyboard:

S/O: Spectra on/off.
Turns on or off recording of a spectra from the celestial object. In the on state number of cycles and spectra intensity are written on the screen.

1,3 4,6: PMT voltage.
Sets the PMT voltage the same way as with DOAS MENU.

z,x k,m: Position adjust.
With these keys the equatorial coordinates of the celestial object are adjusted. The deviations are called dh and ddelta (in degrees) on the computer screen. Adjustment also can be achieved by the joy stick.

0: Quit.
Returns to the ASTRO MENU.

+: (not transcribed)
As mentioned before the coordinates for north and zero altitude of the telescope stepper motors are stored. By typing + these coordinates are recalculated with respect to the changes in dh and ddelta. The best way to do this adjustment is by the sun or a star because their algorithms have the greatest accuracy.

F5: Spectrum parameters: Some parameters that deal with the spectral recording can be changed:

Part cycles: How many cycles the spectrum is measured before new coordinates are calculated (100).

Tot cycles: The total number of cycles the measurement comprise.

Zero spectrum: Determines whether the spectrum will be zeroed before the measurement starts or not.

F6: Store spectrum: Stores a spectrum on the hard disk in the directory DOASDATA.

F7: North with sun: This function is used the first time the coordinates for north and zero altitude are set. With stepper motor steering the telescope is directed towards the sun. After typing 0 the coordinates automatically are calculated. These are stored as the telescope stepper motors zero_position variables.

F8: Test: Tests some of the procedures from the ASTRO unit.

F10: Quit: Exits the ASTRO MENU.

5.4 EXTFKN

Extends Turbo Pascal with some functions: arcsin, arccos and sgn. Two functions for conversion between radians and degrees: rad and deg. Simple procedures for safe input of integer, longint and real variables. Finally, a procedure reading the computer time and date.

5.5 STEPMOT

Admits steering of up to 8 (can be changed to more) stepper motors from the computer via the PC-36 I/O-card. Only one stepper motor can be turned at the time. Each stepper motor requires three bits: step, direction and opto switch. Further it's possible to turn two stepper motors from from the computer keyboard or an external joy stick (also

connected to the I/O-card). Data for the motors are found in a vector which can be stored and read as a text file on the hard disc. At present the text file is STEPMOT.DAT found in the library TEL, but can be changed. The parameters in the vector must be typed on the text file the first time the program is used.

Each stepper motor record contains the following variables:

steps:	The stepper motors actual position in steps. If the opto switch is activated steps = 0.
steps_per_rew:	Steps per rewind by the stepper motor or what is driven by it.
zero_position:	The zero position of the stepper motors. It is used in the DOAS program to save the coordinates for north and zero altitude for the telescope motors.
max_delay, min_delay:	Max and min delay per step (max_delay > min_delay) The delay is obtained by a "for i:= 1 to delay do"-loop and varies with the computer clock frequency.
acceleration_steps:	The number of steps the stepper motor requires for acceleration from max_delay to min_delay (same for retardation).

The last parameters touch the communication with the I/O-card.

sm_in_port:	The address for the opto_switch in port.
sm_out_port:	The address for the step and direction out port.
step_bit:	Bit for step 0-7.
direction_bit:	Bit for direction 0-7.
opto_switch_bit:	Bit for opto-switch 0-7.
direction_1_level:	Determines the rotational direction of the motor for logical value 1 on the direction-bit (0 or 1).
opto_switch_on:	Determines whether the opto_switch is active high or low (0 or 1).

Data for the joy stick is also given in the text file:

number_of_keys:	Number of keys.
key_on:	Active high or low (0 or 1) for all keys.
key_bit:	Bit 0-7 is given for each key (vector).
key_in_port:	The address for the in port of each key.

The ports of the PC-36 I/O-card must be programmed whether they are in or out ports. This is done by sending a status byte to a certain address of the card. This address and the status byte are found first on the text file. After that is a number called number_of_stepper_motors in the program. It simply tells the program the real length of the stepper motor vector. Then follow the parameters (described above) for each stepper motor. Last on the text file is data for the joy stick port.

Text file example (STEPMOT.DAT):

```
1919    131
5
     7992        321      13400     5 2000  400    50 1918 1916  0  1  3  0  0
      178        219      20000     5 2200 2000    20 1918 1916  2  3  2  1  0
      390        415        857     5  200  100    10 1917 1916  4  5  5  1  1
      315        415        857     5  200  100    10 1917 1916  6  7  6  1  1
  -560000          0        200     1    1    0     1 1918 1918  5  4  1  0  1
4   1
4   1917
3   1917
1   1917
2   1917
```

The most important procedures to manipulate the stepper motors are:

init_stepmot: Reads port status, stepper motor data and joy stick data from the text file. In parameters are directory and file name.

exit_stepmot: Writes port status, stepper motor data and joy stick data on the text file. In parameters are directory and file name.

step: In parameters are one of the stepper motor vector elements (sm) and number of steps (number_of_steps). Turns the stepper motor as

many steps as given by number_of_steps. The variable steps is changed with the same value. The direction is given by the sign of number_of_steps. If the stepper motor activates the opto switch steps = 0.

steering: With this procedure it is possible to turn two stepper motors from the computer keyboard or the joy stick 1, 10, 100 or 1000 steps at the time. Parameters are two of the stepper motor vector elements and joy stick data.

5.6 DOASINT

Contains a number of procedures and functions that are directly tied to the DOAS-measurements, for example setting cycles, PMT voltage, spectrometer wavelength, light optimization etc. Some program parts are taken from the old version of the DOAS program but are here written in procedure form. The procedure names are in most cases self explanatory why only the procedures intensity_maximum and draw_intensity_graph are explained. The variables present_pmt_voltage and max_pmt_voltage are global and their declarations are found in the unit INITIAL.

procedure intensity_maximum

Optimizes the light to the spectrometer by steering the small secondary mirror of the telescope. The stepper motors are scanning a number of points in a matrix whose size (in steps) is settled beforehand. In each point the light intensity from the spectrometer is measured and then the mirror is turned to the point where the highest intensity was found. Finer adjustment can be obtained by optimizing in many steps with diminishing matrix size. The main disadvantage with this method is that the optimization takes rather long time (20 s - 40 s). Smarter algorithms that uses the gradient in light intensity are difficult to use because the signal is rather noisy, i.e. the intensity in a certain point varies from time to time because of the short averaging time. The procedure has an in-parameter of intensity_param_type with the following variables (Reasonable values are given in brackets).

int_cycles:	The number of cycles the light intensity is measured in each matrix point (1 cycle = 1/100 s) (100).
int_steps:	The number of stepper motor steps between the points in the matrix (50 - 100).
max_itera:	The number of iterations during the optimization (1-2).
order:	The matrix side in points (2*order+1) (3 i.e. 25 points in the matrix).
step_factor:	The ratio in matrix size between the iterations (0.5).
start_intensity, stop_intensity:	The light intensity before and after the optimization (dummy values).
write_int:	Logical variable if the result of the optimization is to be written on the computer screen. If the value is true the intensity per cycle and the coordinates of the planar mirror stepper motors(in steps) are written for each matrix point (true).

Int_cycles, int_steps, max_itera and step_factor can be changed by the procedure set_intensity_parameters. The default values of all variables are set in the root program.

procedure draw_intensity_graph

Draws a three dimensional picture on the computer screen of the light intensity as a function of the positions of the planar mirror stepper motors (40 * 40 points). The procedure uses the GSX graphics.

In parameters:

cycles:	Number of cycles in each point (5).
steps:	Number of stepper motor steps between the points (5).
mode:	1 = real graph, 0 = demonstration (1).

With the values given above a picture takes about 6 minutes to draw.

5.7 SPECFKN

In the computer there is a special card (the DOAS card) to collect data from the spectrometer. Data is stored on the card in 1024 channels which can be read from the program to a real vector (a spectrum) with 1024 elements. A spectrum is of the type DOAS_real_size and is declared in the unit INITIAL. SPECFKN contains a number of functions and procedures to manipulate spectra and the DOAS card:

set_cycles: Sets how many cycles (1/100 s) data will be collected to the DOAS card (1 - 30000).

zero_memories: Zeros the memories (all the channels) of the DOAS card to collect new data.

add_memories_to_spectra: Adds the memories of the DOAS card to a spectrum (vector) in the program.

zero_spectra: Zeros a spectra (vector) in the program.

sum_spectra: Function that gives an integral value over all the 1024 elements in a spectrum.

add_spectra: Adds two spectra to a third.

start_sampling: Starts sampling of data to the DOAS card as many cycles that are set with set_cycles. The procedure can be broken by typing 0.

finish_sampling: The program is waiting for the DOAS card to be ready with sampling of data. The procedure can be stopped by typing 0. The program (the computer) can be used for other purposes while the DOAS card is sampling, i.e between start_sampling and finish sampling in the program.

5.8 ASTRO

Procedures for astronomical applications. Among other things the positions for the sun, the moon, the planets and stars can be calculated in equatorial and horizontal coordinates. The error in the calculations is about 0.5 degrees in the worst case. Each procedure is commented in the program. The algorithms are from [9] and [10]. Program list is found in Appendix 1.

6. MEASUREMENT EXAMPLES

The DOAS-system was arranged with three measurement paths overlooking the city of Lund, Sweden (pop. 50000). The xenon lamps were placed in the distances 2000 m, 400 m and 1600 m as shown in Fig. 2. In Fig. 3 a and 4 a raw spectra from SO_2 and NO_2 are shown. The spectral features of the lamp dominate each spectrum. In the NO_2 spectrum two lamp depending peaks are clearly seen. A mask in the spectrometer limits the wavelength region for each raw spectrum to 40 nm. A wavelength interval where the species has a characteristic absorption spectrum is selected. In Fig. 3 a the absorption peaks of SO_2 are clearly seen between 300 nm and 310 nm. To eliminate the instrument dependent features in the raw spectrum (for example the spectral profile of the xenon-lamp) the raw spectrum is divided with a polynom of degree 5 or 6 which has been fitted to the selected part of the spectrum Fig. 3 b,c and 4 b,c. Regions with strong instrument dependent peaks must be avoided because they make the polynomial fitting more difficult. The intensity of the spectrum depends exponentially of the concentration of the species and the length of the measurement path (Beer-Lamberts law) why a logarithmic transformation of the spectrum is made. The concentrations of the species are then evaluated by correlating with laboratory recorded reference spectra Fig. 3 d and 4 d. Concentrations from the three paths during one night are shown in Fig. 5. Concentrations from the 2000 m path during 24 hours are shown in Fig. 6. The diagrams have been produced with a computer program called QUATRO.PRO from the raw data files produced with the DOAS program. An example of printed raw data from the DOAS program is found on the next page.

Using the astro tracking function a solar spectrum and a moon spectrum around 430 nm have been recorded see Fig. 7 a,b. The Frauenhofer absorption lines are clearly seen, also that the moon light is reflected sun light. The moon spectrum was recorded with lower spectral resolution.

Raw data example from the DOAS program. Date, time, wind direction from north, PMT voltage, light intensity factor (LF), regression coefficient, delta (Del), the species and their concentrations and standard deviation are printed. The regression coefficient and delta are reference spectrum fitting parameters.

==========

1990-10-26 13:39:15 spectra not stored

A Skattehuset so2.30
A Skattehuset no2.30

==========

Date	Time	Wind	PMT	LF	Reg	Del	Spec	Conc	Sigma		
A: 10-26	13:40	119	1070	0.735	0.989	-1	SO2:	33.3	0.41	O3:	67.3 21.53
A: 10-26	13:51	124	830	0.730	0.857	-2	NO2:	21.0	0.76		
A: 10-26	14: 1	120	1080	0.894	0.985	-1	SO2:	29.7	0.42	O3:	34.1 21.96
A: 10-26	14.11	106	970	0.802	0.599	-5	NO2:	27.3	2.18		
A: 10-26	14:21	100	1240	0.642	0.990	-2	SO2:	29.7	0.37	O3:	184.6 19.30
A: 10-26	14:31	110	870	0.981	0.916	-4	NO2:	11.6	0.31		
A: 10-26	14:41	95	1050	0.797	0.989	-2	SO2:	19.0	0.23	O3:	73.0 12.25

CITY OF LUND

Fig. 2.

