AUTOMATION OF THE ACQUISITION SYSTEM OF THE 1,9m TELESCOPE FOR THE CHARGE COUPLED DEVICE (CCD) CAMERA

By D.B. Carter

Submitted in part fulfilment of the requirements laid down for the
Master Diploma
in the School of Electrical Engineering
at the Cape Technikon

November 1988

Electronics Section
South African Astronomical Observatory
Council for Scientific and Industrial Research

Abstract

This thesis describes the control system developed to improve the efficiency of star acquisition on a ground–based optical telescope. "Star Acquisition" refers to the process of identifying the star of interest in a field of stars and centering it on the optical axis of the telescope , as well as setting an autoguider detector on a suitable star so the autoguider can improve the tracking performance of the telescope. Efficiency is improved by making all functions remote controlled , so the astronomer does not have to move between the control room and the telescope to operate the instrument.

CONTENTS

AUTOMATION OF THE ACQUISITION SYSTEM OF THE 1,9m TELESCOPE FOR THE CHARGE COUPLED DEVICE (CCD) CAMERA

By D.B. Carter

Submitted in part fulfilment of the requirements laid down for the
Master Diploma
in the School of Electrical Engineering
at the Cape Technikon

November 1988

Electronics Section
South African Astronomical Observatory
Council for Scientific and Industrial Research

Abstract

This thesis describes the control system developed to improve the efficiency of star acquisition on a ground–based optical telescope. "Star Acquisition" refers to the process of identifying the star of interest in a field of stars and centering it on the optical axis of the telescope , as well as setting an autoguider detector on a suitable star so the autoguider can improve the tracking performance of the telescope. Efficiency is improved by making all functions remote controlled , so the astronomer does not have to move between the control room and the telescope to operate the instrument.

CONTENTS

1. Objective.

The objective of the project which is described is to improve the efficiency of an observer in the process of star acquisition , which can be a laborious , time–consuming task for an observing astronomer. It involves moving the telescope to point at the coordinates of the required star , looking through the telescope eyepiece to identify the required star in the field of stars seen (in a crowded field , this can be quite difficult) , and adjusting the telescope position to centre this star on the telescope optical axis. On a large telescope this process can involve a considerable amount of moving around between the control room , the observing floor , and the telescope eyepiece – which involves climbing up and down ladders. If a closed circuit Television system is used instead of an eyepiece , and the telescope acquisition system automated and remotely controllable , then a considerable amount of time can be saved by the observer being able to control most of the process from one place. A detailed explaination of the star acquisition process can be found in section 2.3 pg 4.

This is the background to the proposal for the project to automate the acquisition system of the 1,9m telescope. The Low light level closed circuit television system required has been in use with other instruments on this telescope for a number of years.

A further reason for the project was in connection with the 1986 apparition of comet Halley. To photograph a comet , which is moving relative to the stars , with a telescope which is designed to track the movement of stars across the sky , requires a system that can compensate for the relative movement between the stars and the comet and for this system to control the telescope drive system appropriately. The remote controlled , automated X–Y motion table (called the XYslide) which forms part of the automated acquisition system , can , in conjunction with an existing instrument called an autoguider , perform this task.

2. INTRODUCTION.

The South African Astronomical Observatory undertakes Ground–based Optical Astronomical research at it's observing station near Sutherland in the Karoo. The Electronics group forms part of the technical support group necessary for the operation of the telescopes. The electronics group undertakes developement of improvements to the telescope control systems and detector instrumentation , by modifying existing equipment and building new systems.

2.1. The 1,9m Telescope.

The telescope is a 1,9m f18 reflector instrument mounted on a polar axis system (ie one axis is parallel to the axis of rotation of the earth). The dimension 1,9m refers to the diameter of the main mirror in the optical system. The optics of the telescope are illustrated in Fig. 1.

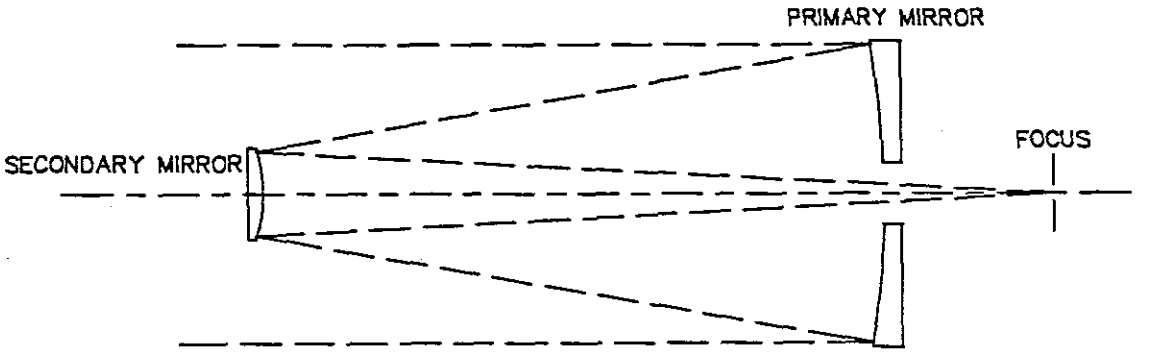


Fig. 1.

A computerized encoding and display system gives the telescope's pointing position. A computerized slow motion drive system controls the slow movement of the telescope to track the path of a star across the sky. The turret rotation is computer controlled so that the slit (opening in the turret) automatically follows the telescope movement. Data acquisition instruments are bolted onto the telescope and the data are stored in a minicomputer system. The instruments are all controlled by the computer to some degree.

2.2. Telescope Instrumentation and Acquisition System.

All Instruments bolted onto the telescope contain a system of lenses and mirrors referred to as the acquisition system. This enables the astronomer to see the stars in the telescope field of view , identify the program star and align it on the telescope's optical axis. If required , he also identifies a nearby star for use as a guide star , and sets the autoguider probe on it.

An autoguider is an instrument which detects and compensates for small errors in the telescope's ability to accurately track the motion of a star across the sky. These errors are introduced by mechanical factors such as flexure of the telescope structure and machining tolerances of the drive gearing , as well as refraction effects of the earth's atmosphere.

The acquisition system used with the CCD camera is illustrated in Fig 2.

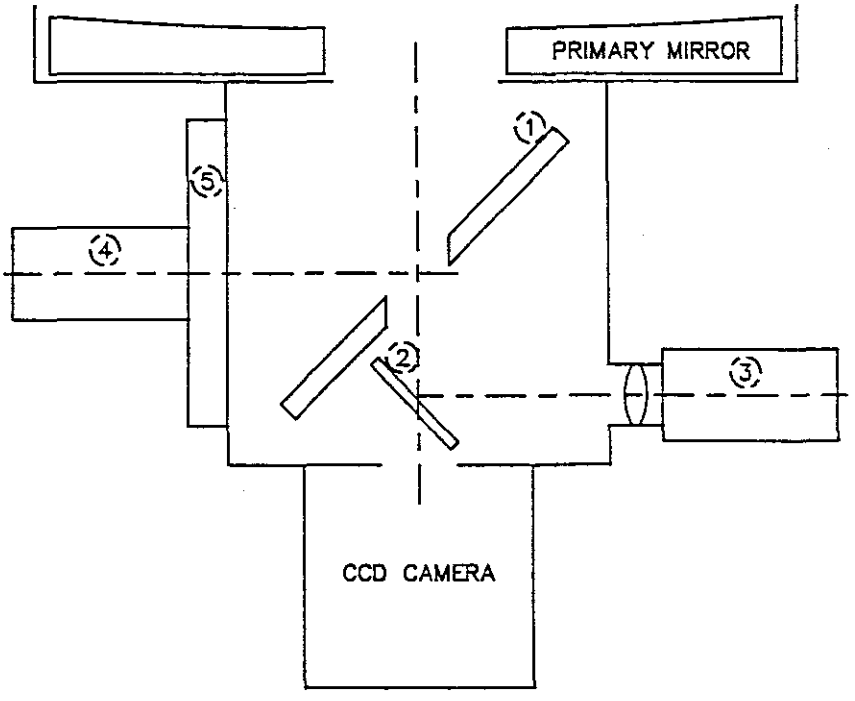


Fig. 2.

The Offset Guide Mirror (1) is a circular flat mirror , with a central hole . It is mounted at 45 degrees to the optical axis on linear bearings such that the hole can be centred on the optical axis or positioned to one side of the axis. The Rear Viewing mirror (2) is a small flat mirror , also mounted at 45 degrees to the optical axis and on linear bearings , able to move in and out of the star beam.

The T.V. camera (3) is a low light level integrating camera used in place of an eyepiece , with a monitor screen in the telescope control room.

The magnification of the autoguider probe (4) optics is such that it can see only a small part of the telescope's field of view. It is thus necessary to mount it on a moveable table (5) so it can be positioned to see a star anywhere in the field of view. The developement of this moveable table , called the XYslide , is the subject of the major part of this thesis.

2.3. Use of the Acquisition System.

A typical sequence of operations is described in using the acquisition system to identify and centre a program star and set up a guide star.

The telescope is first moved to point at the coordinates of the program star. Then the following is done.

1) With the offset guider mirror in the CENTRE position and the rear viewing mirror IN the beam , the star field is examined on the acquisition TV system. The program star is identified , and using the telscope slow motion controls , the telescope is moved to centre the program star on the telescope optical axis.

2) With the help of a finding chart , a suitable guide star in the field of view is identified and the XYslide is moved the precalculated position X' , Y' so that the autoguider probe is centred on the guide star. The astronomer looks through the autoguider eyepiece to check that the probe is correctly positioned. If not the manual control pushbuttons are used to drive the XYslide to the correct position.

3) The rear viewing mirror is moved OUT of the beam.

4) The autoguider GUIDE function is started.

5) A short exposure is taken on the CCD camera to check that the program star is suitably central in the CCD frame , and if necessary the telescope pointing is adjusted.

6) An exposure is taken for data acquisition. The exposure time will depend on the magnitude (light intensity) of the object.

3. Discussion of design Principles.

With reference to the required specifications (see App. A pg A–1) , the design decisions to be made are :

3.1. Choice of Computer System.

The SAAO Electronics group had standardized on using Motorola 6800–series processors in the SABUS (South African Standard Bus)environment for small computer controlled projects. Due to the expertise and experience available in this low–cost , reliable system , it was decided to implement this project in the SABUS environment. As the SABUS is designed to be compatible with the INTEL series of microprocessors , the 6809 cpu board design had to generate various control signals not output by the microprocessor itself.

In retrospect , having completed the project , the choice of CPU was not ideal , as the combination of high level programming language and performance required approached the limits of the CPU capabilities. If the software had been implemented in assembler language or a faster processor used there would have been fewer constraints.

3.2. Choice of the User Interface.

A traditional control panel with switches and status annunciators was considered as a user interface , with the advantages of simplicity and single 'keystroke' control. This option was discarded in favour of a standard computer terminal due to the inherent flexibility of such a system. The advantages of single keystroke operation were retained by programming the terminal function keys to implement certain of the most often used commands. The video screen is also very useful for status information.

Consideration was also given to using the terminal screen in a screen mapped mode , to provide permanent real time status information on mirror and autoguider probe positions. This option was not implemented due to the software overhead required for cursor addressing , and also text string handling , which can get very complicated and unnecessarily slow the processor down.

3.3. Choice of Motive Power.

Three separate units needed to be moved to satisfy the requirements of acquisition system automation. These are :

a) The autoguider probe.

b) The Offset Guider mirror in the acquisition box.

c) The Rear–viewing mirror in the acquisition box.

The acquisition box mirror positioning system was already defined. It had originally been designed as a manually operated pneumatic system , where manually operated valves controlled a pneumatic cylinder and piston which moved the mirror slides. The manual controls were situated on a control panel attatched to the acquisition box. As the system had proved to be reliable , it was decided to retain most of it and only change the manually operated control valves for electro–mechanical equivalents. The design decisions involved the modifications required to provide local/remote control of the pneumatic system.

The existing manually operated XYslide used two leadscrews rotated by means of crankhandles to move the autoguider probe mounting plate. The leadscrew method was retained , due to the simplicity and ruggedness of the design , so the design decisions required were what motive power and what form of positional feedback to use. A D.C. motor servo system was considered due to the efficiency and small physical size of the motors and drive electronics. The alternative was stepping motor drive. The stepping motor system was chosen due to it's suitability to the velocity mode requirements of the specification. To improve efficiency chopper drives were chosen for the stepping motor translators.

3.4. Choice of position feedack System.

Position status is required of both mirrors in the acquisition box and the autoguider probe.

It is a requirement of the specification that the acquisition box mirror positions be monitored both in and out of the optical beam , so that if the mechanism seizes up at some intermediate position where the mirror obstructs the optical path the system can flag an error. The simplest method of encoding this is with microswitches at each end of the linear travel.

Monitoring the position of the X–Y motion of the autoguider probe is complicated by the accuracy and resolution required (see specifications in Appendix A pg A–1) , and the circular nature of the limits to movement. High resolution linear potentiometers , LVDT transducers and linear optical encoding systems were considered , but were discarded due to large physical size or cost factors. A dual system of low resolution , gear–driven ten–turn potentiometers for absolute position sensing , and software step counters for incremental , high resolution position sensing was implemented.

The ten turn potentiometers are used as voltage dividers and the position–dependant voltage is digitised with Ten–bit resolution. The potentiometers are driven through an 8:1 gear ratio which results in 9,4 of the available ten turns being used to encode the full distance of travel of the probe. This gives a resolution of approximately 31 steps of the stepping motor (27.7 degrees of rotation), equal to 155 micron linear movement of the autoguider probe.

The software step counters are initialized with a count of zero when the probe is at the centre of the movement area , and subsequent step commands to the stepping motors increment or decrement the counters , depending on direction (up/down or left/right). Thus the probe position is known in cartesian coordinates in terms of the number of steps from a centre zero position , the sign (positive or negetive) being significant. The calibration of the stepping motor is :

1 step = 5 microns linear movement.

Thus the user can be given the coordinates of the probe position in terms of millimetres . This system is thus open loop , relying on the stepping motors never to miss a step command. In practise stepping motors cannot miss a single step initiated by the drive electronics , but can jump four or eight steps , depending on the mode of operation (4 or 8 step mode) , the instantaneous load , or the step frequency.

To ensure the software step counters are accurate , an encoding system consisting of a slotted disk with two optical limit switches is mounted on the stepping motor shaft. The encoding disk has ten radially cut slots equally spaced around the circumference , thus there is a slot every 40 steps ofthe stepping motor. One slot is longer than the other nine , defining a reference position once per revolution. One of the optical limit switches is set to detect all ten slots , the other detects the one long slot only, producing two signals FPOS and FREF , where FREF occurs once per revolution and FPOS ten times per revolution of the stepping motor. See Fig. 3.

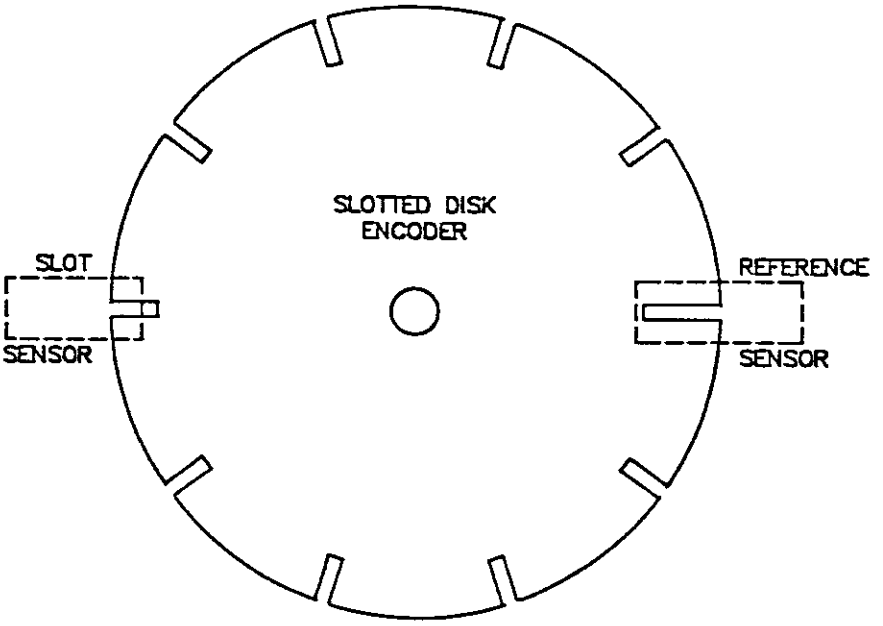


Fig 3.

The software uses this facility as follows :
A modulo 40 counter is kept in software , initialized to 39 when FREF and FPOS are true. Before the start of each move sequence of the X–Y slide this modulo 40 counter is loaded with a value calculated from the current position , such that it will underflow at the instant that a slot signal (FPOS) should be true. The move function is then initiated , and the modulo 40 counter is decremented by one for each step of the stepping motor. When the modulo 40 counter underflows , FPOS is checked. If the signal is correct (i.e. a slot is present) , no error is flagged and the modulo 40 counter is loaded with 39. If the signal is not true (no slot present , therefore the probe is not in the correct position) an error is flagged and the move function is aborted.

The one long slot (FREF) is used to define a reference position once per revolution. This is used in conjunction with the absolute encoders during the initialization process to define the centre of the movement area.

3.5. Implementation of Velocity Mode.

The requirement of the so–called 'velocity' mode called for moving the autoguider probe at a very accurate , slow constant velocity with a high resolution on speed changes. The scheme chosen was to use an accurate temperature stabilized crystal oscillator and frequency synthesizer combination. The frequency synthesizer design gives nine decimal places of resolution , of which six are software programmable and three are preset in the hardware. A characteristic of the rate multiplier integrated circuits used in the frequency synthesizers is the instantaneous variation of the output frequency caused by the randomly varying markspace ratio. This would cause unacceptable errors while moving

the probe. This disadvantage is overcome by following the synthesizer with a divider stage set to divide–by–five thousand , which smooths out the frequency variations to within acceptable limits.

3.6. Choice of Software Environment.

The SAAO Electronics group has a Motorola EXORcisor microcomputer with a 6809 processor which is used as a software developement station. An OMEGASOFT PASCAL compiler was obtained for the system , to develop the software required for this project.

The EXORcisor has an interface for driving the SABUS (ref SAAO internal documents) , so during program development the target system input/ouput hardware is used.

The OMEGASOFT compiler has the useful features of producing ROMable code and supporting merging assembly language code into the high level code. A few procedures were written in assembly language due to execution speed requirements or because structures were required that PASCAL does not efficiently provide. Interrupt routines were also written in assembler to reduce execution time to a minimum.

4. Hardware implementation.

Due to previous experience with noise interference (EMI) problems in the telescope environment , communication between the computer hardware and the instrument was carefully considered. The requirement is to keep cable lengths short and to use well buffered signal lines. The best solution from this point of view is to mount the microcomputer on the telescope mirror cell , as close as possible to the instrument it has to control. This approach was not practical due to lack of available space and weight limitations , so the microcomputer and stepping motor translator card frames were mounted in a rack in the telescope control room , and careful attention given to the cabling /connections and commnication standard used between the microcomputer and the instruments on the Telescope.

The status/control communication between the microcomputer and the instrument on the telescope was implemented using the RS422 differential communication standard. The nature of this standard required doubling up on the number of wires and larger connectors , thus adding to the cost. This was deemed to be justified due to the improved noise performance. In addition , many signals are optically isolted to prevent the possibility of earth loop problems. All power supplies are left floating with respect to safety ground.

The microcomputer system was built in an SABUS card frame. A second cardframe (called the 'musclebox') in the same equipment rack housed the Stepping motor translators and power supply , with optical isolation on all signals between the two cardframes to prevent possible interference from the large currents being switched in the stepping motor translators.

On the telescope a third smaller cardframe housed power supplies and circuitry for conversion between TTL and RS422 signals. This cardframe is used as a plugboard for interconnecting all the units on the telescope , as well as being a convenient place to mount the pushbuttons for controlling the autoguider probe position. A block diagram of the system is represented in Fig. 4.

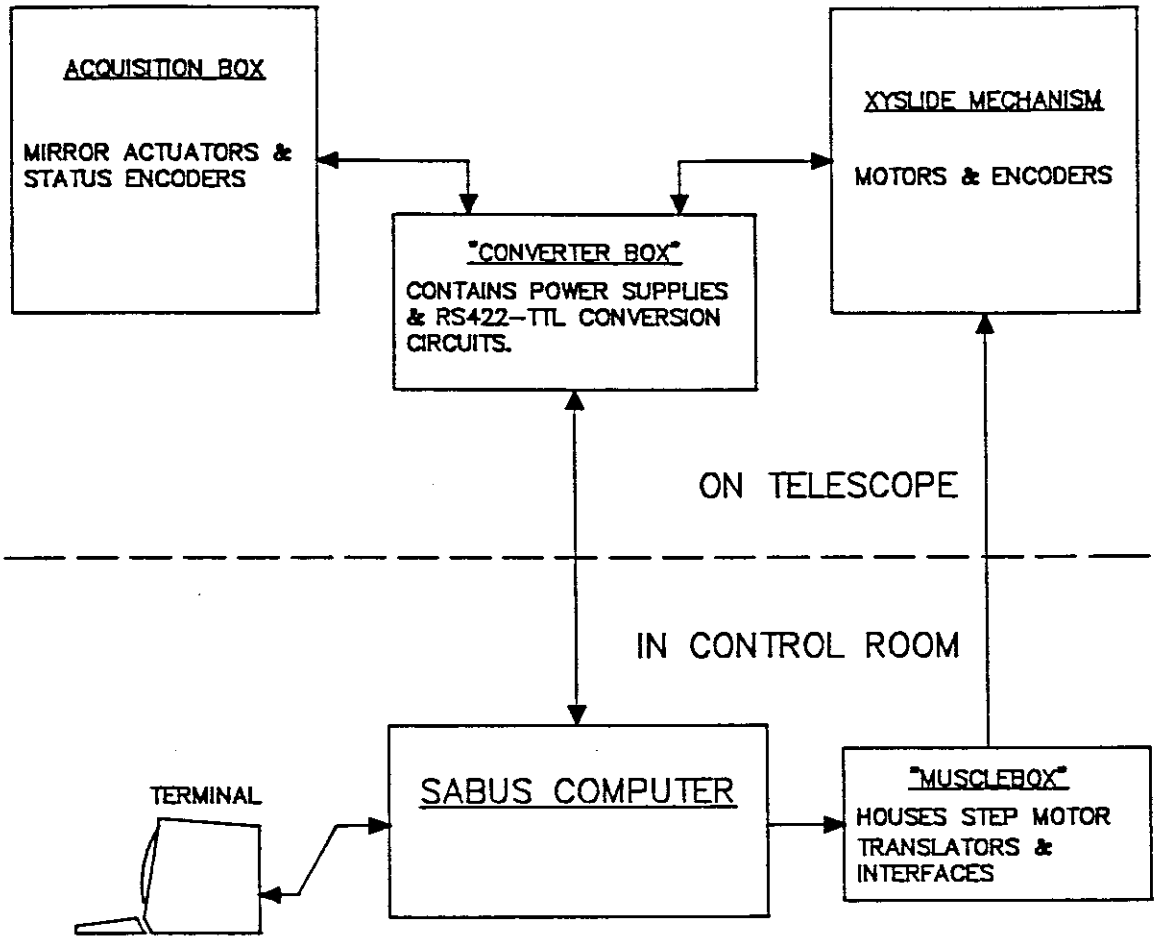


Fig 4.

4.1. The Control computer.

The computer was implemented on the SABUS , in a standard commercially available card frame. Input/Output interfacing was implemented where possible with with commercially available units. Fig. 5 is a block diagram of the SABUS crate. Where these were not available or not entirely suitable , proprietry interfaces were designed and built.

The computer system is assembled from the following modules :

6809 CPU :

This module was designed by G.F.W. Woodhouse of the SAAO. It has a 64K EPROM on board and 2K of RAM. The CPU runs at 1MHz.

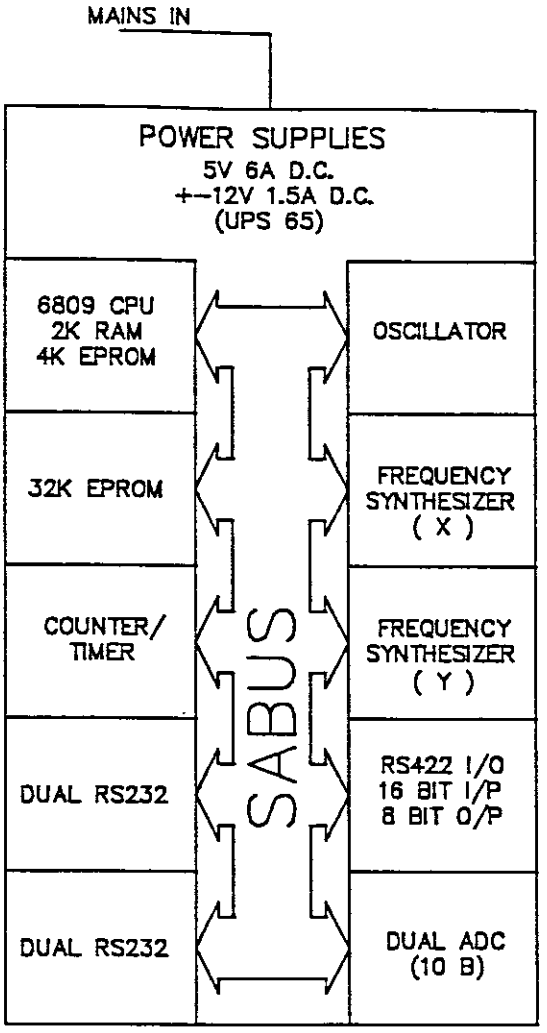


Fig. 5.

32K RAM :

This is a commercially available module produced by Basic Electronics.

Triple Counter/Timer :

This module was designed by G.F.W. Woodhouse of the SAAO. It uses an INTEL Triple counter I.C.

Dual RS232 Interfaces.

These are commercial modules produced by Basic Electronics with two independant RS232 serial ports. Provision is made for communication with the terminal for user input , and for communication with the telescope control slow motion drive and instrument control computers.

The following modules were designed by the author with the exception of the RS422 I/O interface which consists of circuitry designed by the author built on a commercially available wire wrap I/O board which comes complete with all bus buffering and decoding circuitry. These circuits are described in some detail.

4.1.1. Dual Analog to digital Converter.

Refer dwg No. E3–0271, App. D.

The dual A–D converter uses two National Semiconductor 10 bit converters with the following specifications :

Type	: ADC1001
Resolution	: 10 bits
Conversion time	: 200 microsec.

On chip clock generator.
Single 5V supply.
Needs 2.5V reference.
0 to 5V analog input range.
Logic I/O signals are TTL levels.

The 10 bit output is read out as two bytes automatically by strobing the RD line, format left justified and high byte first. The least significant 6 bits of the second byte are set to zero.

Circuit Description.

Addressing : The card address is switch selectable in the standard SABUS format. A switch in the ON state implies the associated address bit must be a 1 to select the card. The circuit occupies four addresses , one of which is unused. The two least significant bits are decoded by IC 1 to directly select the circuits on the card. The address map is as follows :

	Bit1	Bit0
ADC1	0	0
ADC2	0	1
INTR status	1	0
Not used	1	1

The ADC circuits : The ADC1001 can be configured to operate in one of three modes , selectable by links on the P.C. Board. (See Fig. 6).
The modes are :

	Link
1) Continuous conversion	A, D & E
2) Program control, no interrupt	B & C
3) Program control with interrupt	B, C & F

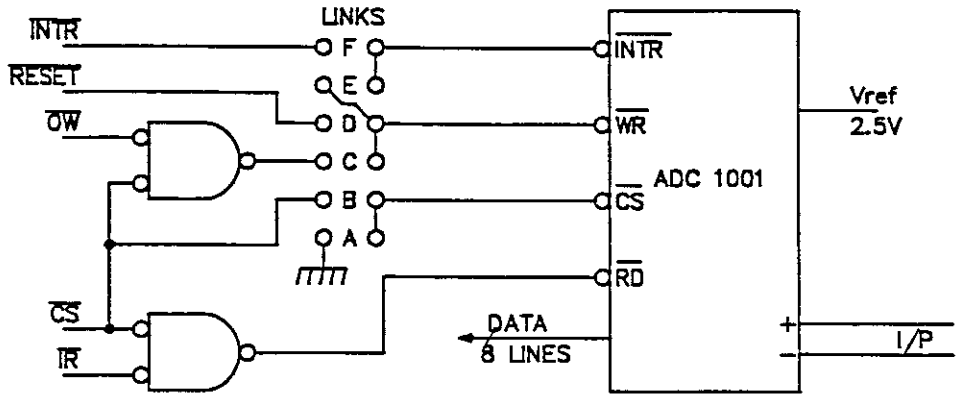


Fig. 6.

Mode 1 – Free run conversion : The computer hardware RESET toggles the WR input and starts the first conversion. Thereafter the EOC signal (INTR o/p) restarts the process via link E, so it becomes self starting. Link A permenently selects the CS i/p, so the output can be obtained by strobing the RD line twice to obtain the two bytes of data.

Mode 2 – Program control, no interrupt : The conversion is started by strobing the WR line with CS held low. Data is read by either waiting for more than 200 microseconds before reading, or monitoring the appropriate bit (6 or 7) of the 8 bit input port (IC 13) and waiting for it to go low.

Mode 3 – Program control with interrupt : conversion is started as above but end of Conversion causes an interrupt via link F. The interrupt routine must poll the 8 bit input port bits 6 & 7 to identify which converter caused the interrupt.

The 8–bit input port : This is primarily for identifying which of the converters has caused an interrupt. Identification is as follows :

	Bit 7	Bit 6
ADC1 (address 00)	1	0
ADC2 (address 01)	0	1

Use is made of the other 6 bits by bringing them out through the front panel on a delta 15–S connector. These inputs can be terminated by inserting a DIL resistor pack in position IC 13, and by means of link U or D can be pulled high (link U) or low (link D).

The analog inputs are available either on the delta 15–S connector or on two LEMO 0304 sockets. 5V power is available on the LEMO and delta connectors.

4.1.2. The RS422/TTL I/O Interface.

Refer dwg.No. E4–0273, & Basic Electronics dwg. 701/1, App. D.