EVALUATION OF DOAS SPECTRUM

SULPHUR DIOXIDE

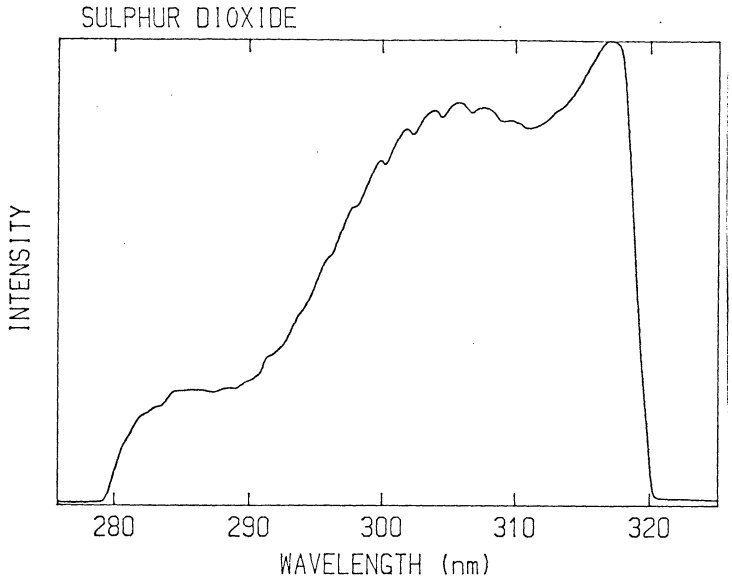


Fig. 3 a. RAW SPECTRUM

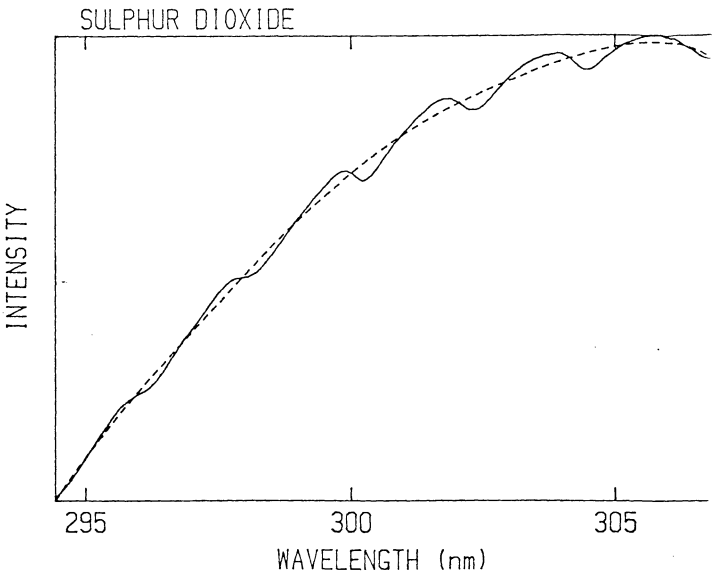


Fig. 3 b. RAW SPECTRUM (solid line) POLYNOMIAL FIT (dashed line)

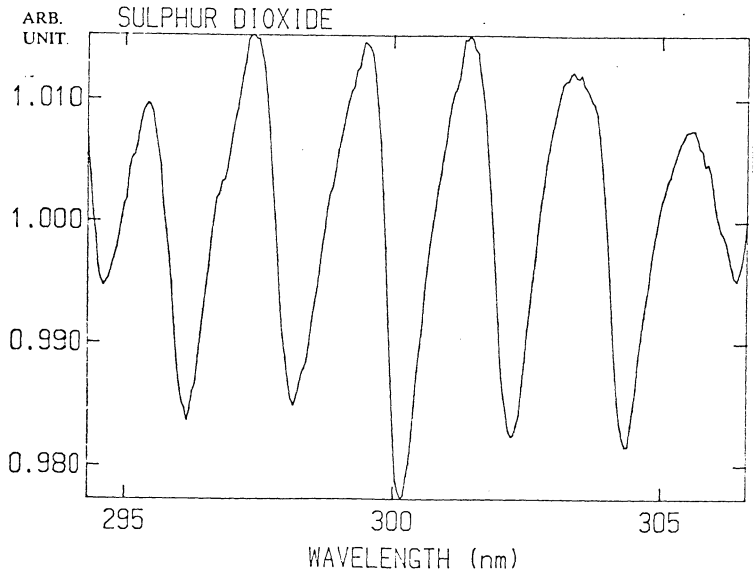


Fig. 3 c. NORMALIZED SPECTRUM (raw spectrum divided with polynomial fit)

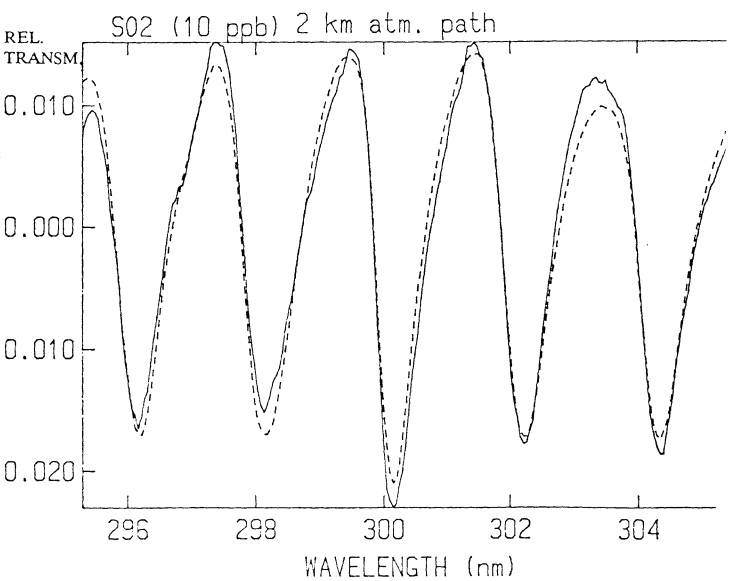


Fig. 3 d. DIFFERENTIAL SPECTRUM after logarithmic transformation and correlation with reference spectrum.

EVALUATION OF DOAS SPECTRUM

NITROGEN DIOXIDE

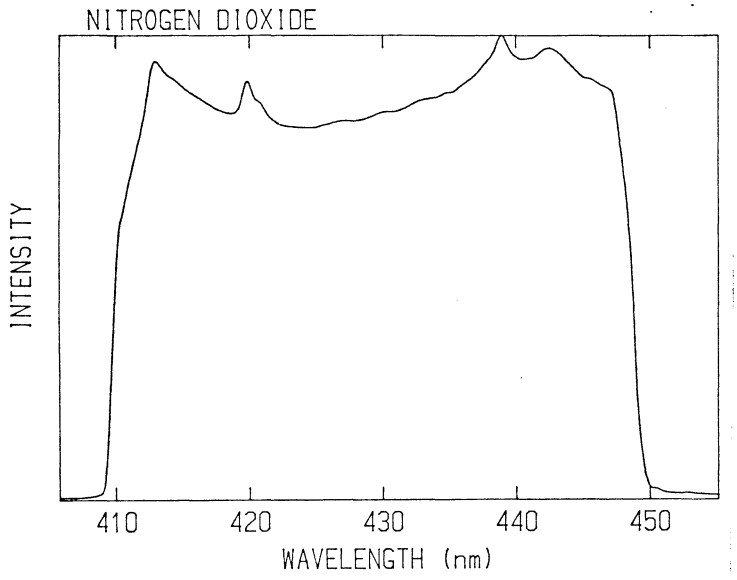


Fig. 4 a. RAW SPECTRUM

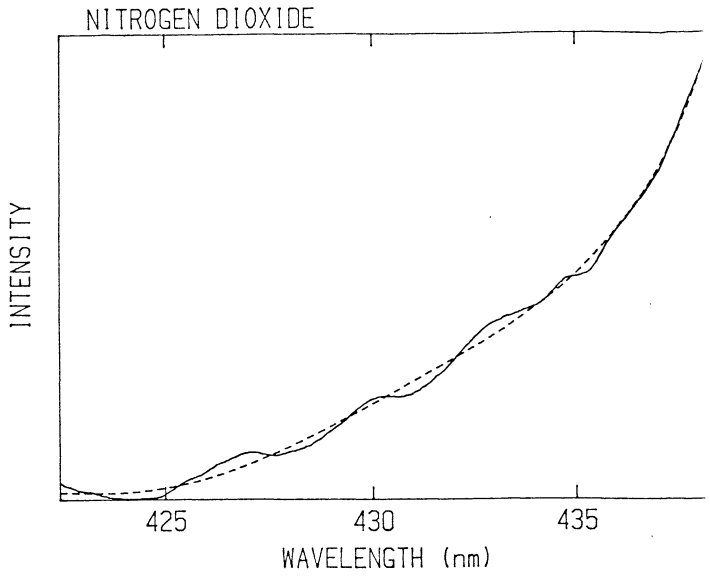


Fig. 4 b. RAW SPECTRUM (solid line) POLYNOMIAL FIT (dashed line)

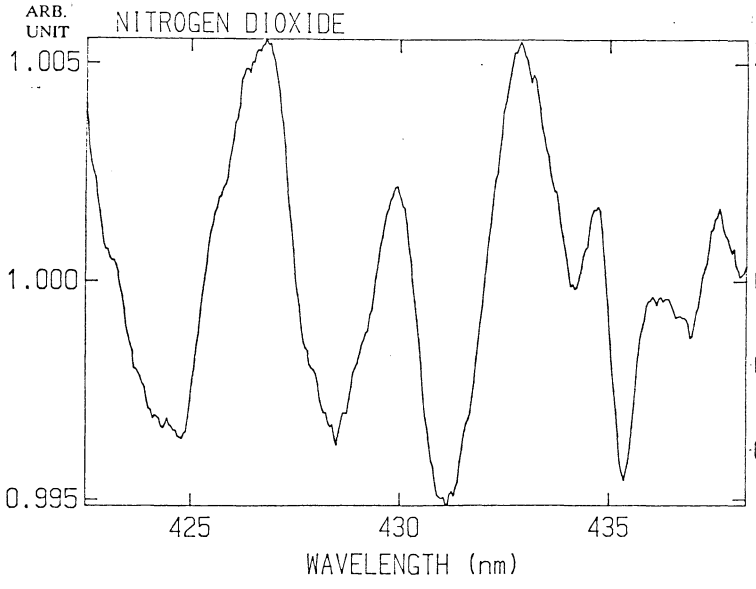


Fig. 4 c. NORMALIZED SPECTRUM (raw spectrum divided with polynomial fit)

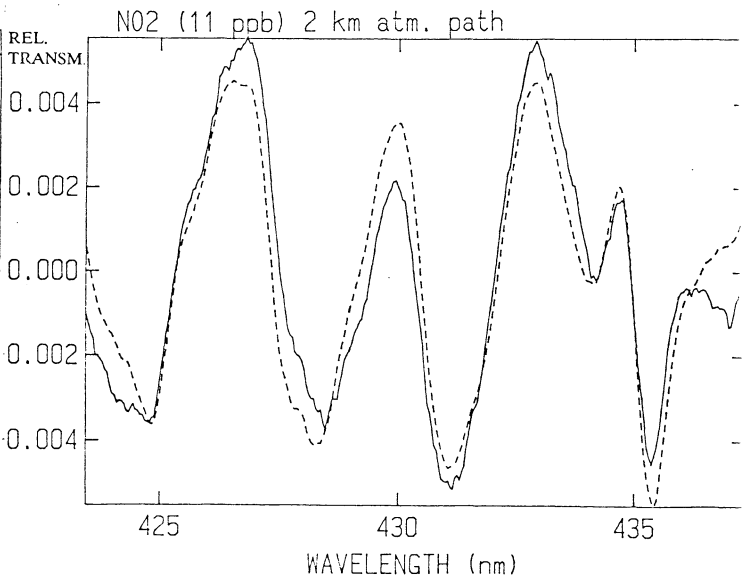


Fig. 4 d. DIFFERENTIAL SPECTRUM after logarithmic transformation and correlation with reference spectrum.

Fig. 5.

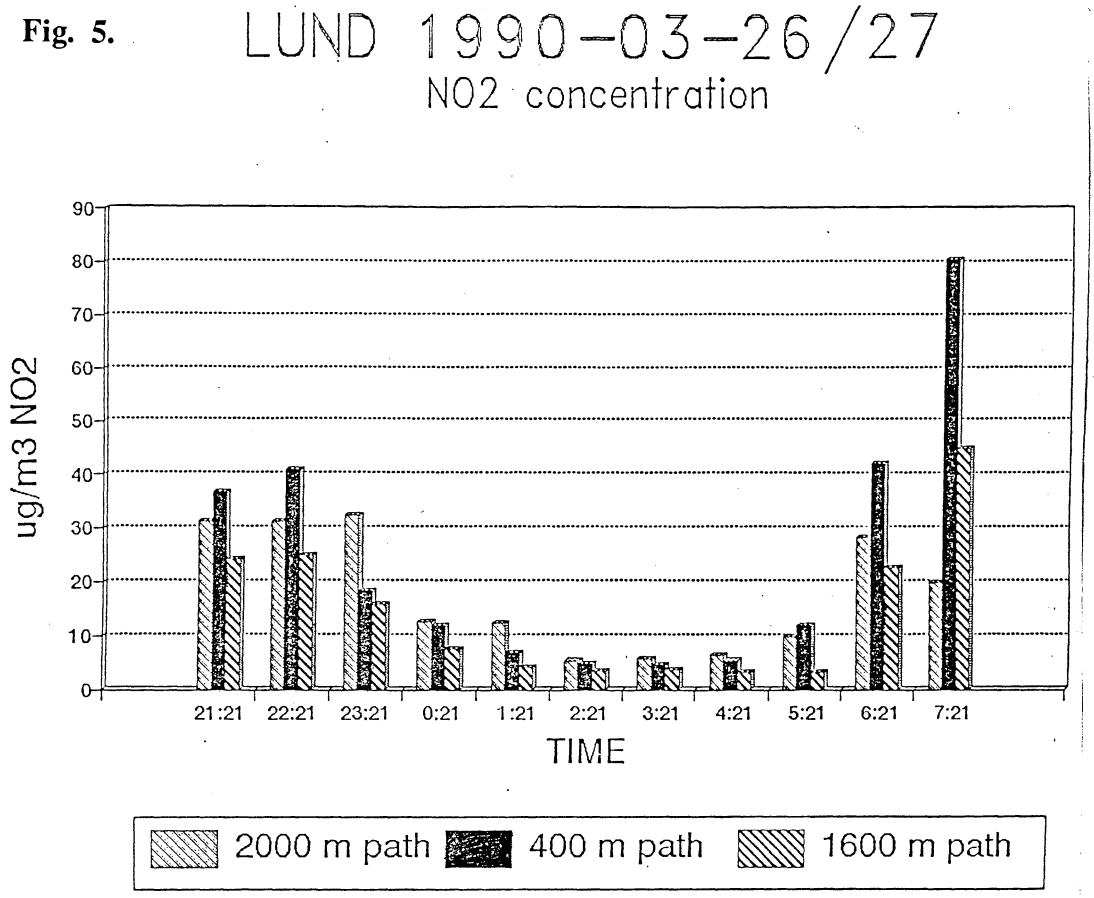

LUND 1990-03-26/27
NO2 concentration
ug/m3 NO2
90
80
70
60
50
40
30
20
10
0
21:21
22:21
23:21
0:21
1:21
2:21
3:21
4:21
5:21
6:21
7:21
TIME
2000 m path
400 m path
1600 m path

Fig. 6.