This module is the main status/control port for the acquisition system hardware. It is built on a commercially available general purpose Wire Wrap board by Basic Electronics. The board is supplied with all address decoding , bus buffering and interrupt circuitry , and has an uncommitted 24 bit programmable peripheral interface with an area of the board avaiable for application specific circuitry.

The requirements of the acquisition system hardware meant that this circuitry had to be a mixture of RS422 differential input/output and optical isolation input.

The interrupts are not used. Note the modification to the board interrupt circuit to prevent noise generating an interrupt.

The 8255 is programmed in mode 0 as 16 input and 8 output lines. Port A and B are defined as inputs, and port C is an output. The control byte is $92.

The inputs and outputs are connected through a DIN41612 AC64 connector on the front panel.

Circuit Description :

Port A : All 8 bits of this input port are status lines from the XYSlide mechanism on the telescope, and are transmitted as RS422 signals. Thus IC 8 & 13 (RS422 receivers) convert the signals to TTL levels before feeding in to the 8255. The RS422 signals are terminated in 150 ohms.

Port B : Bits 0 – 3 are status lines from the X–Y mechanism and are RS422 as above. Bits 4 – 7 are status from the musclebox, and are optically isolated (IC 12 & 17) and buffered before feeding in to the 8255. Note that XFAULT and YFAULT signals are inverted before going to the 8255.

Port C : Only bits 0 – 3 of this port are used to control the acquisition box. IC 9 converts the signals to RS422 levels. Bits 4 – 7 are buffered in IC 10 and outputted as TTL levels . They are not used at present.

The oscillator frequency can be trimmed by means of a 10–turn potentiometer on the P.C. board and a trimmer capacitor internally mounted in the oscillator.

For the XYslide, which requires a master clock of 560KHz for velocity mode, the oscillator output is divided down by a factor of ten before going to the prescaler synthesizer. This is due to the HCD–80 oscillator unit used having a minimum frequency of 4.5MHz.

4.1.3.2. The Frequency Synthesizer.

Refer to drawing No. E3–0127, App. D.

This circuit is designed to accept the output of the Oscillator card and allow programmable control of the output frequency, although any TTL level signal can be used as a source. It also has a seven–bit latched output port. All inputs and outputs are fully buffered TTL signals.

A post divider circuit offers four switch selectable divisor factors to smooth the output pulse train.

The frequency synthesizer output function follows the formula :

$$F_{out} = M \times F_{in}/1000000$$

where M is a six digit decimal number in the range 0 – 999999. The number M is set up as a six figure BCD number in the 24 bit output port under program control.

Circuit Description.

Addressing : Address bus bits 3–7 are switch selectable in the standard SABUS fashion. If a switch is in the ON position the corresponding bit must be a 1 to select the card. Bits 0–2 are decoded as follows :

	AB2	AB1	AB0
7–bit latched O/P	0	0	0
	missing codes not used.		
Synth. control LSB	1	0	0
Synth. control	1	0	1
Synth. control MSB	1	1	0
8255 control byte	1	1	1

These address bits are combined with CARDSEL to produce DATASEL (7–bit latched O/P) & SYNTHSEL in IC 8.

The synthesizer : The synthesizer is formed by IC's 9–11 & 14–16, controlled by the 24 bit number on the O/P of IC12, the 8255. The synthesized output is obtained at IC13 p8. This pulse train is then divided in IC's 18 & 19. The divisor factor is selected by the 4–way DIL switch with the following truth table :

Divisor factor	sw1	sw2	sw3	sw4
200	Cl.	X	Cl.	X
500	Cl.	X	X	Cl.
2000	X	Cl.	Cl.	X
5000	X	Cl.	X	Cl.

Where X = Not allowed.

The synthesizer I/P and O/P are fully buffered. The I/P is via a LEMO 0302 connecter and the O/P is on the Delta 9–S connecter on the front panel.

The latched O/P, (IC17), using the 8212, is limited to 7 bits by the constraints of the 9–way output connecter, which already has the frequency synthesizer output on it, thus only bits 0–6 are used.

Programming : There is only one device on the circuit that needs programming. The 8255 PPI is used in mode 0 only, control byte $80 to configure it correctly.

A peculiarity of the circuit is that after a hardware RESET, due to the I/O pins of the 8255 being programmed to inputs, the frequency synthesizer will have an output of :

$$F_{out} = 0.999999 \times F_{in} / D$$

where D is the divisor factor.

4.2. The Musclebox.

This is the second cardframe situated in the control room rack housing the power supplies and drive translators for the stepping motors , and optical isolation interfaces for the control signals coming from the microcomputer. Fig. 8 illustrates the main components.

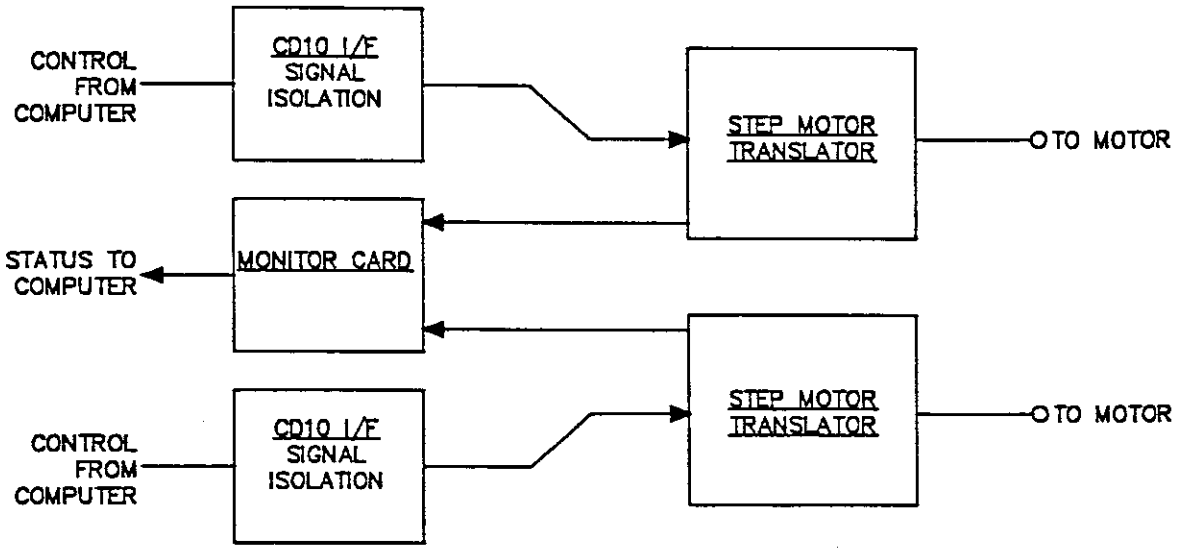


Fig. 8.

The units in the musclebox are as follows :

4.2.1. The CD 10 I/F.

Refer Dwg. No. E3–0162

The CD 10 I/F was designed to interface the Digiplan CD 10 stepper motor driver to the SABUS computer system. It provides the following features :

A) Optical isolation of all control signals from the computer.

B) Single step test facility.

C) Annunciator LED's for direction of rotation, steps in progress, Phase Zero, and CD 10 fault condition.

Circuit Description : The control signals come in on a front panel connecter and are optically isolated in IC's 1–4. If required, the FAST and SLOW signals must be implemented by inserting links 2 & 3.

The STEP input is controlled by STEP ENABLE – this would normally be used to control the input from an external pulse source such as the SABUS frequency synthesizer. The incoming step/pulse signal is shaped in IC 12.

All outputs of the circuit are open collector high voltage drivers to comply with the stepping motor translator input requirements.

4.2.2. The Stepper Motor Translators.

These are commercially available units from Digiplan. They are a bipolar switching design which makes for much improved efficiency. The approximate 20KHz switching frequency generates EMI which degrades the performance of the CCD preamplifiers , so the stepping motors are kept unpowered (de–energised) unless movement of the autoguider probe is required , and a CCD camera data transfer (data readout) is only initiated after the stepping motors have been de–energised..

4.2.3. Monitor Module.

Refer to drawing no. E4–0279, App. D.

This card provides a means of getting the FAULT and Phase Zero status of the two stepper driver modules back to the computer (IC 1), as well as power supply monitor points and visual indication of the stepping motor phases. Two–colour led's are used to monitor the phases. The resistor/diode circuit aroiund the led is designed to get nearly equal brightness from the two colours.

4.3. The Converter Box.

Refer Dwg. No. E3–0275, E3–0276, App. D.

This is the 1/4 size crate that is bolted on to the left hand side of the acquisition box. All communication with the computer is routed through it. A block diagram is seen in Fig. 9. It has two modules as follows :

The Power Supply : This unit has a 5v 3A supply for all the TTL logic and the position encoders, and a +/–12v 240mA encapsulated supply for the autoguider graticule and (regulated to +–5v), for the autoguider preamp.

The Logic Circuit : This module contains the logic circuitry for conversion between RS422 levels and TTL levels. There is also optical isolation for the six signals to and from the acquisition box. The front panel carries led status of the control and status signals. There is provision for a remote handset to duplicate the X–Y position control functions of the pushbuttons on the top panel.

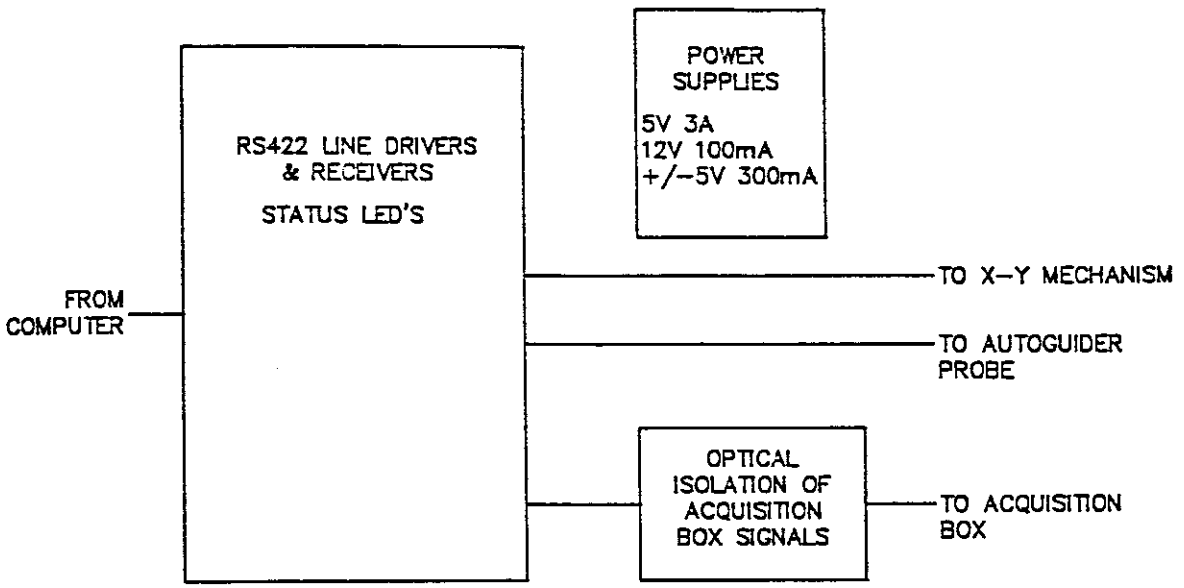


Fig. 9.

4.4. The Acquisition Box.

Refer Dwg. No. E3–0259, E3–0260, E4–0261.

The modifications to the existing acquisition box comprised rewiring the unit to mount a power supply and control circuitry in the control panel , and changing the layout of the control panels to allow for computer or manual control of the mirror positions. The control circuit provides for :

a) Manual or computer control of the position of both mirrors.

b) Manual or computer control of Eyepiece graticule illumination (On/Off only)

c) Interlocking graticule illumination of the Rear viewing eyepiece with mirror position such that the graticule illumunation can be on only when the mirror is in the beam.

d) Switch selection of control of the mirror position from the control panel or from a remote controller(the microcomputer).

e) Mirror position status lamps on the control panel , and status indication to the remote controller.

f) Mirror position status to the data acquisition control computer.

5. The Software.

(See Appendix C for program listings and flowcharts.)

The major part of the program was written in PASCAL using an OMEGASOFT Pascal compiler , with a few sections written in assembler. The source code was split into a number of separate "include" files to facilitate editing.

The command set can be broadly divided into four catagories. (See Appendix B pg B–9 for the complete command set). These catagories are :

A) Commands to do with moving the probe. eg VMODE , MOV , CENTR.

B) Commands to control the Acquisition Box.

C) Utility commands. eg LPOSN , FUNCSET , HELP.

D) Diagnostic commands. eg TEST , MOTON , MOTOFF.

Appendix B – The XYslides User Manual – has a full description on how to use these commands except for the diagnostic commands which are described in para. 6.1. below.

The main program flow is shown in Fig. 10. The software is structured such that most modes of operation are serviced as procedures which are exited only on completion of the command. This implies the program loop time can extend to as much as 15 seconds as in the case of the MOV command.

An exception to this is the velocity mode function VMODE , and the pushbuttons mode. Stepping motor movement under VMODE is controlled by the programmable frequency synthesizers , thus the program only has to set up direction of movement and program the required speed into the synthesizers then enable the stepping motor translators. The probe then moves at the defined speed independant of any further software action , until the movement is terminated or the speed changed. The pushbuttons are checked once through every program loop (see flow chart in Fig. 10) and if a movement pushbutton is pressed the probe moves accordingly. Periodic position checks are required for both pushbutton and velocity mode to ensure the probe does not hit the mechanical limits of movement.