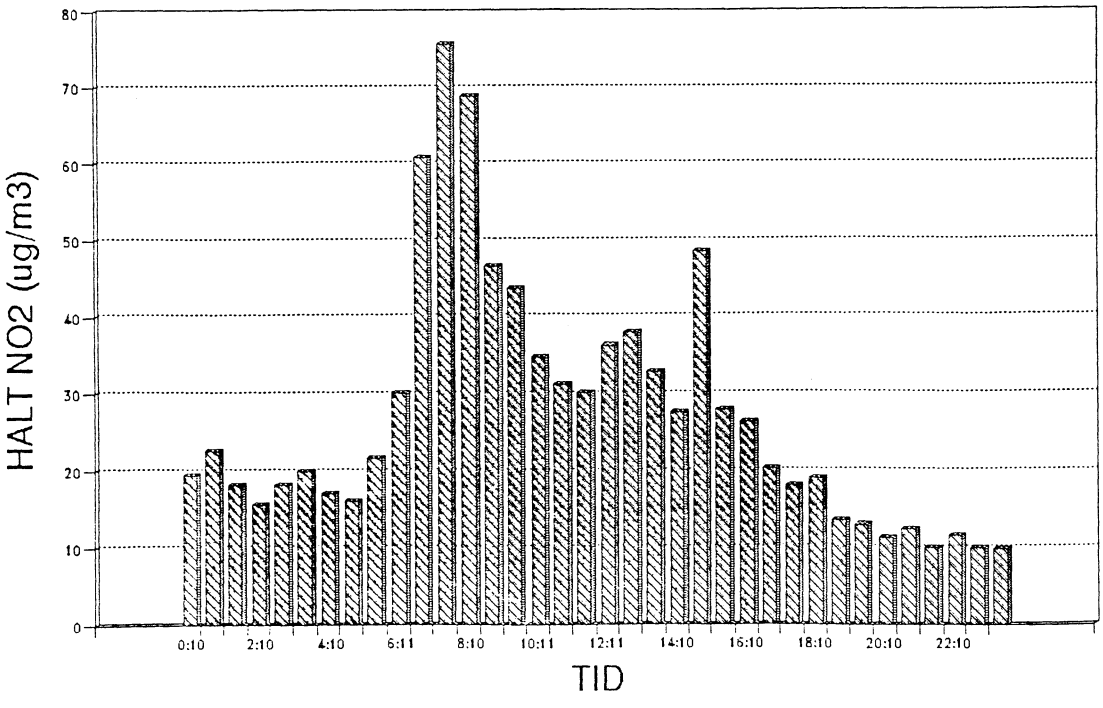

SKATTEHUSET 90-04-03
Dygnsmedelvärde: 27.1 ug/m3
HALT NO2 (ug/m3)
80
70
60
50
40
30
20
10
0
0:10
2:10
4:10
6:11
8:10
10:11
12:11
14:10
16:10
18:10
20:10
22:10
TID

Fig. 7 a.

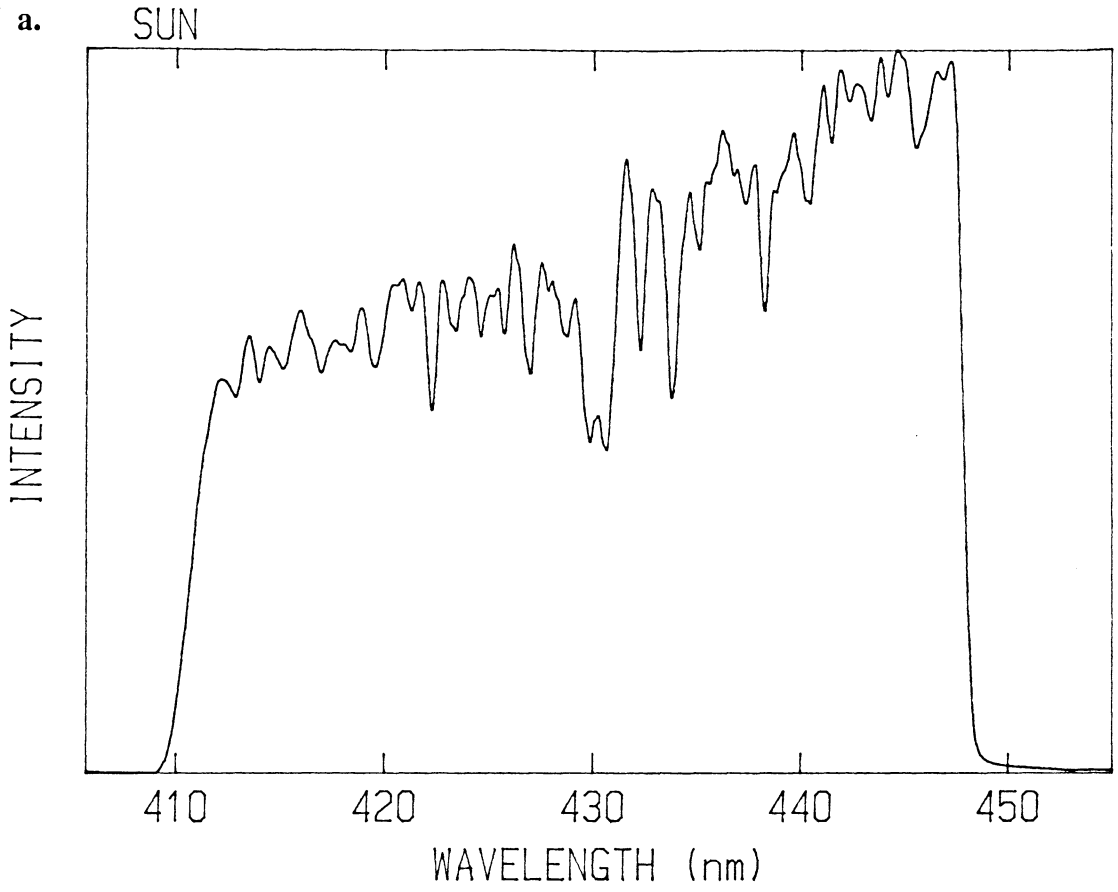
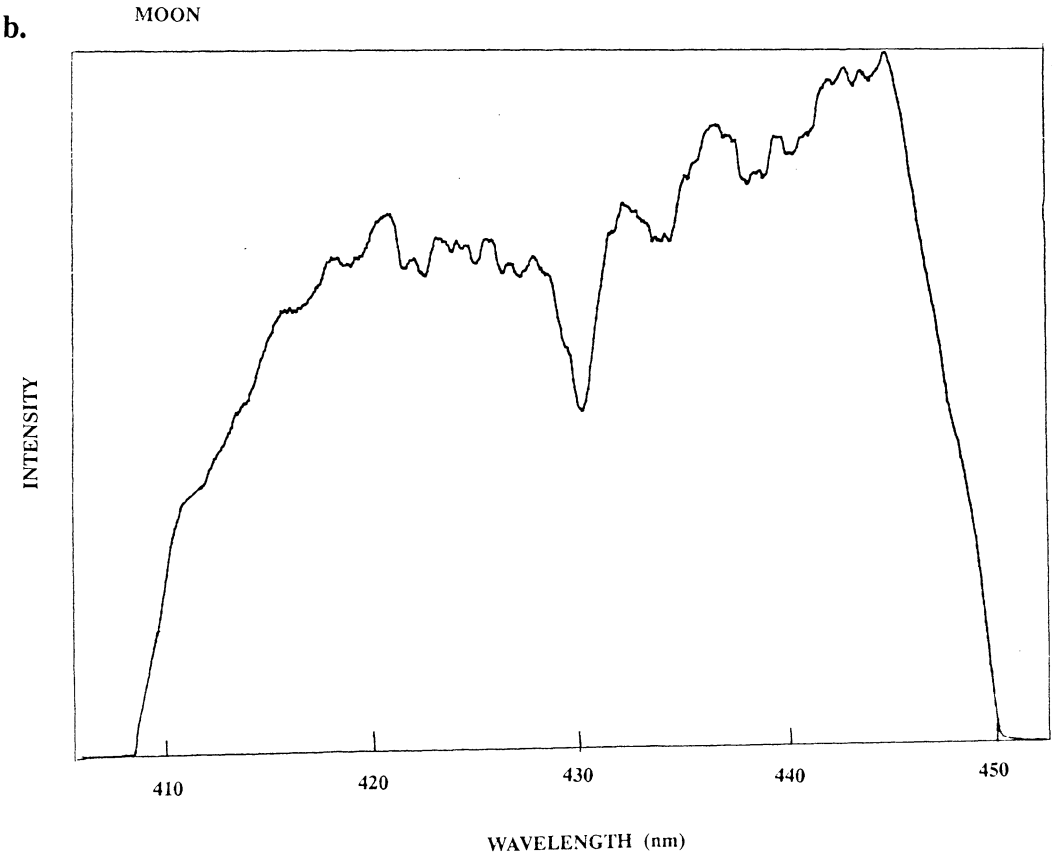

SUN
INTENSITY
410
420
430
440
450
WAVELENGTH (nm)

Fig. 7 b.


MOON
INTENSITY
410
420
430
440
450
WAVELENGTH (nm)

ACKNOWLEDGEMENT

I am most grateful to all those people who in different ways have helped me with this diploma work. I would especially like to thank my instructor Ph. D. Hans Edner and the Ph. D. graduate students Pär Ragnarsson and Eva Wallinder for all help and for a pleasant time together. I would also like to thank professor Sune Svanberg for introductory ideas and suggestions, Per Olsson for help especially with the computer program and Åke Bergquist for help with the electronics. Last but not least I would like to thank Rolf Olofsson for a first class work with the gear box at the mechanical work shop of the department.

References

1. D.K. Killinger and A. Mooradian, Eds., Optical and Laser Remote Sensing Techniques (Springer, Berlin, Heidelberg 1983).

2. R.M. Measures, Laser Remote Sensing (Wiley-Interscience, New York 1984).

3. S. Svanberg, "Optical Environmental Monitoring", in M. Inguscio and W. Demtröder, Eds., Applied Laser Spectroscopy (Plenum Press, New York 1990).

4. U. Platt, D. Perner and H.W Pätz, " Simultaneous Measurement of Atmospheric CH_2O, O_3, and NO_2 by Differential Optical Absorption", J. Geophys. Res. **84**, 6329 (1979).

5. U. Platt and D. Perner, "Direct Measurement of Atmospheric CH_2O, HNO_3, O_3 and SO_2 by Differential Absorption in the near Infrared", J. Geophys. Res. **85**, 7453 (1980).

6. U. Platt and D. Perner, "Measurement of Atmospheric Trace Gases by Long Path Differential UV/Visible Absorption Spectroscopy", in Ref. 1.

7. H. Edner, A. Sunesson, S. Svanberg, L. Unéus and S. Wallin, "Differential Optical Absorption Spectroscopy System for Atmospheric Mercury Monitoring", Appl. Opt. **25**, 403 (1986).

8. H. Edner, R. Amer, P. Ragnarsson, M. Rudin and S. Svanberg, "Atmospheric NH_3 Monitoring by Long-path UV Absorption Spectroscopy, International Congress on Optical Science and Engineering: Environment and Pollution Measurement Sensors and Systems, The Hagne, The Netherlands, March 14-15 1990, SPIE Proceedings **1269**, 14 (1990).

9. P. Duffet-Smith, "Practical Astronomy With Your Calculator" third edition, Cambridge University Press 1988.

10. H. Karttunen, P. Kröger, H. Oja, M. Poutanen and K.J. Donner, "Fundamental Astronomy", Springer Verlag Berlin Heidelberg 1987.