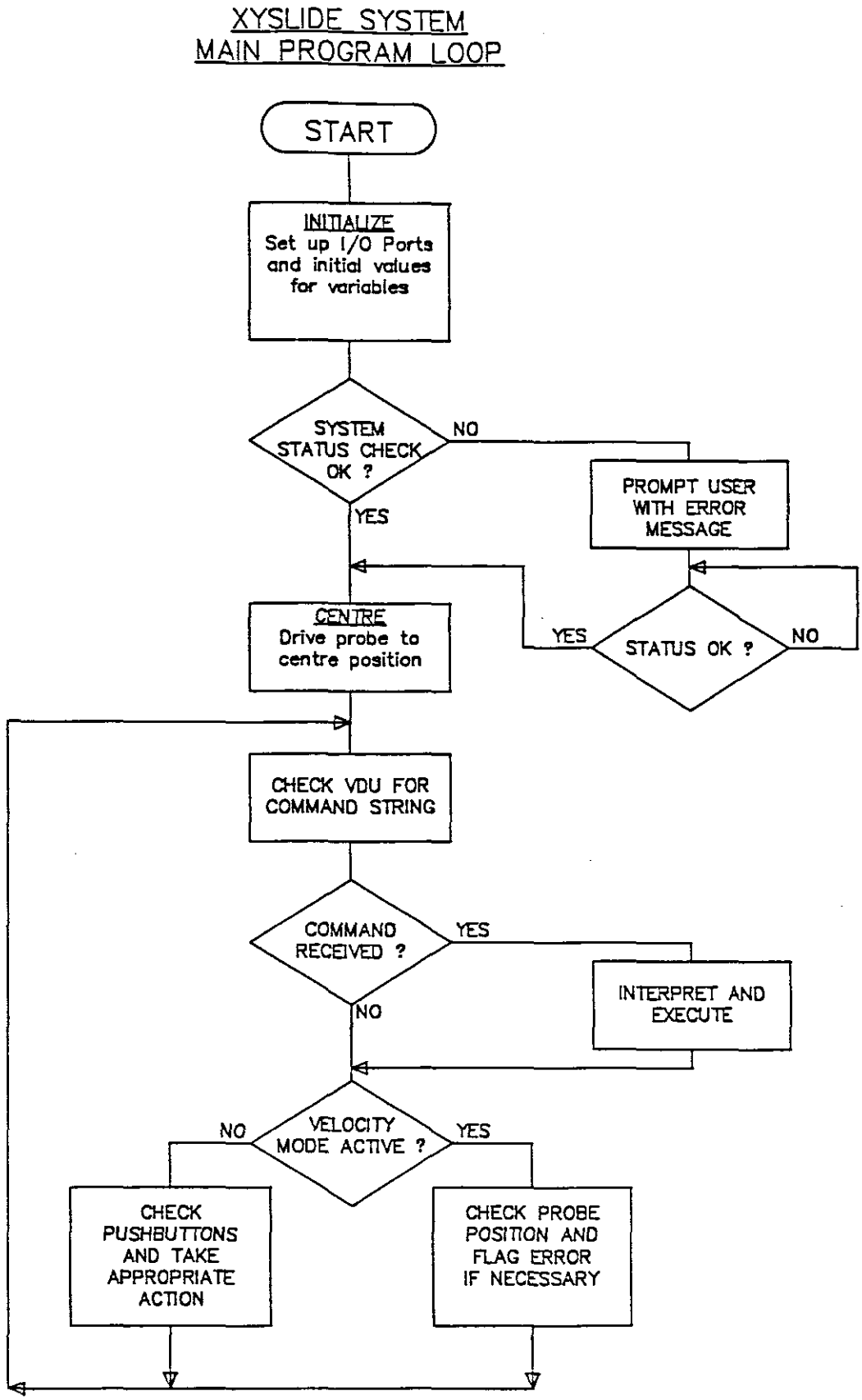

XYSLIDE SYSTEM
MAIN PROGRAM LOOP
START
INITIALIZE
Set up I/O Ports and initial values for variables
SYSTEM STATUS CHECK OK ?
NO
YES
PROMPT USER WITH ERROR MESSAGE
STATUS OK ?
YES
NO
CENTRE
Drive probe to centre position
CHECK VDU FOR COMMAND STRING
COMMAND RECEIVED ?
YES
NO
INTERPRET AND EXECUTE
VELOCITY MODE ACTIVE ?
NO
YES
CHECK PUSHBUTTONS AND TAKE APPROPRIATE ACTION
CHECK PROBE POSITION AND FLAG ERROR IF NECESSARY

Fig. 10.

A fast interrupt routine (FIRQ) for controlling the movement of the stepping motors was written in assembler to reduce execution time to a minimum. The routine repeats at a 2KHz rate (every 500 microseconds) with a 265 microseconds worst case execution time. The normal interrupt (IRQ) is reserved for a future update.

The Motorola 6809 CPU has three levels of interrupts – Non Maskable Interrupt (NMI) , Fast Interrupt (FIRQ) , and normal interrupt. The design of the CPU circuit board does not allow for interrupt management , and the three interrupt levels were assigned as follows :

NMI – Not Used

FIRQ– Control of stepping motors for normal movements

IRQ – *reserved for a future update*

The lack of interrupt management severely restrained the software design , as the terminal and pushbuttons could not be interrupt driven , but had to be serviced as part of the main program loop. The disadvantage of this approach is that if any part of the program loop lengthens appreciably (e.g. position checking calculations requiring square root functions) the terminal fails to respond , and user input is lost. Thus program loop time had to be kept to a minimum to prevent data being lost from the terminal/computer communication, or for there to be an unacceptably slow reaction time on the pushbutton controls. This feature is going to be changed as part of the IRQ update.

All commands to move the probe are mutually exclusive i.e. CENTR exor MOV exor VMODE exor PUSHBUTTONS. Software traps had to be inserted to prevent any possibility of simultaneous execution of these commands.

5.1. Diagnostic Tools.

The software includes some 'hidden' commands which do not appear in the 'HELP' menu. These functions are designed to make faultfinding a little easier. The functions available are :

MOTON Applies power to the stepping motors (ENERGISE function) , which are de–energised by default unless actually moving the slides. Useful for setting up the slotted disk encoders etc.

MOTOFF De–energises the stepping motors.

BUZZ Sounds the buzzer in the converter box on the telescope for approximately one second.

TEST Very useful! This command gives you five options of items to test. The options are :

1. Print status bits of STAT1 port.
2. Print status bits of STAT2 port.
3. Print hex number reading of X & Y potentiometers.
4. Do a Testmove.
5. Change hex value of centre co–ordinates.
6. Exit to command processor.

The eight–bit status ports have the following bit assignments

STAT1 (8 bits)

Bit				
	0	A/G 45deg mirror	0 = out	1 = in beam.
	1	Acqu. box COMPUTER/MANUAL select switch status		
			0 = manual.	1 = computer.
	2	Rear view mirror position	0 = in	1 = out beam.
	3	Guide mirror position	0 = in	1 = out beam.
	4	Xleft pushbutton		
	5	Xright pushbutton		
	6	Yup pushbutton	0 = inactive	1 = asserted.
	7	Ydown pushbutton		

STAT2 (8 bit)

Bit		
	0	Xref
	1	Yref
	2	Xslot
	3	Yslot
	4	Xfault
	5	Yfault
	6	XPh0
	7	YPh0

The potentiometer readings are given as 4 digit hex numbers. In the centre position the value should be approximately $200 ($0A either way is acceptable). Other typical figures are :

X Axis	extreme left	– $3C
	extreme right	– $3BD
Y Axis	extreme up	– $3C3
	extreme down	– $41

These figures will vary depending on which X–Y mechanism is being checked and whether it has been disassembled at any stage. The reason for the difference is that the potentiometers are driven through a gearbox , thus after disassembly it is unlikely that the gears will be meshed in exactly the same position.

The TESTMOVE function prompts for two sets of co–ordinates and then moves continuously between them and the centre position. This is useful for doing oscilloscope tests on any of the stepping motor control bits or the feedback signals (FPOS, FREF, PH0). Pressing any key will stop this test at the end of the current cycle.

The 'CHANGE VARIABLES' option should really only be used if there is a suspicion that the XYslide is not centering properly , or if the mechanism has been stripped and needs to be re–setup. It allows changing the reference values used in the calculations for finding the centre position during the initialization process.

5.2. Memory Map.

As the Motorola 6809 Processor does not have separate I/O and memory addressing capability , all input/output is mapped into the main memory. SAAO has standardized on allowing 256 bytes for input/output , starting at location $E000. Limiting the I/O to 256 locations simplifies the I/O port address decoding as only eight bits have to be decoded. The CPU card generates the required bus control signals to differentiate between memory and I/O accesses by decoding the address bus.

The memory is apportioned as per Fig. 11.

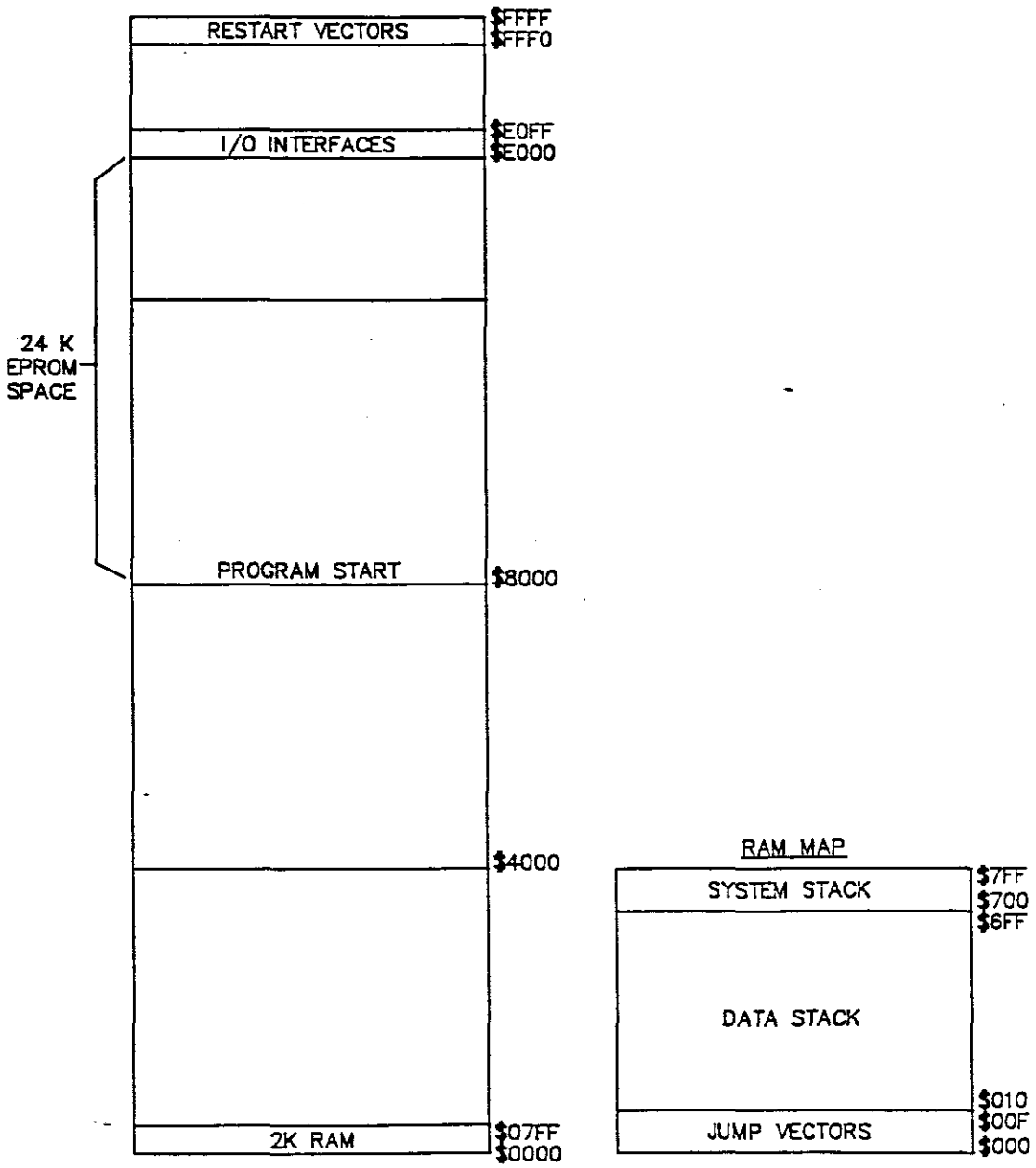


Fig. 11.

The I/O section of the map is apportioned as follows :

$E000 : X motor control
$E004 : LS Byte
$E005 : Middle byte frequency synthesizer step rate control
$E006 : MS Byte
$E007 : Synthesizer control word (set to $80)

$E008 : Y motor control
$E00C : LS Byte
$E00D : Middle byte freq. synthesizer step rate control
$E00E : MS Byte
$E00F : Synthesizer control word (set to $80)

$E010 : Terminal data
$E011 : Terminal port control word
$E012 : Nova data
$E013 : Nova port control word
$E018 : Slow motion drives data
$E019 : slow motion drives port control word
$E01A : Data – spare port
$E01B : control word – spare port

$E030 : not used
$E031 : timer card status
$E032 : Timer card control
$E033 : 8255 control word
$E034 : Counter 0 – pulse counter
$E035 : Counter 1 – FIRQ timer
$E036 : Counter 2 – IRQ timer
$E037 : Counter chip control word

$E040 : X axis A–D converter
$E041 : Y axis A–D converter
$E042 : X–Y ADC interupts & 6 bit I/P (not used)

$E080 : STAT1
$E081 : STAT2
$E082 : AQBOX control
$E083 : I/O control word

6. Problems encountered.

6.1. Stepping Motor Phase lag at high step rates.

An unforeseen problem was encountered when running the X–Y mechanism stepping motors at a 2KHz step rate. The instantaneous phase lag of the permenent magnet rotor w.r.t. the rotating magnetic field of the stepping motor (see fig. 12.) meant that when the program checked for the presence of a slot , the slots of the slotted disk encoder were not in the correct position , although the motor had not 'lost' any steps. Thus a movement error was flagged. This phase lag varies with speed and instantaneous load (which changes with telescope position).

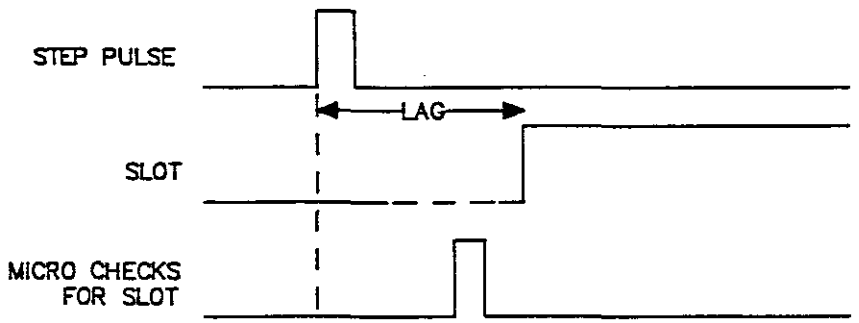


Fig. 12.

The slot width was manufactured to be equivalent to 6 steps of the stepping motor , which means that there is a three step leeway for each direction of rotation. In practice , due to the difficulty of setting the position of the optical limit switch detector accurately in the middle of the slot , the leeway tends to be about 2.5 to 3 steps on one side. This , combined with the natural oscillation around a step position , is not enough to ensure that the slot will be true at exactly the right moment.

The solution to the problem was implemented in the software , in the FIRQ interrupt routine. At the time when a slot is supposed to be true , a counter is initialized with a value of 2. The counter is then decremented as each step is performed , and a check is done on the slot. If the slot status is true before the counter underflows , no error is flagged. If the counter underflows before the slot status comes true , an error is flagged and the move sequence is aborted , with appropriate error messages being printed on the terminal. The flowchart in fig. 13 illustrates this.

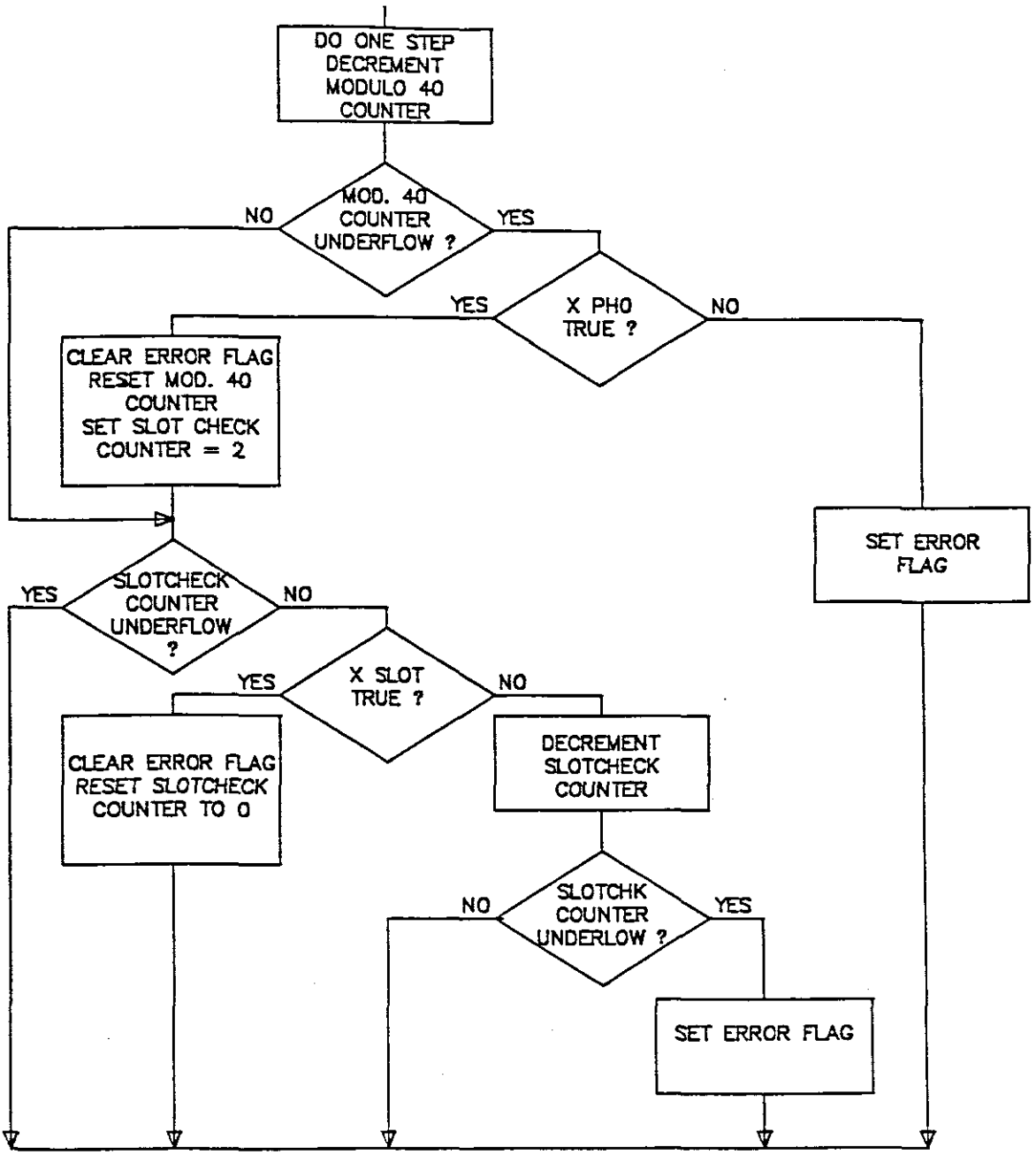


Fig. 13.

6.2. System 'Health' Check on Program Start–up.

As the power supplies for the separate units that make up the system are individually controlled , it is possible to start the system up without switching on power to all of the units. Due to the number of plug–in connections , it is also possible to start up with some of the connections not made.

This combination of circumstances can result in the program reading false data from the encoders when initializing the probe to the centre position. This can lead to the probe being driven into the mechanical limits of movement , which is undesirable!

An oversight at the design stage meant that there was no direct method of checking that the encoder circuits and musclebox were powered up before starting to move the probe to the centre position during the initialize sequence.

The solution was found to be to check the data obtained from the position encoding potentiometers against a minimum valid value. If the reading obtained is less than the minimum , this is interpreted as meaning the encoder reading is invalid , and an appropriate error message is given to the operator to check the connections and power supplies to the encoder. See the flowchart in Fig. 14.

The musclebox is checked in a similar fashion by checking the FAULT and PH0 status signals from the stepping motor translators. If the FAULT signals are true , or if the PH0 signals are false , an error condition is flagged and the operator is prompted to check the hardware.

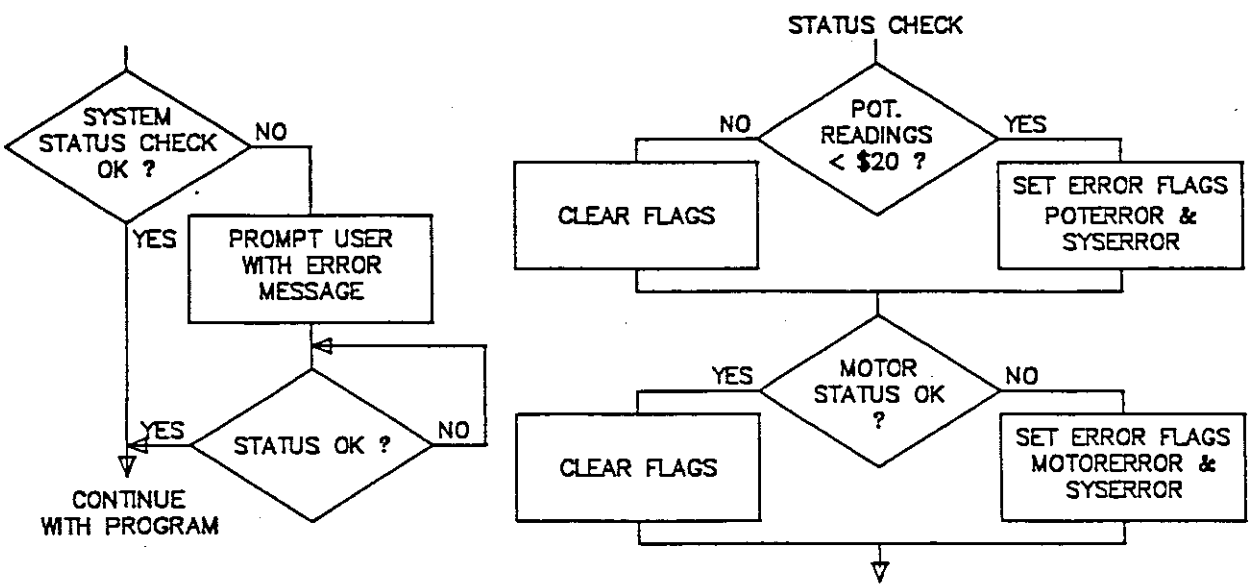


Fig. 14.

6.3. Response of Keyboard.

Due to the lack of interrupt management facilities in the target system , the terminal keyboard entry is serviced as part of the program loop. The program loop time thus had to be kept below about 2mS to ensure capture of every character from the terminal (4800 baud rate = 3.1mS per character). This was not possible in all cases , especially where calculation routines performed square root functions , thus there are periods when the keyboard 'dies'. The solution to this is to make the keyboard interrupt driven. This will be implemented in a future update where the interrupt will occur at 1mS intervals , each time checking the keyboard entry port for waiting characters.

6.4. Alignment of Slotted disk Encoder.

The alignment of the slotted disk encoder on the stepping motor shaft with the optical limit switch detectors used to produce the signals FPOS and FREF is critical and rather difficult.

The slots in the encoder disk are machined to the equivalent of 6 steps of the stepping motor to allow for the natural oscillation around a step position of a stepping motor. This slot could not be made any wider as the stepping motor phase pattern repeats every eight steps , and the slot was required to be unique to only one eight–step pattern. The slots were originally made much narrower , but due to the phase lag problem explained in section 6.1. above were widened somewhat.

The process of aligning the stepping motor encoding disks is part of the X–Y mechanism set–up procedure which is explained below :

6.4.1. Setting Up the X–Y Mechanism.

This procedure has three steps as follows :

a) Manually move the probe carriage to the centre of the movement area as defined by the dimensions X in Fig. 15.

b) Rotate the Ten turn potentiometer until a reading of $200 is obtained using the TEST command , then mount it in the gearbox , being careful not to rotate the shaft. Do this for each axis.

c) Align the slotted disk encoder on each shaft as follows :

1) Set the system up on a bench and connect up everything. Switch on in the normal way. The XYslide will drive to the centre point (The procedure can be done on the telescope, but it is a bit awkward.)

2) Set the X–Y mechanism in the centre of the movement area by manually rotating the the leadscrews and setting the dimension X shown in fig. 15 to 77.5mm.

3) Use the command MOTON to energise the stepping motors. Ensure the SD2 stepper drivers are at Phase zero.

4) Loosen the grubscrews holding the slotted disk on to the motor shaft and rotate until both REF and SLOT signals are true. (This can best be monitored by watching the appropriate led's on the converter box front panel.) Ensure that the longest slot activates the REF optical limit switch.

5) Rotate the slotted disk on the shaft while watching the led's to obtain the centre point of the slot over the optical switch. Tighten the grubscrews slightly.

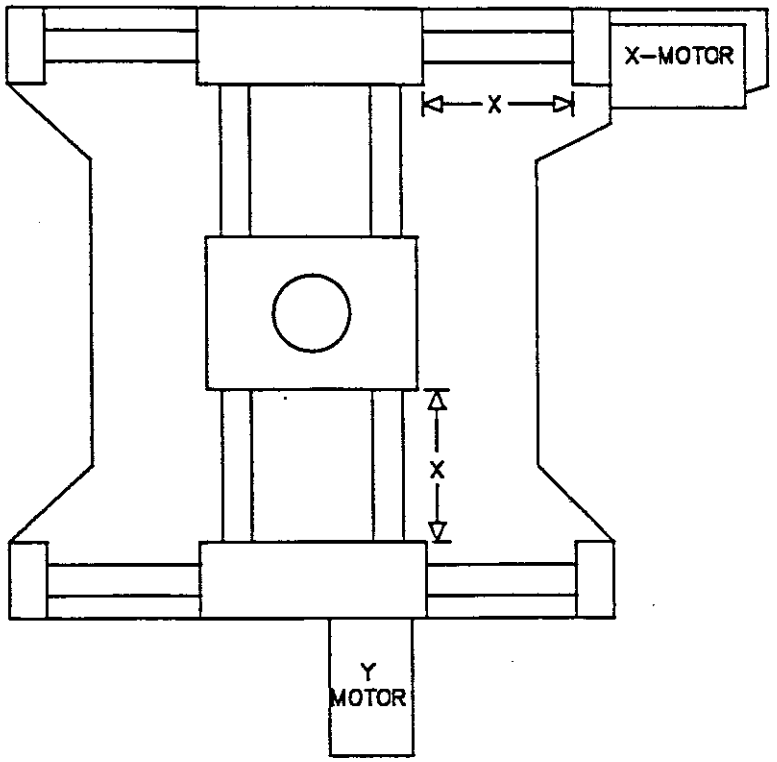


Fig. 15.

6) Use the single step function of the CD10 I/F with a test rig to control the direction input. Rotate the stepping motor in each direction until the led's switch off. The led's should go off after an equal number of steps in each direction from the centre as defined by phase zero.

7) Repeat steps 5) & 6) until satisfied that the slotted disk is well centered. Finally tighten the two grubscrews. DO NOT OVERTIGHTEN as this causes the grubscrew to nick the shaft which makes adjusting it in future nearly impossible.

More precise control of the optical limit switch position is necessary to enable proper adjustment under dynamic conditions. This will be rectified by re-designing the mounting arrangement of the optical limit switches.

7. Test Results.

Due to the inherent accuracy of the stepping motor (5% of a step position = 0,25 micron) and the resolution of the feedback (equal to one step) , the only parameter to test is the accuracy of positional repeatability of the X–Y mechanism. This is , in effect, a measure of the mechanical backlash in the drive mechanism. This parameter was tested using a dial gauge to set up a reference position , the probe was then moved away from this position and back again , and a reading taken of the error. The test was repeated for different directions of movement in both axes , in the horizontal plane. The test results are as follows :

X Axis :

Test No.

1. overshoot .010 micron
2. overshoot .000
3. overshoot .004

Y Axis :

Test

4. overshoot .000
5. overshoot .003
6. overshoot .003
7. overshoot .008

Average : 0.007 micron

Thus the average error of position repeatability is 7 microns (equivalent to 1.4 steps of the stepping motor). The specifications (see appendix A) called for 200 microns.

For other results see appendix A. for a comparison of achieved performance versus required specifications.

8. Future Plans.

The Acquisition control system is planned to be merged with the new Autoguider currently under developement. The merger involves additions to the microcomputer hardware and a substantial update of the software by merging two separately developed programs.

A second system will be manufactured for use on the 1.0m telescope at Sutherland , on which a Charge Coupled Device Camera has been in use for some years. Many of the remote control features of the system will not be used on this telescope as there is at present no closed circuit TV system available.

9. Glossary of Terms.

Autoguider. – A system which , with the telescope slow motion drive system controls the telescope following a star track across the sky. Many factors can influence a telescope ability to accurately follow a star , among which are :

a) Machining tolerances in the drive gearing.
b) Refraction caused by the atmosphere.
c) Flexure of the telescope structure.

The autoguider has a light detector which can detect minute relative motion between the star and the telescope , and generates an error signal which controls the slow motion drive to counteract these movements. Thus the autoguider and slow motion drive system become a servo control system to keep the telescope stationary with respect to the star.

Control Room. – An enclosed area on the side of the observing floor which acts as the control centre for most of the operations involved in telescope control and data capture. All the control computers are in this area.

Guide Star. – A star in the telescope field of view other than the program star , used as a reference for guiding. The autoguider system would use this star to detect any movement between the telescope and the star.

Observing Floor. – The floor of the telescope building where the telescope is. This is normally the first floor of the building.

Program Star. – The star which is to be studied as part of a scientific research project.

Polar Axis System. – A mounting system for a telescope where one axis is mounted such that it is parallel to the axis of rotation of the earth. The advantage of this system as against an Altitude–Azimuth mounting is that only one axis need be driven at a slow , constant velocity to track the path of a star across the sky. This makes the control and drive systems very much simpler. A disadvantage is the increased mass of the mounting to obtain the necessary mechanical rigidity.

Slit. – The opening in the telscope dome through which the telscope can see the sky. The slit is closed by specially shaped doors called shutters.

Slow Motion Drive. – A computerized control system for moving the telescope at a slow , constant velocity to follow a star track across the sky. The system controls both axes of the telescope when the observer is aiming the telescope at a star.

Telescope Pointing. – A phrase refering the position of the telescope when it is being set to aim at a particular star.

Turret. – The telescope dome of the 1,9m telscope is referred to as the Turret due to it's shape – a cylinder with a flat roof , instead of the more usual dome shape.

10. Bibliography.

The TTL Data Book for Design Engineers.
Texas Instruments Corporation.

Peripheral Design Handbook.
Intel Corporation.

SABUS 6809 CPU with EPROM/RAM.
SAAO Internal Document by G.F.W. Woodhouse.

SABUS Counter/Timer.
SAAO Internal Document by G.F.W. Woodhouse.

The South African Microprocessor Bus Standard , Bus Specification.
Basic Electronics Document.

SABUS Dual Channel Serial Communication Card – Module 805/1.
Basic Electronics.

Maggs Industrial Micros SABUS Cards. General Purpose I/O Wire Wrap card 701/1
Basic Electronics.

SD2 & SD3 Stepping Motor Translator Provisional Data Sheet.
Digiplan (Pty.) Ltd.

OmegaSoft 6809 Pascal Language Handbook.
Certified Software Corporation.

6809 Asssembly Language Programming.
L.A. Leventhal , Osborne/McGraw–Hill.

11. Acknowledgements.

I wish to thank my collegues in the electronics section of S.A.A.O. , especially Mr. G. Woodhouse , with whom many helpful discussions were held.