Appendixes

1. ASTRO Turbo Pascal unit.

2. Steering system electronics.

3. Steering system cable connections.

4. Steering system drawings.

```
(* Unit: ASTRO                                                    *)
(* Stefan Spnnare 900210                                         *)
(*                                                               *)
(* Methods from:                                                 *)
(* Practical Astronomy With Your Calculator                      *)
(* Third edition                                                 *)
(* By Peter Duffett-Smith                                        *)
(*                                                               *)
(* Mean orbital elements of the planets from:                    *)
(* Fundamental Astronomy                                         *)
(* By H. Karttunen et. al.                                       *)
(*                                                               *)
(* Mean orbital elements  are found in the textfile:             *)
(* ASTRO.DAT (default name)                                      *)
(*                                                               *)
(* Some abbreviations are used in the program:                   *)
(*                                                               *)
(* ut  = universal time = Greenwich civil time                   *)
(* lct = local civil time = local clock time                     *)
(* gst = Greenwich sideral time = Greenwich star time            *)
(* lst = local sideral time = local star time                    *)
(* mst = mean sun time                                           *)
(* rst = real sun time                                           *)
(*                                                               *)
(* ecl = ecliptic coordinates                                    *)
(* equ = equatorial coordinates                                  *)
(* hor = horizon coordinates                                     *)

{$N+}  (* Numeric coprocessor on *)
Unit Astro;
Interface
uses Dos,Crt;

type   str40 = string(.40.);
       planet_type = record
                       name: str40;
                       s: array(.1..5,1..4.) of real;
                       a0:     real;
                     end;

     planet_vector= array(.1..8.) of planet_type;
var  planet:          planet_vector;

function arcsin(x:real):real;
function arccos(x:real):real;
function sgn(x:real):integer;
function rad(deg:real):real;
function deg(rad:real):real;
function in_24(x:real):real;
function in_360(x:real):real;
function in_2pi(x:real):real;
procedure clock(var year,month,day,hour,min,sec,hsec:integer);
function dec_hour(hour,min,sec,hsec:integer):real;
procedure hms(var hour,min,sec:integer;dec_hour:real);
procedure write_hms(dec_hour:real);
procedure writeln_hms(dec_hour:real);

procedure init_astro(directory,file_name:str40);
procedure exit_astro(directory,file_name:str40);

function jd(year,month,day:integer):real;

function lct_ut(lct,zone:real):real;
function ut_lct(ut,zone:real):real;
function gst_lst(gst,longitude:real):real;
function ut_gst(year,month,day:integer;ut:real):real;
function gst_ut(year,month,day:integer;gst:real):real;

procedure auto_lct(var year,month,day:integer;var lct:real);
function auto_lst(longitude,zone:real):real;

procedure precession(var alfa2,delta2:real;year2,month2,day2:integer;
                         alfa1,delta1:real;year1,month1,day1:integer);

function refraction(altitude,air_pressure,air_temperature:real):real;


procedure ecl_equ(var alfa,delta:real;lambda,beta,eps:real);
procedure equ_hor(var altitude,azimuth:real;delta,fi,h:real);
function mst_rst(year,month,day:integer;alfa:real):real;

procedure sun_equ(var alfa,delta:real;year,month,day:integer;ut:real);
procedure moon_equ(var alfa,delta:real;year,month,day:integer;ut:real);
```

```
procedure planet_equ(var alfa,delta:real;year,month,day:integer;ut:real;
                     var planet,earth:planet_type);

Implementation

var  planet_file:      text;
     number_of_planets: integer;

(****** EXTRA FUNCTIONS AND PROCEDURES ******)

function arcsin(x:real):real;
begin
   if x=1.0 then
      arcsin:=pi/2.0
   else
      arcsin:=2.0*arctan(sqrt((1+x)/(1-x)))-pi/2.0;
end; (* arcsin *)

function arccos(x:real):real;
begin
   if x=0.0 then
      arccos:=pi/2.0
   else if x>-1.0 then
      arccos:=2.0*arctan((1+sqrt(1-x*x))/x)+pi/2-pi*x/abs(x)
   else
      arccos:=pi;
end; (* arccos *)

function sgn(x:real):integer;
begin
   if x=0.0 then
      sgn:=0
   else if x>0.0 then
      sgn:=1
   else
      sgn:=-1;
end; (* sgn *)

function rad(deg:real):real;
begin
   rad:=pi*deg/180.0;
end; (* rad *)

function deg(rad:real):real;
begin
   deg:=180.0*rad/pi;
end; (* deg *)

function in_24(x:real):real;  (* 0.0 <= in_24 < 24.0 *)
var x2:real;
begin
   x2:=x/24.0;
   x2:=24.0*(x2-int(x2));
   if x2<0.0 then
      x2:=x2+24.0;
   in_24:=x2;
end; (* in_24 *)

function in_360(x:real):real;   (* 0.0 <= in_360 < 360.0 *)
var x2:real;
begin
   x2:=x/360.0;
   x2:=360.0*(x2-int(x2));
   if x2<0.0 then
      x2:=x2+360.0;
   in_360:=x2;
end; (* in_360 *)

function in_2pi(x:real):real;   (* 0.0 <= in_2pi < 2*pi *)
var x2,pi2:real;
begin
   pi2:=2*pi;
   x2:=x/pi2;
   x2:=pi2*(x2-int(x2));
   if x2<0.0 then
      x2:=x2+pi2;
   in_2pi:=x2;
end; (* in_2pi *)


(* Procedure clock:                                          *)
(* Reads computer time and date.                             *)
(* hsec = 1/100 seconds                                      *)
(* registers from Dos unit                                   *)

procedure clock(var year,month,day,hour,min,sec,hsec:integer);
const datenr=$2a;
      timenr=$2c;
      inter=$21;

 var regs:registers;

begin
   with regs do begin
      ah:=datenr;
      intr(inter,regs);
      year:=cx;
      month:=dh;
      day:=dl;

      ah:=timenr;
      intr(inter,regs);
      hour:=ch;
      min:=cl;
      sec:=dh;
      hsec:=dl;
   end; (* with *)
end; (* clock *)


function dec_hour(hour,min,sec,hsec:integer):real;
begin
   dec_hour:=hour+(min+(sec+hsec/100.0)/60.0)/60.0;
end; (* dec_hour *)

procedure hms(var hour,min,sec:integer;dec_hour:real);
var m:real;
begin
   hour:=trunc(dec_hour);
   m:=60.0*(dec_hour-int(dec_hour));
   min:=trunc(m);
   sec:=trunc(60.0*(m-int(m)));
end; (* hms *)

procedure write_hms(dec_hour:real);
```

```
var h,m,s:integer;
begin
   hms(h,m,s,dec_hour);
   write(h:2,':',m:2,':',s:2);
end; (* write_hms *)

procedure writeln_hms(dec_hour:real);
begin
   write_hms(dec_hour);
   writeln;
end; (* writeln_hms *)

(******  ASTRO PROCEDURES  ******)

(* Procedure read_planet_data:                                    *)
(* Reads mean orbital elements of planets from disk.              *)
(* Filename: ASTRO.DAT (default name)                             *)

procedure read_planet_data(var planet_file:text;
                           directory:str40;
                           file_name:str40;
                           var planet:planet_vector;
                           var number_of_planets:integer);
const disk=0;
var i,j:integer;
    deg,m,s1,s2,s3,s4:real;
    act_dir:str40;
begin
   getdir(disk,act_dir);
   chdir(directory);
   assign(planet_file,file_name);
   reset(planet_file);
   if not eof(planet_file) then begin
      readln(planet_file,number_of_planets);
      for i:=1 to number_of_planets do begin
         with planet(.i.) do begin
            readln(planet_file,name);
            for j:=1 to 4 do begin
               readln(planet_file,deg,m,s1,s2,s3,s4);
               s(.j,1.):=deg+m/60.0+s1/3600.0;
               s(.j,2.):=s2/3600.0;
               s(.j,3.):=s3/3600.0;
               s(.j,4.):=s4/3600.0;
            end; (* for *)
            readln(planet_file,s1,s2,s3,s4);
            s(.5,1.):=s1; s(.5,2.):=s2; s(.5,3.):=s3; s(.5,4.):=s4;
            readln(planet_file,a0);
         end; (* with *)
      end; (* for *)
   end; (* if *)
   close(planet_file);
   chdir(act_dir);
end; (* read_planet_data *)

(* Procedure init_astro:                                          *)
(* Same as read_planet_data but without                           *)
(* unnecessary parameters.                                        *)

procedure init_astro(directory,file_name:str40);
begin
   read_planet_data(planet_file,
                    directory,
                    file_name,
                    planet,
                    number_of_planets);
end; (* init_astro *)

(* Procedure exit_astro:                                          *)
(* Dummy procedure.                                               *)

procedure exit_astro(directory,file_name:str40);
begin
end; (* exit_astro *)

(* Function jd:                                                   *)
(* Calculates Julian day number.                                  *)
(* Correct if year > 1582.                                        *)

function jd(year,month,day:integer):real;
var f,g,a:real;
begin
   if month>=3 then begin
      f:=year;
      g:=month;
   end
   else begin
      f:=year-1;
      g:=month+12;
   end;
   a:=2-int(f/100.0)+int(f/400.0);
   jd:=int(365.25*f)+int(30.6001*(g+1))+day+a+1720994.5;
end; (* jd *)

(* Functions lct_ut,ut_lct and gst_ut:                            *)
(* Converting between different times.                            *)
(* Exampel: lct_ut converts from lct to ut.                       *)
(* All parameters in decimal hours.                               *)

function lct_ut(lct,zone:real):real;
begin
   lct_ut:=in_24(lct-zone);
end; (* lct_ut *)

function ut_lct(ut,zone:real):real;
begin
   ut_lct:=in_24(ut+zone);
end; (* ut_lct *)

function gst_lst(gst,longitude:real):real;
begin
   gst_lst:=in_24(gst+longitude);
end; (* gst_lst *)

(* Function ut_gst and gst_ut:                                    *)
(* Converting between ut and gst.                                 *)
(* Requires year,month,day and decimal ut or gst.                 *)
(* Accuracy better then 1/10 second.                              *)

function ut_gst(year,month,day:integer;ut:real):real;
var jd0,s,t,t0,lct,gst:real;
begin
```

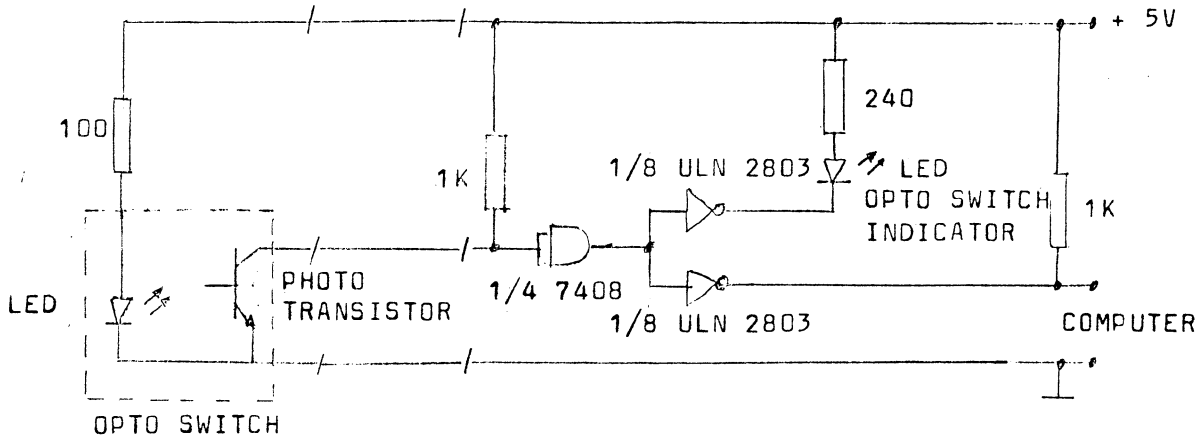
```
   jd0:=jd(year,month,day);
   s:=jd0-2451545.0;
   t:=s/36525.0;
   t0:=6.697374558+2400.051336*t+0.000025862*t*t;
   t0:=in_24(t0);
   gst:=in_24(t0+1.002737909*ut);
   ut_gst:=gst;
end; (* ut_gst *)

function gst_ut(year,month,day:integer;gst:real):real;
var jd0,s,t,t0,lct,ut:real;
begin
   jd0:=jd(year,month,day);
   s:=jd0-2451545.0;
   t:=s/36525.0;
   t0:=6.697374558+2400.051336*t+0.000025862*t*t;
   t0:=in_24(t0);
   ut:=0.9972695663*in_24(gst-t0);
   gst_ut:=ut;
end; (* gst_ut *)


(* Procedure auto_lct:                                          *)
(* Same as clock but time (lct) in decimal hours.               *)

procedure auto_lct(var year,month,day:integer;var lct:real);
var hour,min,sec,hsec:integer;
begin
   clock(year,month,day,hour,min,sec,hsec);
   lct:=dec_hour(hour,min,sec,hsec);
end; (* auto_lct *)


(* Function auto_lst:                                           *)
(* Calculates lst (local sideral time) from computer            *)
(* time and date and local logitude and zone (in hours).        *)

function auto_lst(longitude,zone:real):real;
var year,month,day:integer;
    gst,lst,lct,ut:real;
begin
   auto_lct(year,month,day,lct);
   ut:=lct_ut(lct,zone);
   gst:=ut_gst(year,month,day,ut);
   lst:=gst_lst(gst,longitude);
   auto_lst:=lst;
end; (* auto_lst *)


(* Procedure precession;                                        *)
(* Calculates equatorial coordinates (alfa2 and delta2)         *)
(* from any epoch (date) (year >1582) to another.               *)
(* This is due to the gyrating motion the Earth's axis.         *)
(* The method is correct for long periods                       *)
(* (hundreds of years).                                         *)

procedure precession(var alfa2,delta2:real;year2,month2,day2:integer;
                         alfa1,delta1:real;year1,month1,day1:integer);
type matrix33= array(.1..3,1..3.) of real;
     matrix3=  array(.1..3.) of real;
var  p1,p2:matrix33;
     v,w,s:matrix3;
     zaeta,z,taeta,jd1,jd2,cx,sx,cz,sz,ct,st:real;

   procedure matrix_mult(var m3_out,m3_in:matrix3; var m33_in:matrix33);
   const n=3;
   var i,j:integer;
       sum:real;
   begin
       for i:=1 to n do begin
           sum:=0;
           for j:=1 to n do begin
              sum:=sum+m33_in(.i,j.)*m3_in(.j.);
           end;
           m3_out(.i.):=sum;
       end;
   end; (* matrix_mult *)

   procedure param(var zaeta,z,taeta:real;jd:real);
   var t,t2,t3:real;
   begin
      t:=(jd-2451545.0)/36525.0;
      t2:=t*t; t3:=t*t*t;
      zaeta:=0.6406161*t+0.0000839*t2+0.0000050*t3;
      z:=    0.6406161*t+0.0003041*t2+0.0000051*t3;
      taeta:=0.5567530*t-0.0001185*t2-0.0000116*t3;
   end; (* param *)

   procedure sin_cos(var sx,cx,sz,cz,st,ct,zaeta,z,taeta:real);
   begin
      sx:=sin(rad(zaeta)); cx:=cos(rad(zaeta));
      sz:=sin(rad(z));     cz:=cos(rad(z));
      st:=sin(rad(taeta)); ct:=cos(rad(taeta));
   end; (* sin_cos *)

begin

   jd1:=jd(year1,month1,day1);
   param(zaeta,z,taeta,jd1);
   sin_cos(sx,cx,sz,cz,st,ct,zaeta,z,taeta);

   p1(.1,1.):=cx*ct*cz-sx*sz; p1(.1,2.):=cx*ct*sz+sx*cz; p1(.1,3.):=cx*st;
   p1(.2,1.):=-sx*ct*cz-cx*sz; p1(.2,2.):=-sx*ct*sz+cx*cz; p1(.2,3.):=-sx*st;
   p1(.3,1.):=-st*cz;          p1(.3,2.):=-st*sz;          p1(.3,3.):=ct;

   v(.1.):=cos(alfa1)*cos(delta1);
   v(.2.):=sin(alfa1)*cos(delta1);
   v(.3.):=sin(delta1);

   matrix_mult(s,v,p1);

   jd2:=jd(year2,month2,day2);
   param(zaeta,z,taeta,jd2);
   sin_cos(sx,cx,sz,cz,st,ct,zaeta,z,taeta);

   p2(.1,1.):=cx*ct*cz-sx*sz; p2(.1,2.):=-sx*ct*cz-cx*sz; p2(.1,3.):=-st*cz;
   p2(.2,1.):=cx*ct*sz+sx*cz; p2(.2,2.):=-sx*ct*sz+cx*cz; p2(.2,3.):=-st*sz;
   p2(.3,1.):=cx*st;          p2(.3,2.):=-sx*st;          p2(.3,3.):=ct;

   matrix_mult(w,s,p2);

   alfa2:= arctan(w(.2.)/w(.1.));
   delta2:=arcsin(w(.3.));

   while (alfa1-alfa2)>pi/2.0 do
       alfa2:=alfa2+pi;
   while (alfa2-alfa1)>pi/2.0 do
       alfa2:=alfa2-pi;
```

```
end; (* precession *)

(* Function refraction:                                          *)
(* Calculates the atmospheric refraction from                    *)
(* altitude, air pressure (mbar) and temperature (centigrade).   *)
(* You find the apparent altitude a' from the true               *)
(* altitude a by: a'= a + refraction.                            *)

function refraction(altitude,air_pressure,air_temperature:real):real;
var r,a:real;
begin
   a:=deg(altitude);
   if a > 15.0 then
      r:=0.00452*air_pressure*(sin(pi/2-altitude)/cos(pi/2-altitude))
         /(273.0+air_temperature)
   else
      r:=air_pressure*(0.1594+0.0196*a+0.00002*a*a)/
         ((273.0+air_temperature)*(1+0.505*a+0.0845*a*a));
   refraction:=rad(r);
end; (* refraction *)


(* Procedures ecl_equ:                                           *)
(* Coordinate transformation from ecliptic coordinates           *)
(* lambda,beta and eps to equatorial coordinates                 *)
(* alfa and delta.                                               *)
(* (eps see procedure eps_earth)                                 *)

procedure ecl_equ(var alfa,delta:real;lambda,beta,eps:real);
var sl,cl,sb,cb,se,ce,sd,cd,sa,ca,a:real;
begin
   se:=sin(eps); ce:=cos(eps);
   sb:=sin(beta); cb:=cos(beta);
   sl:=sin(lambda); cl:=cos(lambda);
   sd:=sb*ce+cb*se*sl;
   delta:=arcsin(sd);
   a:=arctan(sl/cl*ce-sb/cb*se/cl);
   while (lambda-a)>pi/2.0 do
      a:=a+pi;
   while (a-lambda)>pi/2.0 do
      a:=a-pi;
   alfa:=a;
end; (* ecl_equ *)

(* Procedure equ_hor:                                            *)
(* Coordinate transformation from equatorial coordinates         *)
(* delta, fi (local latitude) and h (hour angel) to              *)
(* horizon coordinates altitude and azimuth.                     *)

procedure equ_hor(var altitude,azimuth:real;delta,fi,h:real);
var sd,sf,cf,sa,az:real;
begin
   sd:=sin(delta);
   sf:=sin(fi); cf:=cos(fi);
   sa:=sd*sf+cos(delta)*cf*cos(h);
   altitude:=arcsin(sa);
   az:=arccos((sd-sf*sa)/(cf*cos(altitude)));
   if sin(h)>0.0 then
      azimuth:=2.0*pi-az
   else
      azimuth:=az;
end; (* equ_hor *)


(* Function mst_rst:                                             *)
(* Calculates the difference between mean sun time and           *)
(* real sun time from date and sun coordinate alfa               *)
(* at the same date.                                             *)

function mst_rst(year,month,day:integer;alfa:real):real;
var ut:real;
begin
   ut:=gst_ut(year,month,day,alfa);
   mst_rst:=12.0-ut;
end; (* mst_rst *)


(* Function kepler_solve:                                        *)
(* Help function to sun_equ and planet_equ.                      *)
(* Calculates a solution E (Newton Rhapson) to                   *)
(* Keplers equation: E-e*sin(E) = M.                             *)
(* e = eccentricity of orbit < 0.1                               *)
(* M = mean anomaly of orbit  (m)                                *)
(* E = eccentric anomaly of orbit  (en)                          *)

function kepler_solve(e,m:real):real;
const eps=1.0e-15;
var en,en2:real;
begin
   en:=m;
   en2:=en+1.0;
   while abs(en-en2)>eps do begin
      en2:=en;
      en:=en-(en-e*sin(en)-m)/(1-e*cos(en));
   end;
   kepler_solve:=en;
end; (* kepler_solve *)


(* Function eps_earth:                                           *)
(* Help function to sun_equ, moon_equ and planet_equ.            *)
(* Calculates the mean obliquity of the ecliptic,                *)
(* ie. the angel between Earth's axis and a normal               *)
(* to Earth's orbit plane.                                       *)
(* Long time accuracy for hundreds of years.                     *)

function eps_earth(year,month,day:integer;ut:real):real;
var jd0,t:real;
begin
   jd0:=jd(year,month,day)+ut/24.0;
   t:=(jd0-2451545.0)/36525.0;
   eps_earth:=rad(23.43929167+(-46.815*t-0.0006*t*t+0.00181*t*t*t)/3600.0);
end; (* eps_earth *)


(* Procedure sun_ecl:                                            *)
(* Help procedure to sun_equ and moon_equ.                       *)
(* Calculates ecliptic coordinates lambda,beta and               *)
(* mean anomaly m of sun.                                        *)
(* Long time accuracy to 1/10 arc minute.                        *)

procedure sun_ecl(var lambda,beta,m:real;year,month,day:integer;ut:real);
var epsg,wg,e,en,v,l,jd0,t:real;
begin
   jd0:=jd(year,month,day)+ut/24.0;
   t:=(jd0-2415020.0)/36525.0;
   epsg:=rad(in_360(279.6966778+36000.76892*t+0.0003025*t*t));
```

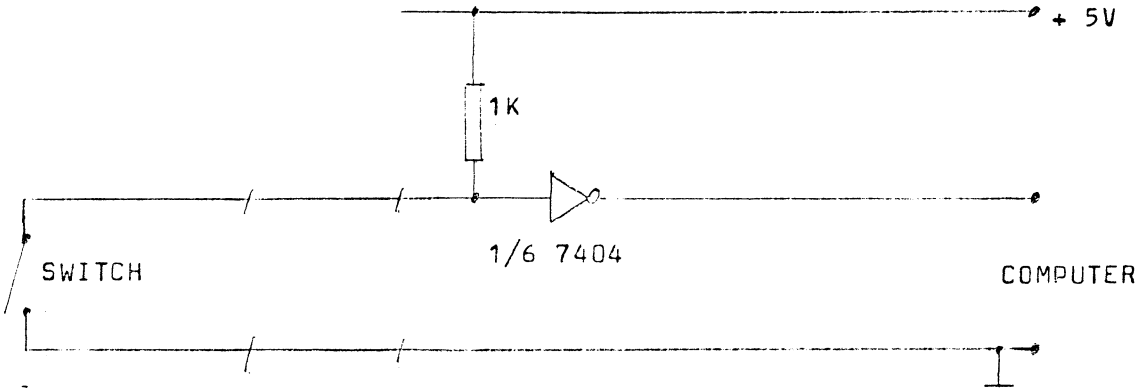
```
   wg:=  rad(in_360(281.2208444+1.719175*t+0.000452778*t*t));
   e:=   0.01675104-0.0000418*t-0.000000126*t*t;
   m:=in_2pi(epsg-wg);
   en:=kepler_solve(e,m);
   v:=2.0*arctan(sqrt((1+e)/(1-e))*sin(en/2.0)/cos(en/2.0));
   l:=in_2pi(v+wg);
   lambda:=l;
   beta:=0.0;
end; (* sun_ecl *)


(* Procedure sun_equ:                                          *)
(* Calculates equatorial coordinates alfa and delta of sun     *)
(* from date and universal time.                               *)
(* Long time accuracy to 1/10 arc minute.                      *)

procedure sun_equ(var alfa,delta:real;year,month,day:integer;ut:real);
var lambda,beta,m,epse:real;
begin
   sun_ecl(lambda,beta,m,year,month,day,ut);
   epse:=eps_earth(year,month,day,ut);
   ecl_equ(alfa,delta,lambda,beta,epse);
end; (* sun_equ *)


(* Procedure moon_equ;                                         *)
(* Calculates equatorial coordinates alfa and delta of moon    *)
(* from date and universal time.                               *)
(* Accuracy to 1/5 degree for tenths of years from 1990.       *)
(* No compensation for parallax error.                         *)

procedure moon_equ(var alfa,delta:real;year,month,day:integer;ut:real);
const l0=318.351648;
      p0=36.340410;
      n0=318.510107;
       i=5.145396;
       e=0.054900;
       thaeta0=0.5181;
       pi0=0.9507;
var l,mm,n,ev,ae,a3,a4,sms,mmprim,ec,lprim,lbis,v,nprim,
    d,jd0,lambda,beta,lambda_sun,beta_sun,m_sun,tc,ln,epse:real;
begin
   sun_ecl(lambda_sun,beta_sun,m_sun,year,month,day,ut);
   jd0:=jd(year,month,day);
   d:=jd0-2447891.5+ut/24.0;
   l:=in_360(13.176396*d+l0);
   mm:=in_360(l-0.1114041*d-p0);
   n:=in_360(n0-0.052953*d);
   ev:=1.2739*sin(2.0*(rad(l)-lambda_sun)-rad(mm));
   sms:=sin(m_sun);
   ae:=0.1858*sms;
   a3:=0.37*sms;
   mmprim:=mm+ev-ae-a3;
   ec:=6.2886*sin(rad(mmprim));
   a4:=0.214*sin(2.0*rad(mmprim));
   lprim:=l+ev+ec-ae+a4;
   v:=0.6583*sin(2.0*(rad(lprim)-lambda_sun));
   lbis:=v+lprim;
   nprim:=n-0.16*sms;
   ln:=rad(lbis-nprim);
   tc:=arctan(sin(ln)/cos(ln)*cos(rad(i)));
   while (tc-ln)>pi/2.0 do
      tc:=tc-pi;
   while (ln-tc)>pi/2.0 do
      tc:=tc+pi;
   lambda:=tc+rad(nprim);
   beta:=arcsin(sin(ln)*sin(rad(i)));
   epse:=eps_earth(year,month,day,ut);
   ecl_equ(alfa,delta,lambda,beta,epse);
end; (* moon_equ *)


(* Procedure planet_parameters:                                *)
(* Help procedure to planet_equ.                               *)
(* Calculates mean orbital elements of a planet                *)
(* at julian date jd.                                          *)

procedure planet_parameters(var l,w,ohm,i,e,a:real;
                            var planet:planet_type;jd:real);
var t,t2,t3:real;
    j:integer;
begin
   t:=(jd-2415020.0)/36525.0;
   t2:=t*t; t3:=t2*t;
   with planet do begin
      l:=  rad(in_360(s(.1,1.)+s(.1,2.)*t+s(.1,3.)*t2+s(.1,4.)*t3));
      w:=  rad(in_360(s(.2,1.)+s(.2,2.)*t+s(.2,3.)*t2+s(.2,4.)*t3));
      ohm:=rad(in_360(s(.3,1.)+s(.3,2.)*t+s(.3,3.)*t2+s(.3,4.)*t3));
      i:=  rad(in_360(s(.4,1.)+s(.4,2.)*t+s(.4,3.)*t2+s(.4,4.)*t3));
      e:=  s(.5,1.)+s(.5,2.)*t+s(.5,3.)*t2+s(.5,4.)*t3;
      a:=a0;
   end; (* with *)
end; (* planet_parameters *)


(* Procedure planet_equ;                                       *)
(* Calculates equatorial coordinates alfa and delta            *)
(* of a planet from date and universal time.                   *)
(* Long time accuracy to 1/5 degree.                           *)
(* The procedure requires planet and earth parameters.         *)
(* planet: planet(.1.),  1 <= i <=8,  i <> 3.                  *)
(* earth:  planet(.3.), usually but earth-planet can           *)
(* be arbitarly choosen.                                       *)

procedure planet_equ(var alfa,delta:real;year,month,day:integer;ut:real;
                     var planet,earth:planet_type);
var lp,wp,ohmp,ep,ip,ap,le,we,ohme,ee,ie,ae,jd0:real;
    lambda,beta,mp,enp,vp,rp,psip,lprimp,lpr2,lpo,rprimp,
    me,ve,ene,re,epse,lple:real;
begin
      jd0:=jd(year,month,day)+ut/24.0;
      planet_parameters(lp,wp,ohmp,ip,ep,ap,planet,jd0);
      mp:=in_2pi(lp-wp);
      enp:=kepler_solve(ep,mp);
      vp:=2.0*arctan(sqrt((1+ep)/(1-ep))*sin(enp/2.0)/cos(enp/2.0));
      rp:=ap*(1-ep*ep)/(1+ep*cos(vp));
      lpo:=in_2pi(lp-ohmp+vp-mp);
      psip:=arcsin(sin(lpo)*sin(ip));
      lpr2:=arctan(sin(lpo)/cos(lpo)*cos(ip));
      while (lpr2-lpo)>pi/2.0 do
         lpr2:=lpr2-pi;
      while (lpo-lpr2)>pi/2.0 do
         lpr2:=lpr2+pi;
      lprimp:=lpr2+ohmp;
      rprimp:=rp*cos(psip);

      planet_parameters(le,we,ohme,ie,ee,ae,earth,jd0);
      me:=in_2pi(le-we);
```

```
      ene:=kepler_solve(ee,me);
      ve:=2.0*arctan(sqrt((1+ee)/(1-ee))*sin(ene/2.0)/cos(ene/2.0));
      re:=ae*(1-ee*ee)/(1+ee*cos(ve));
      lple:=lprimp-le;

      if ap>ae then
         lambda:=arctan(re*sin(lple)/(rprimp-re*cos(lple)))+lprimp
      else
         lambda:=arctan(rprimp*sin(-lple)/(re-rprimp*cos(-lple)))+le+pi;

      lambda:=in_2pi(lambda);
      beta:=arctan((rprimp*sin(psip)/cos(psip)*sin(lambda-lprimp))/
                   (re*sin(lprimp-le)));
      epse:=eps_earth(year,month,day,ut);
      ecl_equ(atfa,delta,lambda,beta,epse);
end; (* planet_equ *)

end. (* Astro *)
```