Appendix A.
XYslide Specifications

1. Mechanical.

	Ideal	Achieved
Resolution	100 microns	10 microns
Limits of travel X axis	145 mm.	Circle of
Y axis	135 mm.	radius 140 mm.
Accuracy(repeatability)	200 microns	See note 1 below
nonlinearity(max)	300 microns	See note 2 below
Velocity mode accuracy	5 microns	5 microns
Time to travel 100mm	5 seconds	10 seconds

2. Functions.

2.1. User interaction : The user must be able to control the system from a standard computer terminal.

2.2. Autocentre : Probe to automatically find the centre position on power up. i.e. a position approximately at the centre of the telescope field of view.

2.3. Move to a position : System must be capable of moving to a position defined by coordinates entered in at the terminal.

2.4. Manual position control : The User must be able to control probe movement from the eyepiece position.

2.5. Velocity Mode : The probe must be able to move in a defined direction at a defined speed, suitable for tracking comets (i.e. speeds ranging from 0,1 mm/S to 10mm/S.)

2.6. Control of Acquisition Box : The system must have control of the position of the mirrors in the acquisition box.

Note 1 : Positional repeatability depends on the specification of the stepping motor and minimum flexure/backlash in the mechanical coupling between the motor and the leadscrew. The motor has a specified repeatability of 5% of a step position. As one step is equivalent to 5 microns linear movement, the uncertainty is 5% of 5 micron = 0.25 micron. The backlash in the coupling was measured at an average of 2.5 microns for repeated tests on both axes in the horizontal plane.

Note 2 : Nonlinearity is a function of the accuracy to which the precision–ground leadscrew is machined, plus step motor repeatability (0.25 micron) plus coupling backlash (2.5 micron). The leadscrew used has a maximum specified machining error of $^{+}$–10 microns over 300 mm. Thus the maximum linearity error is $^{+}$–12.75 microns.

Appendix B.

Automated X–Y Slides
User's Manual

By : D Carter July '86.

INDEX

1. Introduction.

The X–Y Slide system is a computer controlled two dimensional movement mechanism designed to facilitate the positioning of the telescope autoguider probe. The system block diagram is represented in Fig.1.

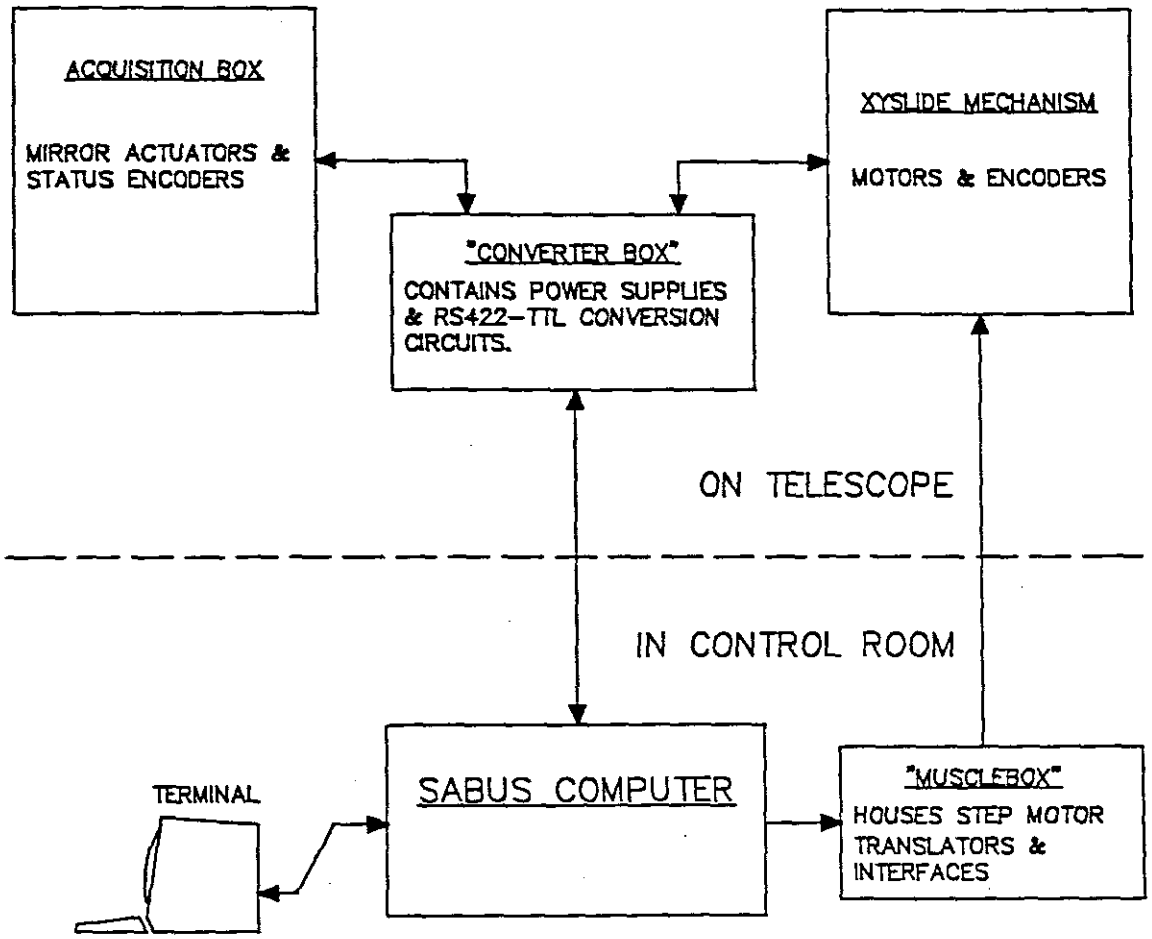


Fig. 1.

The block labelled 'CONVERTER BOX' in Fig. 1 is a small ELMASET card frame crate bolted on the side of the acquisition box. It contains some electronic circuitry , and incorporates a set of pushbutton switches for manual control of the X–Y mechanism position , as well as graticule illumination control for the autoguider eyepiece. (This latter will only work with the SAAO autoguider , not the Grubb–Parsons equipment.)

2. XYSlide Vital Statistics.

Area of movement	: Circle of radius 70mm
Movement method	: Stepping motors and precision ground leadscrew of 2mm pitch
Position encoding	: 10–turn potentiometer and slotted disk encoder with software counters.
Step resolution	: 5 microns
Positioning resolution	: 10 microns
Maximum speed	: 2200 steps/second, limited to 2000, or 10mm/second.
Velocity mode method	: Programmable frequency synthesizer and temperature–stabilized crystal oscillator.

3. Getting Started.

3.1. Switch on power to the musclebox first , then switch on the microcomputer. This should result in the following sequence :

A) The terminal should come up with :

```
*****************************************
        XYSLIDE CONTROL PROGRAM
          ver. 5.03    Mar. 86
*****************************************
```

B) The X–Y mechanism motors should operate, driving the autoguider probe to the centre of the field. A buzzer should then sound at the telescope for approximately 1 second.

C) The following should appear on the terminal :

```
XYSLIDE CENTRED
Setup complete
Type 'HELP' for command menu
#####
Enter Command :
```

The XYSlide is now ready to use.

3.2. Alternatively , the terminal bell will ring and the following message will appear :

SYSTEM HARDWARE ERROR DETECTED

and then one or other (or both) of the following messages

a) Check mains supply to Converter Box.

b) Check Musclebox CD10 I/F modules :

FAULT lights must be OFF , and

PH0 lights must be ON.

If not , switch Musclebox OFF then ON again.

Message (a) most probably means the mains supply to the converter box (labelled MANUAL CONTROL BOX in Fig. 1) has been switche off or disconnected. Message (b) implies that the musclebox is in an incorrect state – to rectify , switch it off , wait ten seconds , and switch on again. As soon as the fault state has been rectified , the program will continue with the initialize sequence as in para. 3.1.

3.3 WHAT HAPPENS IF ... it doesn't work. Either go to bed or if all else fails call the duty electronics technician.

4. Use/Capabilities of the system.

The equipment as configured at the moment has operations/commands that can be divided into four catagories :

4.1. XYslide Positioning : Move to a defined position, centering or re–initializing itself , moving in response to pushbuttons, list current position of probe.

Commands are : MOV, CENTR, LPOSN, SPEED

4.2. XYSlide Tracking – or Velocity mode : Moving in a defined direction at a defined speed.

Commands are : VMODE, VSTOP, VC, VHOLD, VSPEED

4.3. Acquisition box Control : Control of the position of the mirrors in the acquisition box , and the graticule illumination of the Rear viewing optics.

Commands are : GMVIEW, GMCEN, RVMIN, RVMOUT

4.4. Test Functions : These are a series of commands to facilitate fault finding the equipment. The commands do not appear in the 'HELP' menu, but are explained in the technical manual.

5. Explaination of Commands.

5.1. Use of the keyboard.

5.1.1 There are two methods to enter commands : Either type in the command in full – shortened forms are not accepted – or use the function keys as explained in para. 5.1.3. below.

5.1.2. Typing errors : the ESPRIT terminal unfortunately does not allow the use of the backspace key , but CTRL–H (CTRL key and H pressed simultaneously) does the correct backspace function to correct typing errors.

5.1.3. Function key use : Certain commands are available as single keystroke commands , (as well as being typed in full) , by using the ESPRIT III terminal function keys. To implement the command simply press the apropriate function key – it is not neccessary to press return. The commands are :

Command	Function key
CENTR	F1
MOV	F2
GMCEN	F3
GMVIEW	F4
RVMIN	F5
RVMOUT	F6
HELP	F11

NOTE : 1) This facility only works with the ESPRIT terminal.

5.2. XYSlide Positioning commands.

5.2.1. MOV (or F2) will prompt you for a destination position in X and Y. The positioning system is based on normal cartesian coordinates , as in Fig. 2.

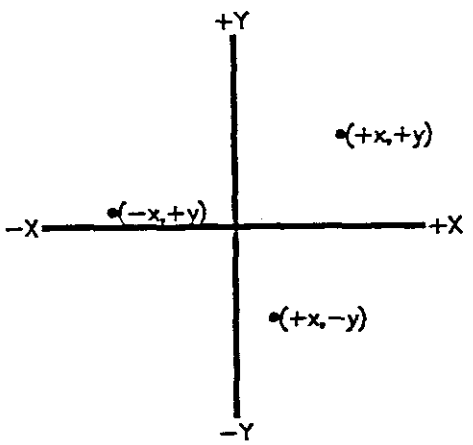


Fig. 2.

The units are hundreths of millimetres. The available area of movement is a circle of 70mm radius. If the position you defined is outside this circle, the terminal will come up with :

POSITION OUT OF RANGE!
Have another go!

the process is aborted and you have to start again.

5.2.2. CENTR (or F1) This will move to the hardware defined centre position and re–initialize the software position counters , useful if you think the XYslide has forgotten where it is !

5.2.3. LPOSN Will list the current X–Y position on the terminal in units of hundreths of mm.

5.2.4. SPEED Allows changing the speeds of moving for the MOV command and the pushbuttons. Default values are 2000Hz and 500Hz respectively, which are 10mm/sec and 2,5mm/sec. linear speed.

The pushbuttons at the telescope are 'live'at all times except if velocity mode is active. The position is checked continually , and if you attempt to move out of the valid area , the appropriate pushbutton dies and the buzzer sounds. To extricate yourself from this predicament , press the appropriate button to reverse direction.

5.3. XYSlide Tracking Commands.

5.3.1. VMODE Initiates the velocity mode. The terminal will prompt you to press RETURN to apply previously entered speeds, or press any other key to enter new speeds, in which case it prompts for speeds in X and Y – direction is indicated by + and – signs , default is +. Maximum speed allowed is 0.46mm/second siderial. Units are hundreths of mm. per second siderial.The terminal will prompt for any key to be pressed when you want the tracking to start. There is a crude position check done at regular intervals , and if the X–Y mechanism attempts to move outside the available area , it is automatically stopped. A message appears on the terminal as follows :

XYSLIDE IN INVALID POSITION !
use VSTOP command to reset it.

At this stage use of VSTOP will re–centre the X–Y mechanism to re–initilize the positioning system. (See explaination of VSTOP command in para. 5.3.4 below.) It is essential to use VSTOP at this point.

5.3.2. VC Allows you to change the velocity mode speeds currently used. The command prompts for new speeds and implements them after the second speed has been entered.

5.3.3. VHOLD Suspends the tracking function , but does not exit the velocity mode , and enables the pushbuttons for 'tweaking' the position of the autoguider probe. Prompts for any key to be pressed to continue tracking.
IMPORTANT : The position checking does not operate in this state , so it is possible to move into the limits of travel. BE CAREFUL!

5.3.4. VSTOP Stops velocity mode operation and automatically recentres the X–Y mechanism. (This is a requirement of the hardware , because as soon as VMODE is entered , the X–Y position is no longer updated in software.)

5.3.5. VSPEED Allows changing the calibration constant applied in the conversion of the user entered speeds to a number that is applied to the synthesizer to generate the required speed.

Due to errors introduced by rounding and resolution this constant might need to be 'tweaked'. Below is an example of how the constant was calculated , but first an explaination of how the velocity mode speed is obtained. A block diagram of the frequency synthesizer is shown in Fig. 3.

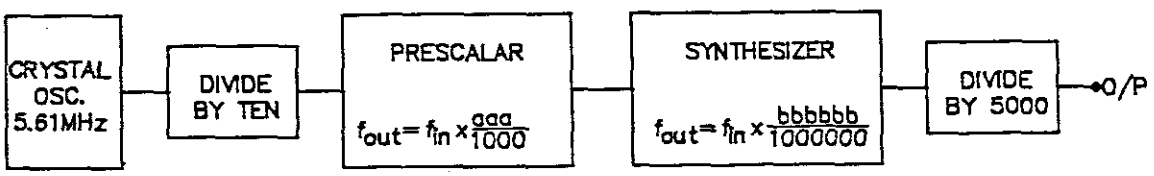


Fig. 3.

The three digit number 'aaa' is fixed in the hardware by switches. The six digit number 'bbbbbb' is set by user input at the keyboard to control the speed.

Constants : Scale : 200 steps = 1mm of movement.
6 arcsec = 1mm (74")
13.8 arcsec = 1mm (40")

Using the slowest specified speed of 1 arcsec per hour of time :

For 74" : 1 arcsec/Hr = 0.166mm/Hr.

$= 46.296 \times 10^{-6}$ mm/sec

$= 46.296 \times 10^{-6} \times 200$ steps/sec

$= 0.009258$ steps/sec

Synthesizer input frequency = 462962.5 Hz

Required synthesizer output frquency $= 0.009258 \times 5000$

$= 46.292$ Hz

Thus the synthesizer control number to obtain this frequency :

$N = 46.292/462962.5$

$= 0.000100$

This is the number applied to the synthesizer control input. To obtain this number from the keyboard input (which is in units of hundreths of mm/sec) , the VSPEED constant is applied :

VSPEED constant $= 0.0001/0.004629$

$= 0.021601$

5.4. Acquisition box commands

For these commands to function , the COMPUTER/MANUAL switch on the acquisition box front control panel must be set to COMPUTER.

5.4.1. GMCEN (or F3) Puts the offset guider mirror to the centre (out of beam) position.

5.4.2. GMVIEW (or F4) Puts the offset guider mirror into the view (in the beam) position.

5.4.3. RVMIN (or F5) Puts the Rear viewing mirror into the beam.

5.4.4. RVMOUT (or F6) Puts the rear viewing mirror clear of the beam.

5.4.5. GROFF switches the graticule illumination of the rear viewing optics ON.

5.4.6. GROFF switches the graticule illumination OFF.

NOTE : GRON and GROFF do not do anything at present if the LLTV camera is mounted , as there is no graticule to illuminate in the optics of the camera.