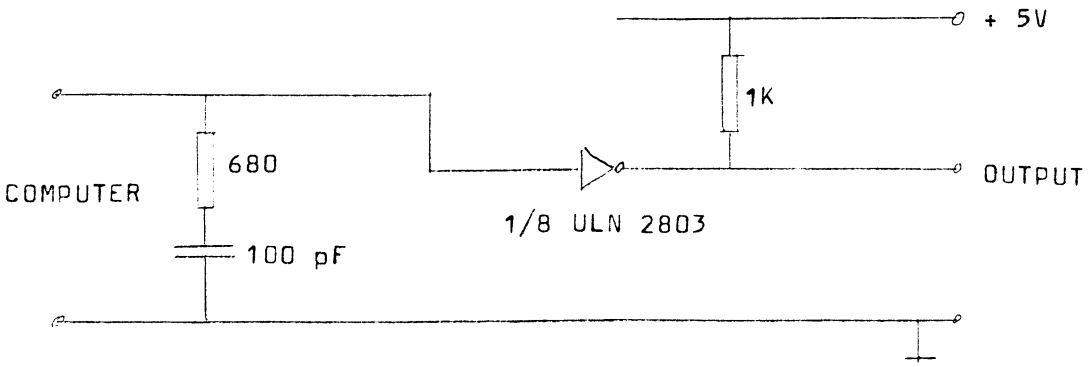
1. OPTO SWITCH CONNECTION



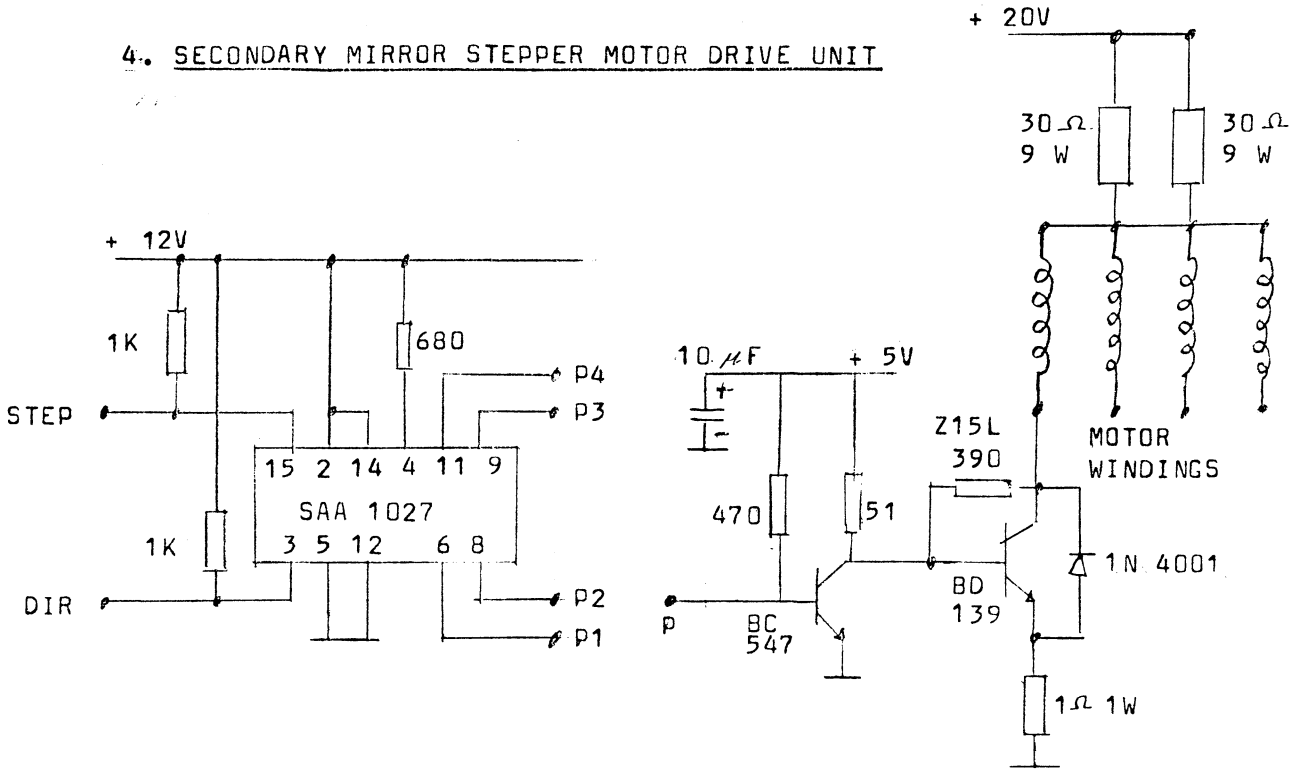
2. JOY STICK SWITCH CONNECTION



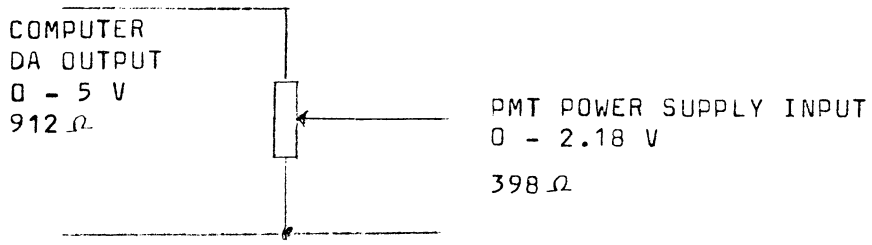
3. COMPUTER OUTPUT



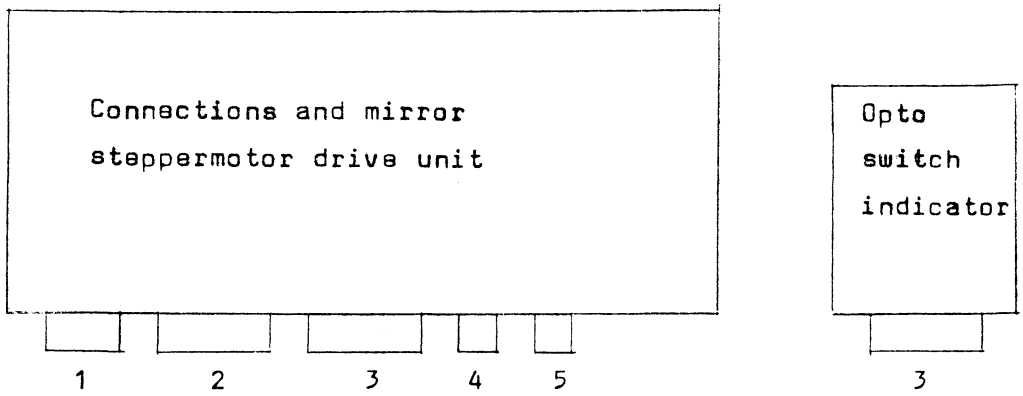
4. SECONDARY MIRROR STEPPER MOTOR DRIVE UNIT



5. PMT POWER SUPPLY COMPUTER CONNECTION



Connections



1 Telescope steppermotors

2 Mirror steppermotors

3 Computer (opto switch indicator)

4 Spectrometer wavelength and prism steppermotor

5 Steering switches

1 Telescope stepper motors

1	GND
2	GND
3	GND
4	
5	
6	+ 5 V
7	+ 5 V
8	+ 5 V
9	Direction horizontal motor
10	Step "
11	Direction vertical motor
12	Step "
13	
14	Opto switch horizontal motor
15	Opto switch vertical motor

2 Mirror stepper motors

1	GND	
2	GND	
3	GND	
4	Opto switch motor 1	
5	Opto switch motor 2	
6		
7		
8		
9		
10	+ 5 V	
11	+ 5 V	
12	+ 5 V	
14	+ 20 V, 15 ohm	
15	+ 20 V, 15 ohm	
16	Phase 1	Motor 1
17	" 2	
18	" 3	
19	" 4	
20	+ 20 V, 15 ohm	
21	+ 20 V, 15 ohm	
22	Phase 1	Motor 2
23	" 2	
24	" 3	
25	" 4	

3 Computer

1	GND
2	GND
3	GND
4	Steering switch 1
5	" 2
6	" 3
7	" 4
8	Direction spectrometer wavelength motor
9	Step "
10	(No wire)
11	+ 5 V
12	+ 5 V
13	+ 5 V
14	Direction horizontal motor
15	Step "
16	Direction vertical motor
17	Step "
18	Direction mirror motor 1
19	Step "
20	Direction mirror motor 2
21	Step "
22	Opto switch horizontal motor
23	Opto switch vertical motor
24	Opto switch mirror motor 1
25	Opto switch mirror motor 2

(Spectometer wavelength motor has no opto switch connection)

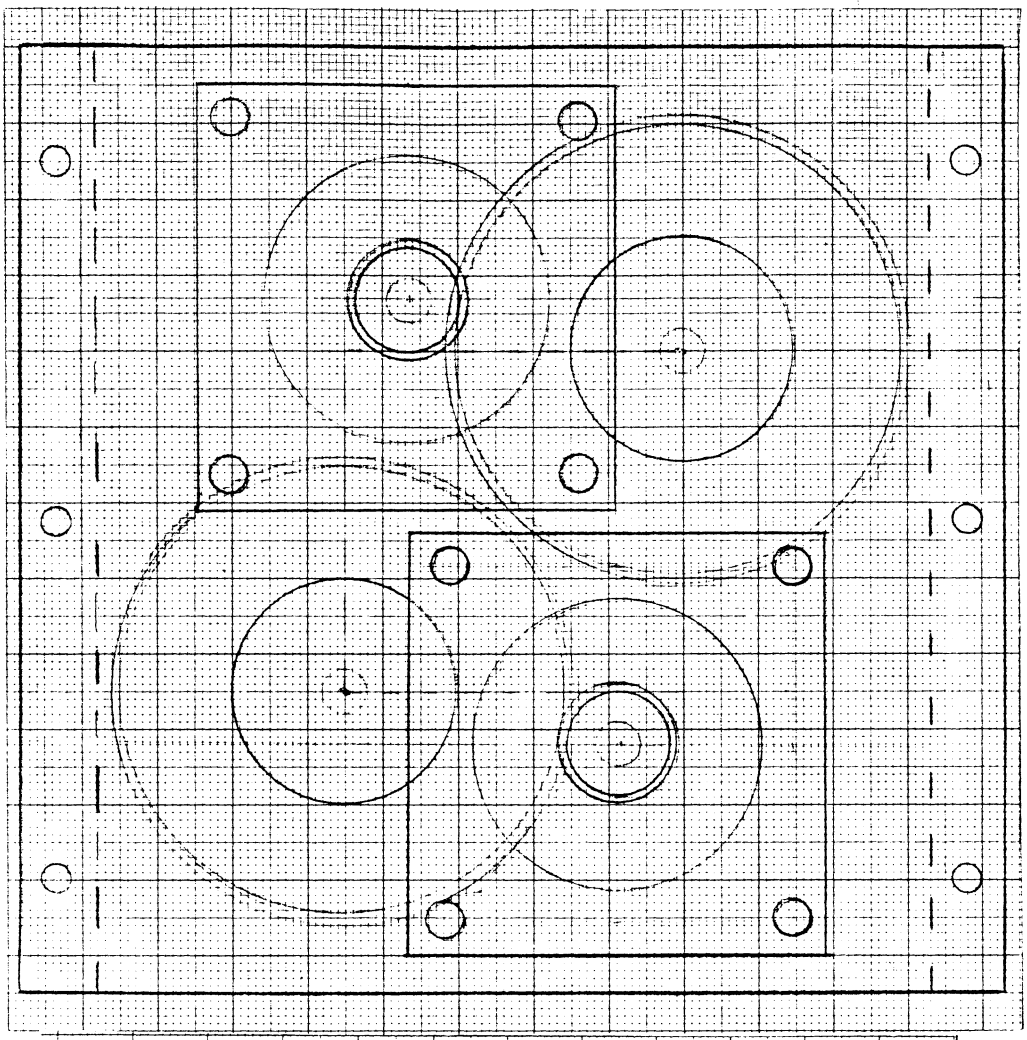
4 Spectrometer wavelength and prism stepper motors

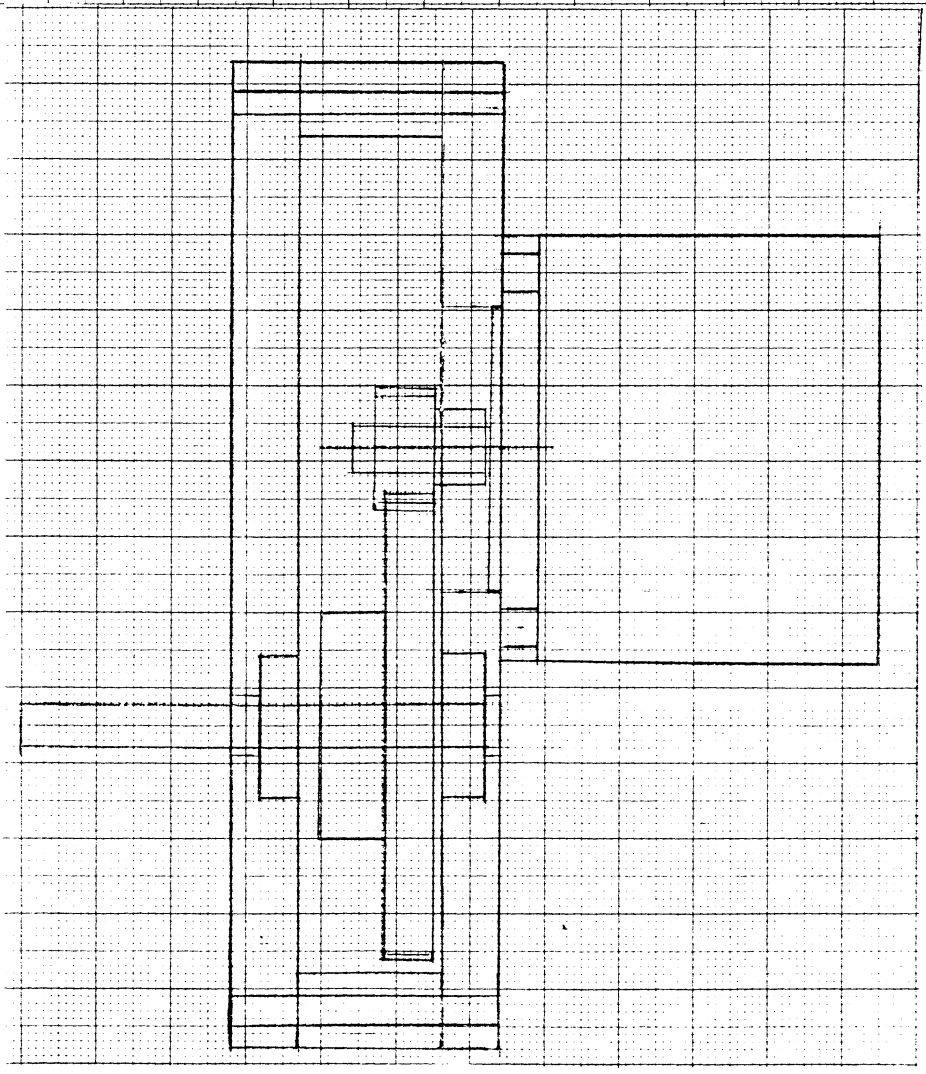
1	GND
2	Direction spectrometer wavelength motor
3	Step "
4	
5	+ 12 V, 1 A
6	Phase 1
7	" 2 Prism motor
8	" 3
9	" 4