6. Command Set.

CENTR	Move to the initialize centre position.
FUNCSET	Program the Terminal function Keys.
GMCEN	Move the Offset Guider Mirror out of the Beam.
GMVIEW	Move the Offset Guider Mirror to the View position
HELP	List the Help Menu on the screen.
LPOSN	List the current X–Y coordinates of the probe on the screen.
MOV	Move the probe to the specified coordinates.
RVMIN	Move the Rear–viewing mirror into the View postion.
RVMOUT	Move the Rear–viewing mirror out of the Beam.
SPEED	Change the probe movement speeds.
VC	Change Velocity mode speeds.
VHOLD	Temporarily suspend velocity mode to change probe position.
VMODE	Initiate Velocity Mode.
VSPEED	Change The Constant used in velocity mode speed calculations.
VSTOP	Exit velocity mode.

Appendix C

XYSlide Software Manual

By : D. Carter Oct. '87

INDEX

1. Introduction.

The XYslide program is written in Omegasoft Pascal , compiled to run on a Motorola 6809 processor. The important feature of this implementation of Pascal is that the compiler produces fully ROMable code. It also provides easy interfacing to assembly language procedures. The program was developed as a stand–alone system , but has since been converted to form a module of the combined XYslide/Autoguider program.

2. Include Files.

The Omegasoft Pascal compiler provides for include files, which facilitates separating the source code into small files to make editing very much easier. The XYslide Module code has been split into many include files , each combining a logical grouping of procedures and functions.

3. Assembly Language Code.

A few procedures were written in assembler for reasons of speed or efficiency , and a fast interrupt routine for stepping the motors was also written in assembler because of execution time constraints.

The Xyslide uses the 6809 Fast interrupt line (FIRQ) to run the code that moves the stepping motors under 'normal' operation – i.e. Not Velocity mode. This code is assembled using the Omegasoft Relocatable Assembler , and linked to the compiled PASCAL code by the Omegasoft Linkage creator.

There is also a short assembly code program called VECTOR.SA which creates the restart vectors at the top of the memory map ($FFF8 – $FFFF). This code is assembled using the Motorola Asembler program RASM09 instead of the Omegasoft assembler as the latter will not produce absolute code.

4. Procedure for Program Creation.

a) Enter the source code using a text editor. (The Motorola text editor E <filnam> can be used , or the text can be entered using the Turbo PASCAL editor on a PC and then posted across to the Exorcisor on the RS232 link using the utility programs SND.PAS and REC.PAS).

b) Compile the source code with the Omegasoft PASCAL compiler , the command is – PC <filnam> , then assemble the code with the command – RA <filnam>

c) Enter the text for the Asssembler code modules and assemble them with the command – RA <filnam>.

d) Run the linkage creator program to link all XYslide assembled modules and create an object code file. Command – LC <filnam>.

The code can now be run in the EXORcisor for test purposes. To do this the SABUS crate with all the XYslide interfacing must be connected to the EXORcisor via the EXORcisor–to–SABUS interface. The restart and interrupt vectors at top of memory must also be set up – Use the EXORcisor Monitor EXBUG to do this.

e) If the unmodified CPU board is being used, use the Motorola utility PP509 to program a set of 2732 EPROMs. Aproximately 20 memory chips will be necessary. Transfer the code in 2K chunks into successive memories, with an offset of $800. i.e. program only the top 2K of each EPROM chip. When finished, insert the memory chips into the 32K memory board. program the restart vectors into a 2732 memory chip and insert it into the socket on the CPU board.

f) Alternatively, if the new (modified) CPU board is being used in the target system, do the following :

Post the debugged code to the PC compatible computer using the utility programs OBJSND and OBJREC. This step is necessary because the Motorola Prom programmer cannot handle 64K x 8 EPROM's. Now burn the code into a 27512 memory chip using the command EPROM on the PC–compatible. Also transfer the restart vectors into the top 16 locations. Insert the program code EPROM into socket on the SABUS 6809 CPU board and ensure that a 32K memory board with at least 6K of RAM is plugged into the bus, and test the system.

5. Source files for the XYslide.

The following is a list of all the files needed to create an XYslide Program. The version numbers are correct as at 24/11/87.