5 Steering switches

1	GND
2	
3	
4	
5	
6	Switch 1
7	" 2
8	" 3
9	" 4

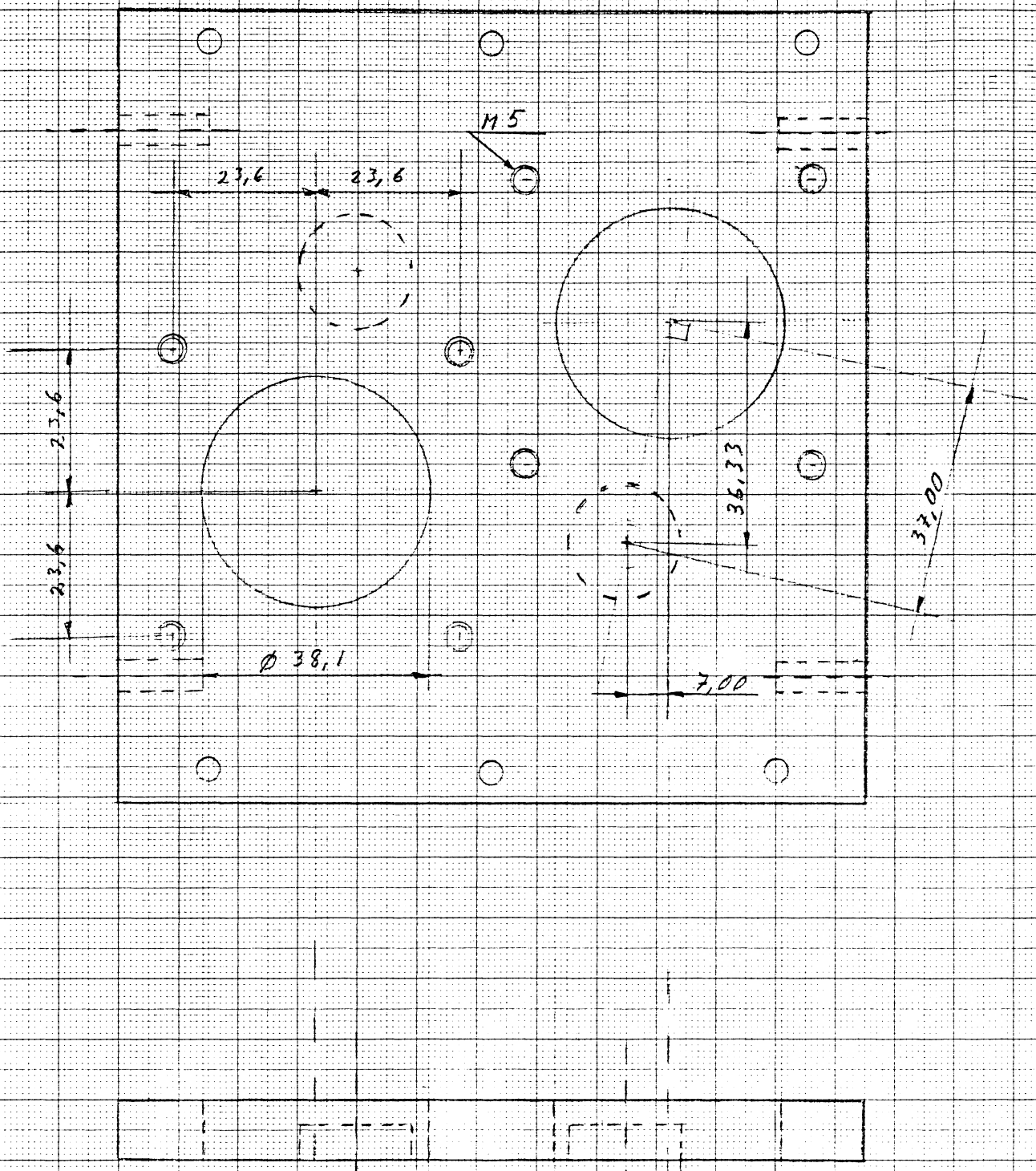
GEAR BOX OF SECONDARY MIRROR



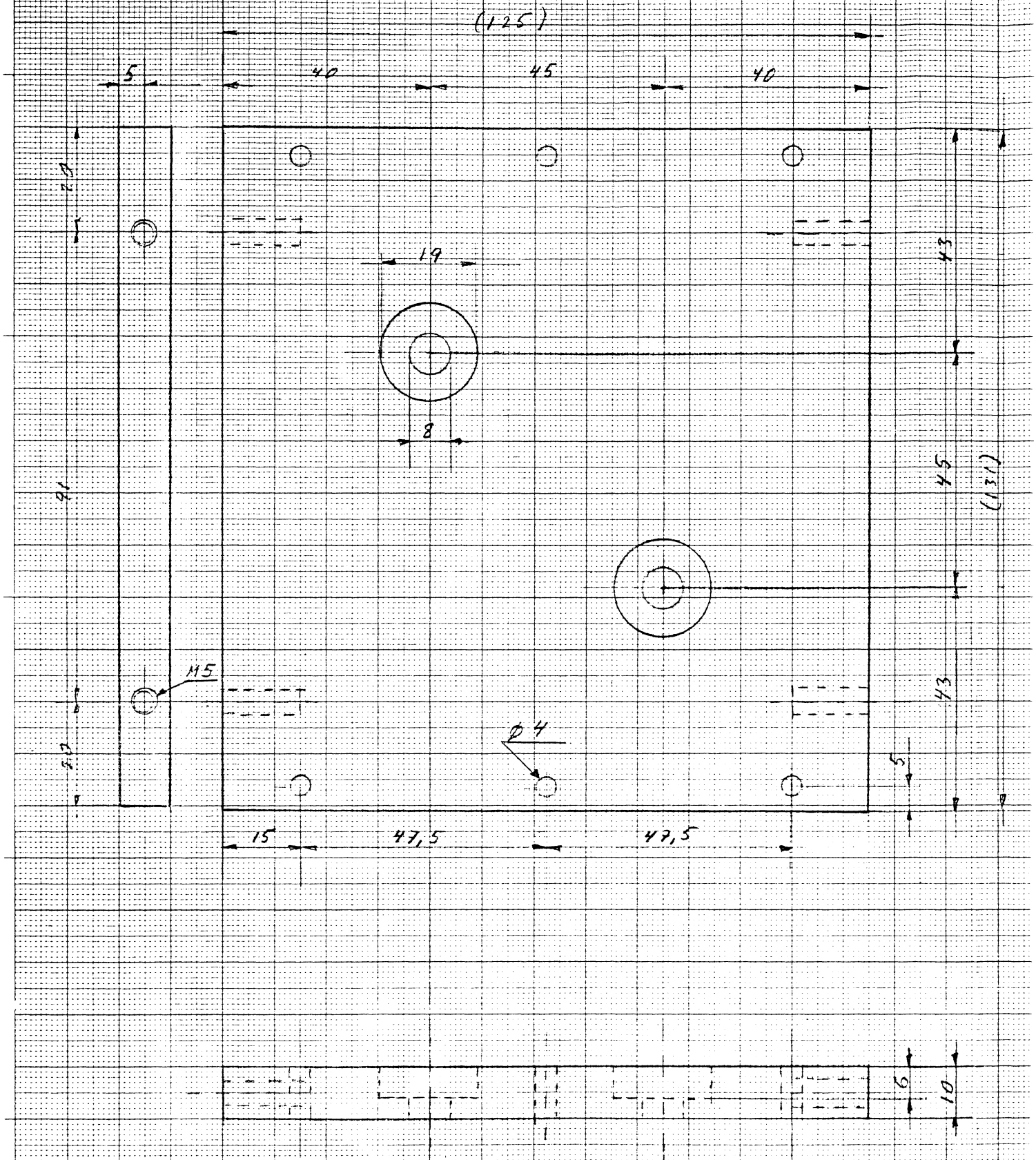


GEAR BOX OF SECONDARY MIRROR

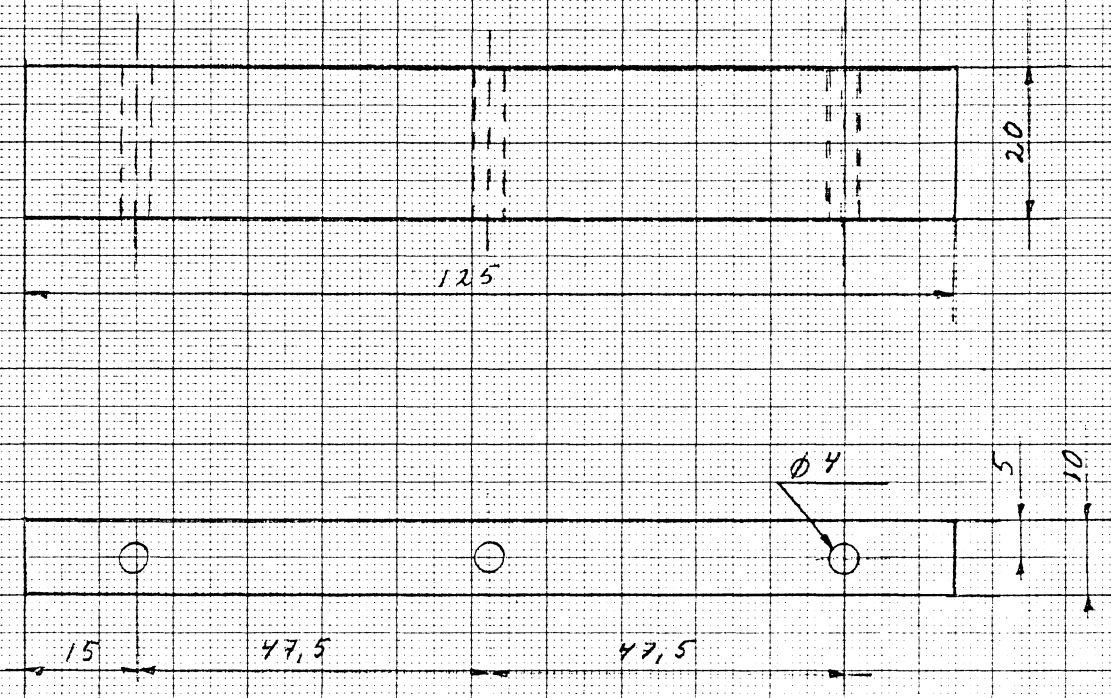
Appendix 4



GEAR BOX OF SECONDARY MIRROR

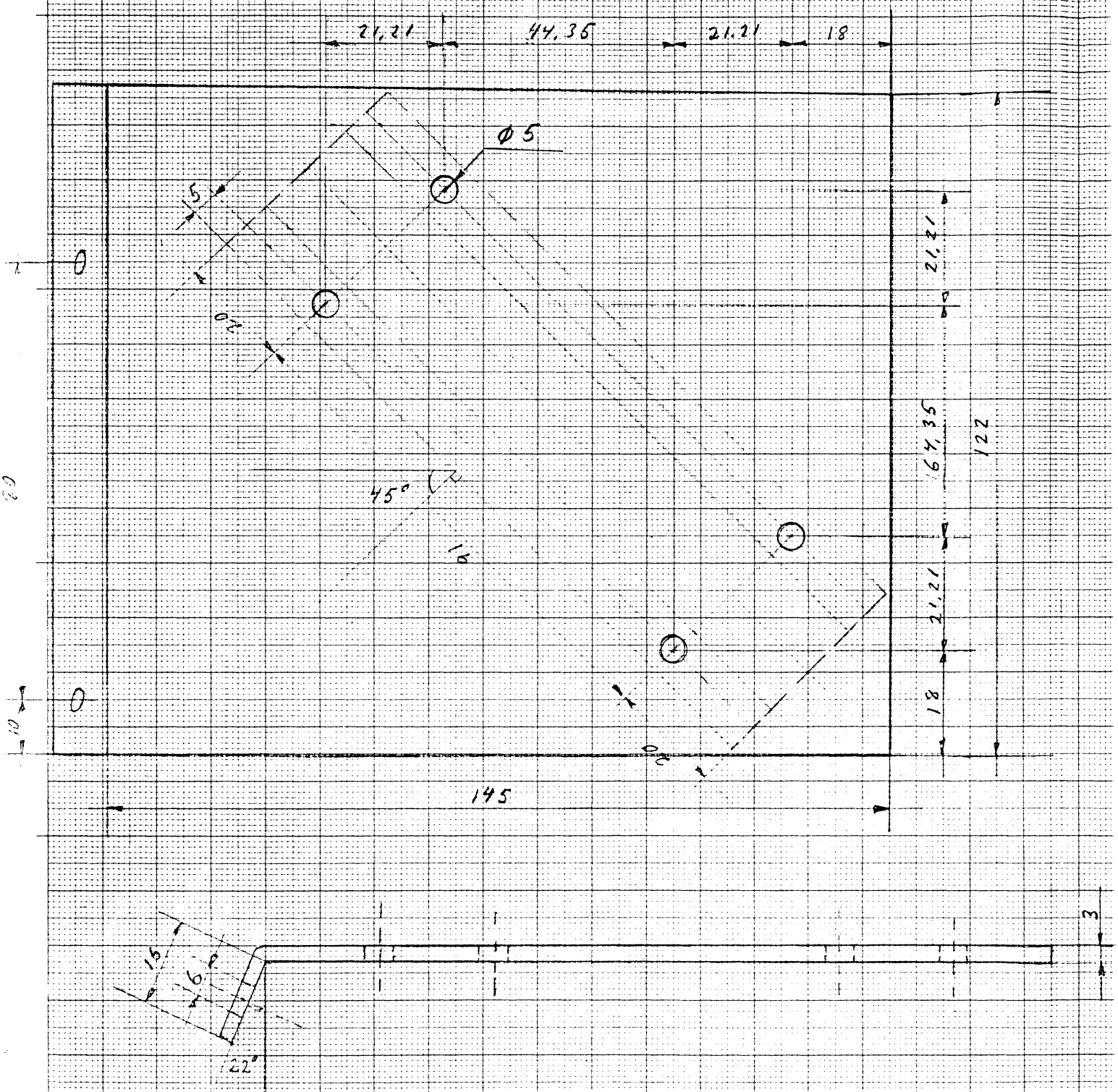


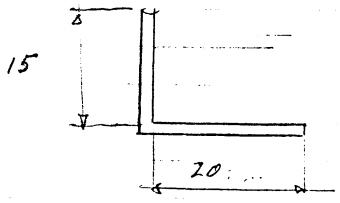
GEAR BOX OF SECONDARY MIRROR

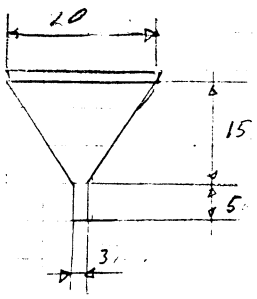


GEAR BOX OF SECONDARY MIRROR

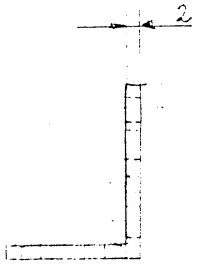
Appendix 4

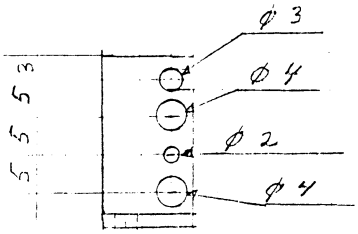




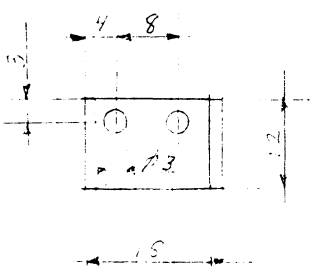


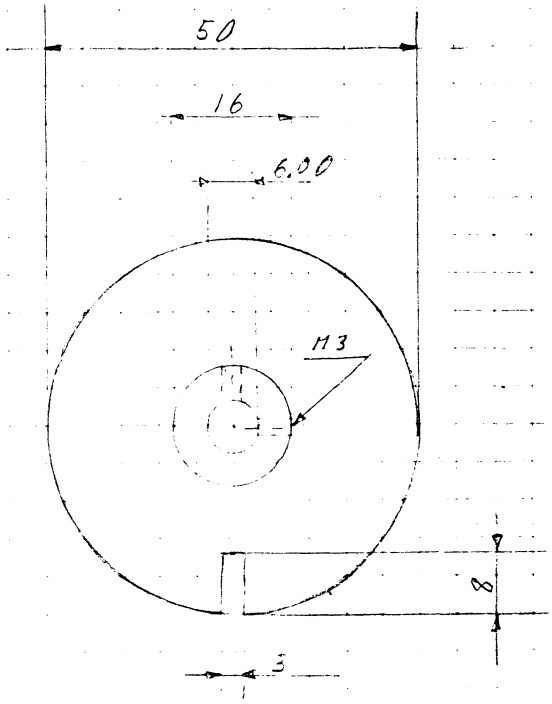
OPTO SWITCH LIMITATION FOR HORIZONTAL STEPPER MOTOR



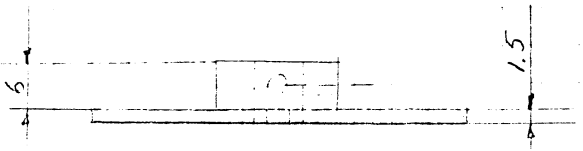


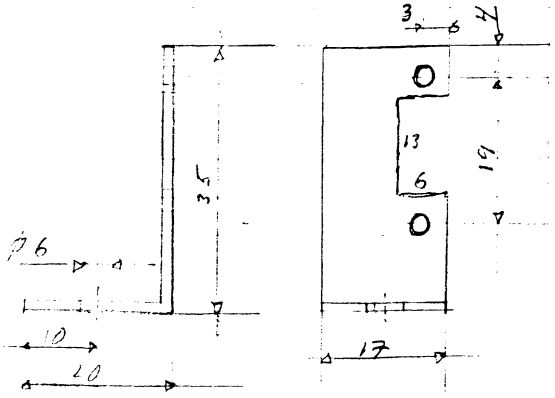
OPTO SWITCH HOLDER



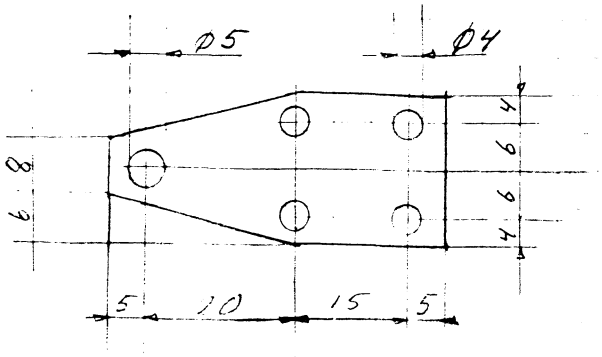


OPTO SWITCH LIMITATION FOR SECONDARY MIRROR STEPPER MOTORS

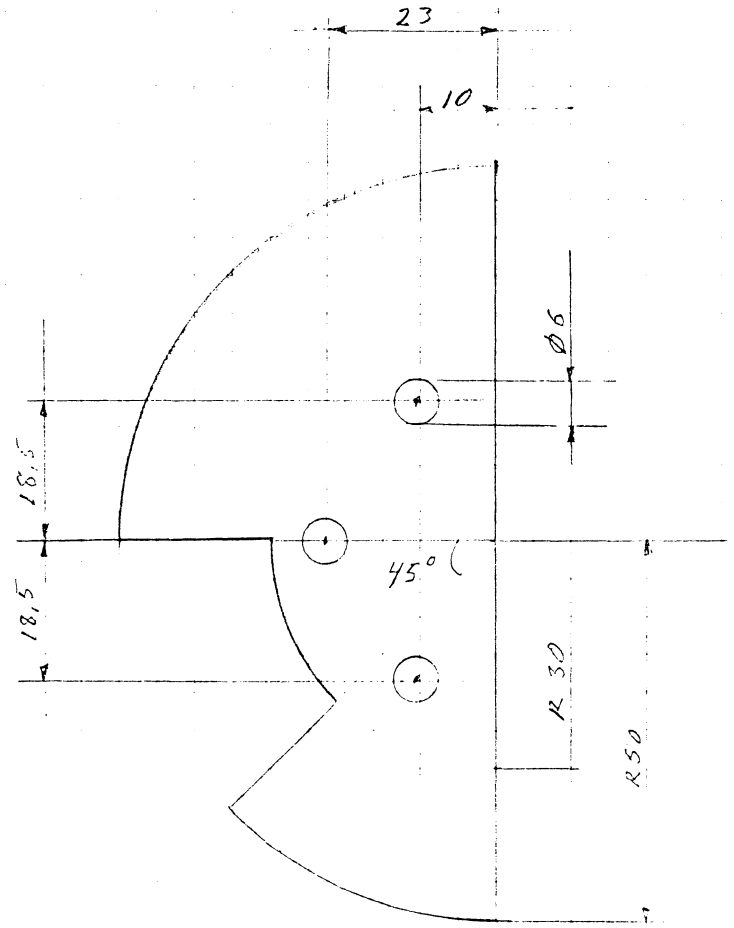




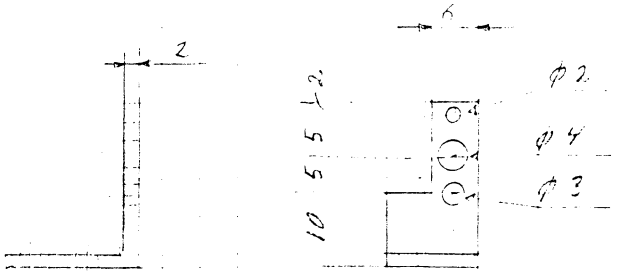
OPTO SWITCH HOLDER

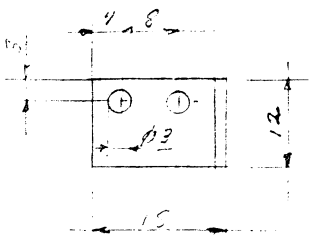


CABEL HOLDER



OPTO SWITCH LIMITATION FOR
VERTICAL STEPPER MOTOR





OPTO SWITCH HOLDER