XYSLIDE.SA	v6.06	Main Program
XYVAR.SA	v1.09	Constants, types & Variable definitions
XYMOV.SA	v2.04	Move routines
XYCEN.SA	v1.05	Centre routines
UTIL1.SA	v1.05	various utility routines
VELOCE.SA	v1.02	Velocity mode routines
HANDSET.SA	v1.03	Pushbutton moving routines
VDUDRIVE.SA	v1.01	Terminal driver
AQBOX.SA	v2.05	Acquisition Box control routines
POTPOSN.SA	v1.00	Assembler code A–D Converter read routine
INTRPT.SA	v1.00	Assembler code to set interrupt vector
FIRQ1.SA	v1.06	Assembler code Fast interrupt routine
FIRQ2.SA	v1.12	Assembler code Fast interrupt routine
FIRQ3.SA	v1.00	Assembler code Fast interrupt routine
VECTOR.SA	v1.03	Assembler code restart vectors.

```
{**********************PROGRAM XYSLIDE*****************************}

{
 This is ver 6.06, the standalone ROMable version
 with restructured memory map.
   Start address $8000
   Data stack $10 - $4FF
   Systemstack $500 - $5FF
   FIRQ vector is $0. This has a jump instr.to start of FIRQ routine

*************************************************************************
*  ver 5     25/01/86    : restructured I/O addresses                    *
*  ver 5.01   4/02/86    : Esprit function key programming               *
*  ver 5.02  27/02/86    : Changes to velocity mode structure            *
*  ver 5.03  24/03/86    : VHOLD command added, changes to vmode for     *
*                          error checking. UTIL1 include file created    *
*  ver 5.04  12/08/86    : Changes to HANDSET to  remove ´1st time       *
*                          thru ´ bug. Consolidating Sendline statements *
*  ver 5.05  15/08/86    : Changes to fast interrupt routine             *
*                          to overcome step motor phase lag problem      *
*  ver 5.06  21/08/86    : Changes to VELOCE to tidy up VHOLD and add    *
*                          trap to TRACK for implementing old speeds     *
*                          also change to BUZZER for hardware mod        *
*  ver 5.07  18/12/86    : tidy up of sendline statements and code       *
*  ver 6.00  20/12/86    : Vmodeflag to VmodeActive                      *
*  ver 6.01  21/12/86    : Handsetflag to PBactive, delete Syserror      *
*  ver 6.02  21/12/86    : Delay calls changed                           *
*  ver 6.03  22/12/86    : Systemcheck more intelligent                  *
*  ver 6.04  28/01/87    : corrections to 6.03 , RVMINTOBEAM             *
*  ver 6.05  28/01/87    : added DISPLAY                                 *
*  ver 6.06  28/01/87    : corrections to DISPLAY                        *
*                                                                        *
*************************************************************************
}

PROGRAM XYSLIDE ;

{$IXYVAR.SA:1}          {GLOBAL VARIABLES INCLUDE FILE}

{$IVDUDRIVER.SA:1}

{$IXYMOV.SA:1}          {ROUTINES FOR MOVING IN X & Y}

{$IXYCEN.SA:1}          {ROUTINES FOR INITIALIZING POSITION}

{$IUTIL1.SA:1}          {VARIOUS UTILITIES}

{$IAQBOX.SA:1}           {ROUTINES FOR CONTROLLING AQU. BOX}

{$IHANDSET.SA:1}         {ROUTINES FOR MOVING VIA PUSHBUTTONS}

{$IVELOCE:1}             {ROUTINES FOR VELOCITY MODE}
```

```
{************************PROCEDURE DISPLAY*****************************}

Procedure Display ;
Begin
  Sendline(CONCAT(C,L,L,'Press any key to stop this display mode ',
                        'and wait until the prompt ')) ;
  Sendline(CONCAT(C,L,'appears before entering any other ',
                        'commands.',C,L)) ;
  Charflag  := false ;
  WHILE NOT Charflag DO
  Begin
    Centre ;
    GMView ;
    Delay(30000) ;
    Delay(30000) ;
    Mov(5143,-3789) ;
    Printpos ;
    Getchar(CH);
    IF Charflag THEN Exit ;
    Sendline(CONCAT(C,L,L,'Press any key to stop this display mode ',
                          'and wait until the prompt ')) ;
    Sendline(CONCAT(C,L,'appears before entering any other ',
                        'commands.',C,L)) ;
    RVMintoBeam ;
    GMCentre ;
    Delay(30000) ;
    Mov(-6999,0) ;
    Printpos ;
    Getchar(CH);
    IF Charflag THEN Exit ;
    Delay(30000);
    Mov(-4567,-4567) ;
    Printpos ;
    RVMoutBeam ;
    Delay(30000);
    Mov(0,6789);
    Printpos ;
    Delay(30000) ;
    Getchar(CH) ;
  End ;
  Charflag  := false ;
End ;


{************************PROCEDURE INITIALIZE************************}

PROCEDURE Init ;
Begin
  MskFir ;

  xsltch  := #$0 ;
  ysltch  := #$0 ;

  Switch       := false ;
  Initial      := false ;
  VmodeActive  := false ;
```

```
  MBerror    := false ;
  PotError   := false ;

  PBactive   := false ;
  Timer      := 0 ;
  Goingleft  := false ;
  Goingright := false ;
  GoingUp    := false ;
  GoingDown  := false ;
  Setimer    := false ;

  Con74    := 0.0216637 ;
  Slewspeed := $1F4 ;              (* Default 2000 Hz *)
  Hsetspeed := $7D0 ;              (* Default  500 Hz *)

(* This section initializes the terminal I/O port *)

  Vdusts  := #39 ;
  Vdusts  := #64 ;
  Vdusts  := #78 ;
  Vdusts  := #55 ;
  CH      := Vdudata ;
  SetUpFuncKeys ;

(* This section set's up the I/O Ports *)

  Counterbrdcw    := #$82 ;
  Countercontrol  := chr($55) ;          {All counters inhibit & reset}
  xcontrol  := chr($0) ;
  ycontrol  := chr($0) ;
  xmotor    := xcontrol ;
  ymotor    := ycontrol ;
  xsynthcw  := freqsetup ;
  ysynthcw  := freqsetup ;
  Iocw      := statsetup ;
  Aqbox     := chr($0) ;
  Firvec ;

  Cmndflag  := true ;
  Charflag  := false ;
  MesRx     := false ;
  Msgsent   := false ;
  Cmnd     := '0' ;
  MsgInBuf := '0' ;
  Ptr      := #0 ;
  MsgInBuf[0]   := Ptr ;
  Data     := '0' ;
End ;

{************************PROCEDURE PRINTPROMPT*********************}

Procedure Printprompt ;
Begin
  Sendline(CONCAT(C,L,L,'#####',C,L,L,'Enter Command : ')) ;
End ;
```

```
{**********************PROCEDURE CMNDINT**************************}

Procedure Cmndint ;
Begin
  IF Cmnd = 'MOV' THEN Move
  ELSE
   IF Cmnd = 'CENTR' THEN Centre
   ELSE
    IF Cmnd = 'LPOSN' THEN Printpos
    ELSE
     IF Cmnd = 'GMVIEW' THEN GMView
     ELSE
      IF Cmnd = 'GMCEN' THEN GMCentre
      ELSE
       IF Cmnd = 'RVMIN' THEN RVMintoBeam
       ELSE
        IF Cmnd = 'RVMOUT' THEN RVMoutBeam
        ELSE
         IF Cmnd = 'GRON' THEN Graton
         ELSE
          IF Cmnd = 'GROFF' THEN Gratoff
          ELSE
           IF Cmnd = 'BUZZ' THEN Buzzer
           ELSE
            IF Cmnd = 'MOTON' THEN Motorson
            ELSE
             IF Cmnd = 'MOTOFF' THEN Motorsoff
             ELSE
              IF Cmnd = 'HELP' THEN Help
              ELSE
               IF Cmnd = 'SPEED' THEN Changespeed
               ELSE
                IF Cmnd = 'TEST' THEN Test
                ELSE
                 IF Cmnd = 'VMODE' THEN Track
                 ELSE
                  IF Cmnd = 'VSPEED' THEN Velocityspeed
                  ELSE
                   IF Cmnd = 'VC' THEN VChange
                   ELSE
                    IF Cmnd = 'VSTOP' THEN Vstop
                    ELSE
                     IF Cmnd = 'FUNCSET' THEN SetUpFuncKeys
                     ELSE
                      IF Cmnd = 'VHOLD' THEN Vhold
                      ELSE
                       IF Cmnd = 'DISPLAY' THEN Display
  ELSE
    Sendline(CONCAT('Pardon? I don''t understand',
                    Bell,C,L,L));
  Cmndflag  := true ;
End ;
```

```
{************************MAIN PROGRAM*******************************}

BEGIN
  Init ;
  Sendline(CONCAT(C,L,L,L,L,L,L,L,L,L,L,L,L,L,L,L,L,L,L,L)) ;
  Sendline(CONCAT('*******************************************',C,L,L));
  Sendline(CONCAT('          XYSLIDE CONTROL PROGRAM',C,L,L,
                  '           Ver. 6.06   Jan. 87',C,L,L)) ;
  Sendline(CONCAT('*******************************************',C,L));
  Buzzer ;
  IF Syserror THEN
  Begin
    Sendline(CONCAT(C,L,Bell,L,'SYSTEM HARDWARE ERROR DETECTED .',C,L));
    IF Poterror THEN
      Sendline(CONCAT('Check mains supply to Converter Box.',C,L));
    IF MBerror THEN
    Begin
      Sendline(CONCAT('Check Musclebox CD10 I/F Modules : ',C,L,
                      '   FAULT Lights must be OFF , and',C,L));
      Sendline(CONCAT('   PH0 Lights must be ON .',C,L,'If not ,',
                      ' Switch Musclebox OFF then ON again.',C,L));
    End ;
    Sendline(CONCAT(L,'Then press RESET on XYslides Microcomputer.',C,L));
    WHILE Syserror DO ;
  End ;
  Centre ;
  Sendline(CONCAT(L,'Type "HELP" for Command Menu.',C,L)) ;
  Forever := true ;
  CH  :=Bell ;
  Sendchar(CH) ;
  Printprompt ;
  WHILE forever = true DO
   Begin
    Checkvdu ;
    IF MesRx = true THEN
      Begin
        MesRx  := false ;
        Cmndint ;
        Printprompt ;
      End ;
    IF VmodeActive THEN
      CheckPotPosn
    ELSE
      PushButtons ;
   End ;
End.
```

```
{*****************INCLUDE FILE XYVAR.SA************************************

   This compiles as an include file in PROGRAM XYSLIDE.
   It contains all constant and variable declarations.

*****************************************************************************
*  Version  last update                                                     *
*    1.00       ?                                                          *
*    1.01    18/12/86     remove Syserror                                  *
*    1.02    20/12/86     Vmodeflag to VmodeActive                         *
*    1.03    21/12/86     Delete Selstat                                   *
*    1.04    21/12/86     Delete GMpos , RVMpos, Handsetflag to PBactive   *
*    1.05    21/12/86     Delete Yref, Xref, YPh0, XPh0                    *
*    1.06    21/12/86     Xmeror, Ymeror : boolean                         *
*    1.07    21/12/86     Deleted Delaymax                                 *
*    1.08    22/12/86     Added MBerror, PotError                          *
*    1.09    27/01/87     corrections to 1.08                              *
*                                                                           *
*****************************************************************************

}

CONST
  Xcentre       = $01F0 ;
  Ycentre       = $0201 ;
  cw            = #$8 ;
  energ         = #$1 ;
  stp           = #$20 ;
  Stepenable    = #$80 ;
  Synthenable   = #$40 ;

  statsetup     = #$92 ;
  freqsetup     = #$80 ;

  Xrefmask      = #$01 ;
  XphOmask      = #$40 ;
  Yrefmask      = #$02 ;
  YphOmask      = #$80 ;

  Countmax      = 200 ;
  Calibrate     = $001F ;
  Bell          = #$7 ;
  Buzz          = #$8 ;
  Selstatmask    = #$2 ;

  GMposcontrol  = #$2 ;
  RVMposcontrol = #$1 ;
  RVMoutMask    = #$4 ;
  GMcenMask     = #$8 ;
  Gratcontrol   = #$4 ;

  Firqgate      = #$08 ;
  FirqReset     = #$04 ;
  FirqStatusMask = #$04 ;
  Irqgate       = #$20 ;
  Irqreset      = #$10 ;
  Irqstatusmask = #$10 ;
```

```
  Limit          = 49000000 ;
  ConRxRegMask   = #$1 ;
  ConTxRegMask   = #$2 ;
  TxRdy          = #$5 ;
  RxRdy      = #$2 ;
  C          = #13 ;
  L          = #10 ;


VAR
  xmotor      : byte at $E000 ;
  ymotor      : byte at $E008 ;
  xsynthcw    : byte at $E007 ;
  ysynthcw    : byte at $E00F ;
  XsynthLSB   : Byte at $E004 ;
  XsynthMidB  : byte at $E005 ;
  XsynthMSB   : byte at $E006 ;
  YsynthLSB   : byte at $E00C ;
  YsynthMidB  : byte at $E00D ;
  YsynthMSB   : byte at $E00E ;

  VduSts      : byte at $E011 ;
  VduData     : byte at $E010 ;

  ADC1        : byte at $E040 ;
  ADC2        : byte at $E041 ;

  Stat1       : byte at $E080 ;
  Stat2       : byte at $E081 ;
  Aqbox       : byte at $E082 ;
  Iocw        : byte at $E083 ;

  Counterstatus  : byte at $E031 ;
  Countercontrol : byte at $E032 ;
  CounterBrdcw   : byte at $E033 ;
  FIRQcntr       : byte at $E035 ;
  Irqcntr        : byte at $E036 ;
  Countercw      : byte at $E037 ;

  Xchkstp    : byte entry ;
  Ychkstp    : byte entry ;
  xcontrol   : byte entry ;
  ycontrol   : byte entry ;
  xsltch     : byte entry ;
  ysltch     : byte entry ;
  Initial    : boolean entry ;
  Switch     : boolean entry ;
  Ymeror     : Boolean entry ;
  Xmeror     : Boolean entry ;

  Xdest      : integer entry ;
  Ydest      : integer entry ;
  Xposn      : integer entry ;
  Yposn      : integer entry ;

  ysteps     : hex entry ;
  xsteps     : hex entry ;
```

```
Xdummy     : byte ;
Ydummy     : byte ;
Temp       : byte ;

GoingLeft  : boolean ;
GoingRight: boolean ;
GoingUp    : boolean ;
GoingDown  : boolean ;
Setimer    : boolean ;

PotError   : boolean ;
MBerror    : boolean ;

VmodeActive : boolean ;
Forever     : boolean ;
PBactive    : boolean ;
Msgsent     : boolean ;

Xdestdec   : real ;
Ydestdec   : real ;
Con74      : real ;

Xtemp      : integer ;
Xtempcnt   : integer ;
Ytemp      : integer ;
Ytempcnt   : integer ;
Speedint   : integer ;
Timer      : integer ;
Count, I   : integer ;
N, Dummy   : integer ;

Xstpcnt    : hex ;
Ystpcnt    : hex ;
Speed      : hex ;
Slewspeed  : hex ;
Hsetspeed  : hex ;
Xpos, Ypos  : hex ;

Test       : char ;
Charflag   : boolean ;

Cmnd       : string[80] ;
Data       : string[80] ;
Line       : string[80] ;
MsgInBuf   : string[80] ;
CH         : char ;
Ptr        : char ;
MesRx      : boolean ;
Cmndflag   : boolean ;
```

```
{*****************INCLUDE FILE XYMOV.SA**********************************

    This code compiles as an include file in PROGRAM XYSLIDE.
    It contains routines for moving to specified destination positions
    in X and Y.

*****************************************************************************
*  Version  Last Update                                                     *
*    1.00       ?                                                          *
*    1.01    19/12/86       Restructure of Sendline calls                  *
*    1.02    20/12/86       Put diagnostic traps in abort mov code         *
*    1.03    20/12/86       Vmodeflag to VmodeActive                       *
*    1.04    21/12/86       MskIrq to MskFir , ClrIrq to ClrFir            *
*    2.00    21/12/86       Xmeror, Ymeror : Boolean , Restructure         *
*                           Calcsteps IF THEN ELSE                         *
*    2.01    21/12/86       Changed Delay procedure call                   *
*    2.02    27/01/87       corrections to 2.01                            *
*    2.03    28/01/87       added MOV(X,Y)                                 *
*    2.04    29/01/87       changed DELAY to be longer                     *
*                                                                           *
*****************************************************************************

}


{************************PROCEDURE DELAY*****************************}
PROCEDURE delay(delaymax : integer) ;
VAR
  N  : hex ;

BEGIN
   FOR I := 1 TO delaymax DO
     N  := N + $1 ;
END ;

{************************PROCEDURE MSKIRQ****************************}
{Assembly routine to mask out IRQ}

PROCEDURE MskFir ; external ;

{************************PROCEDURE CLRIRQ****************************}
{Assembly routine to enable IRQ}

PROCEDURE ClrFir ; external ;

{************************PROCEDURE MOTORSOFF*************************}

PROCEDURE MotorsOff ;
Begin
  Delay(1000) ;
  xcontrol := xcontrol AND NOT energ ;
  ycontrol := ycontrol AND NOT energ ;
  xmotor := xcontrol ;
  ymotor := ycontrol ;
End ;
```

```
{**********************PROCEDURE MOTORSON**************************}

PROCEDURE Motorson ;
Begin
  xcontrol := xcontrol OR energ ;
  ycontrol := ycontrol OR energ ;
  xmotor := xcontrol ;
  ymotor := ycontrol ;
  Delay(1000) ;
End;

{**********************PROCEDURE SETSPEED**************************}

PROCEDURE Setspeed (Speed : hex) ;
Begin
  Countercontrol := Countercontrol AND NOT Firqgate ;
  Countercw := chr($74) ;
  Firqcntr := chr(Speed AND $FF) ;
  Speed := Speed >> $8 ;
  Firqcntr := chr(Speed AND $FF) ;
  Countercontrol := Countercontrol OR Firqgate ;
End ;
```

```
{************************PROCEDURE SLEW*****************************}

PROCEDURE Slew ;

PROCEDURE Initialslew ;
Begin
  WHILE Xsteps <> $0 DO
    ClrFir ;
  WHILE Ysteps <> $0 DO
    ClrFir ;
  MskFir ;
  Xmeror := false ;
  Ymeror := false ;
End ;

Begin
  Motorson ;
  Speed := Slewspeed ;
  Setspeed (Speed) ;
  Countercontrol := Countercontrol AND NOT Firqreset ;
  Xmeror := false ;
  Ymeror := false ;
  IF Initial THEN
    Initialslew              {No checks on motors in initialize}
  ELSE
    Begin
      Temp := Ycontrol AND StepEnable ;
      WHILE Temp <> chr(0) DO
      Begin
        Temp := Ycontrol AND StepEnable ;
        ClrFir ;
        IF Xmeror THEN EXIT ;
        IF Ymeror THEN EXIT ;
      End ;
      Temp := Xcontrol AND stepenable ;
      WHILE Temp <> chr(0) DO
      Begin
        Temp := Xcontrol AND stepenable ;
        ClrFir ;
        IF Xmeror THEN EXIT ;
        IF Ymeror THEN EXIT ;
      End ;
      MskFir ;
    End ;
  IF Xmeror THEN
    Sendline(CONCAT(Bell,'XMOTOR OUT OF SYNCH. MOVE ABORTED!',C,L,
                    'Use CENTR command to re-initialize.',C,L)) ;
  IF Ymeror THEN
    Sendline(CONCAT(Bell,'YMOTOR OUT OF SYNCH. MOVE ABORTED!',C,L,
                    'Use CENTR command to re-initialize.',C,L));
  Motorsoff ;
End ;
```

```
{*********************PROCEDURE CALCSTEPS*************************}
{Calculates no. of steps and direction to move to destination}

PROCEDURE Calcsteps ;

VAR
  Xtemp       : integer ;
  Xtempcnt    : integer ;
  Xstpcnt     : hex ;
  Ytemp       : integer ;
  Ytempcnt    : integer ;
  Ystpcnt     : hex ;

BEGIN
  Xtemp := Xposn MOD 40 ;
  Ytemp := Yposn MOD 40 ;
  IF Xposn = Xdest THEN
  Begin
    Xcontrol := Xcontrol AND NOT stepenable ;
    Xtempcnt := 0 ;
  End
  ELSE
    IF Xdest < Xposn THEN
    Begin
      Xcontrol := Xcontrol OR cw ;
      Xcontrol := Xcontrol OR stepenable ;
      IF Xtemp < 0 THEN
        Xtempcnt := 40 + Xtemp ;
      IF Xtemp > 0 THEN
        Xtempcnt := Xtemp ;
      IF Xtemp = 0 THEN
        Xtempcnt := 40 ;
    End
    ELSE
    Begin
      Xcontrol := Xcontrol AND NOT cw ;
      Xcontrol := Xcontrol OR stepenable ;
      IF Xtemp < 0 THEN
        Xtempcnt := ABS(Xtemp) ;
      IF Xtemp > 0 THEN
        Xtempcnt := 40 - Xtemp ;
      IF Xtemp = 0 THEN
        Xtempcnt := 40 ;
    End ;
  IF Yposn = Ydest THEN
  Begin
    Ycontrol := Ycontrol AND NOT stepenable ;
    Ytempcnt := 0 ;
  End
  ELSE
    IF Ydest < Yposn THEN
    Begin
      Ycontrol := Ycontrol AND NOT cw ;
      Ycontrol := Ycontrol OR stepenable ;
      IF Ytemp < 0 THEN
        Ytempcnt := 40 + Ytemp ;
      IF Ytemp > 0 THEN
        Ytempcnt := Ytemp ;
      IF Ytemp = 0 THEN
```

```
        Ytempcnt := 40 ;
    End
    ELSE
    Begin
      Ycontrol := Ycontrol OR cw ;
      Ycontrol := Ycontrol OR stepenable ;
      IF Ytemp < 0 THEN
        Ytempcnt := ABS(Ytemp) ;
      IF Ytemp > 0 THEN
        Ytempcnt := 40 - Ytemp ;
      IF Ytemp = 0 THEN
        Ytempcnt := 40 ;
    End ;
  Ystpcnt := HEX(Ytempcnt) ;
  Ychkstp := CHR(Ystpcnt) ;
  Xstpcnt := HEX(Xtempcnt) ;
  Xchkstp := CHR(Xstpcnt) ;
END ;


{**********************PROCEDURE CHANGESPEED**********************}

PROCEDURE Changespeed ;

VAR
 Decspeed  : real ;

Begin
    Sendline(CONCAT(C,L,L,'Enter new slew speed in Herz(max. 2000)'));
    Cmndflag  := false ;
    Decspeed  := 3000 ;
    WHILE Decspeed > 2001 DO
    Begin
      Sendline(CONCAT(C,L,L,'Enter Slew Speed : ')) ;
      WHILE MesRx = false DO
        CheckVdu ;
      MesRx  := false ;
      Decspeed  := INTEGER(Data) ;
    End ;
    Decspeed := 1000000 / Decspeed ;
    Slewspeed := HEX(Decspeed) ;
    Decspeed  := 3000 ;
    Sendline(CONCAT(L,'Enter new pushbutton speed in herz. (max. 1000)'));
    WHILE Decspeed > 1001 DO
    Begin
      Sendline(CONCAT(C,L,L,'Enter Pushbutton Speed : ')) ;
      WHILE MesRx = false DO
        Checkvdu ;
      MesRx  := false ;
      Decspeed := INTEGER(Data) ;
    End ;
    Decspeed := 1000000 / Decspeed ;
    Hsetspeed := HEX(Decspeed) ;
    Cmndflag  := true ;
    CH  := L ;
    Sendchar(CH) ;
End ;
```

```
{**********************PROCEDURE MOVE*********************************}

PROCEDURE MOVE ;

CONST
  Limit  = 49000000 ;

VAR
  Stepsize  : real ;
  Xmm, Ymm : real ;
  Checkpos : real ;
  Xtemp  : real ;
  Ytemp   : real ;
  Xchk, Ychk : hex ;
  Xdestdec   : real ;
  Ydestdec   : real ;


BEGIN
  IF VmodeActive THEN
    Sendline(CONCAT(C,L,'Cannot move - Velocity mode active.',Bell,C,L))
  ELSE
    Begin
      Sendline(CONCAT('Enter destination position in hundreths of ',C,L)) ;
      Sendline(CONCAT('millimeters from centre',C,L,
                      'i.e.  +3000 and -5347 will move to a position '));
      Sendline(CONCAT('30mm to the right,',C,L,'and 53.47mm below the'));
      Sendline(CONCAT(' centre of the XYslide movement range.',C,L,L,
                      'Enter Destination X : '));
      Cmndflag  := false ;
      WHILE MesRx = false DO
        CheckVdu ;
      MesRx  := false ;
      Xmm    := REAL(Data) ;
      Sendline(CONCAT(L,'Enter Destination Y : '));
      While MesRx = false DO
        Checkvdu ;
      MesRx  := false ;
      Ymm    := REAL(Data) ;
      CH     := L ;
      Sendchar(CH) ;
      Stepsize := 5 / 10 ;
      Xtemp := Xmm ;
      Ytemp := Ymm ;
      Checkpos := SQR(Xtemp) + SQR(Ytemp) ;
      IF Checkpos > Limit THEN
        Sendline(CONCAT(C,L,Bell,'POSITION OUT OF RANGE !',C,L,
                                 '    Have another go !',C,L))
      ELSE
        Begin
          Xdestdec := Xmm / Stepsize ;
          Ydestdec := Ymm / stepsize ;
          Xdest := INTEGER(Xdestdec) ;
          Ydest := INTEGER(Ydestdec) ;
          Calcsteps ;
          Slew ;
        End ;
    End ;
END ;
```

```
{**************************PROCEDURE MOVE(X,Y)**********************}

Procedure Mov(X,Y : integer) ;

BEGIN
  Xdest  := X * 2 ;
  Ydest  := Y * 2 ;
  Calcsteps ;
  Slew ;
End ;

{*******************************************************************}
```

```
{*****************INCLUDE FILE XYCEN.SA***********************************

   This code compiles as an include file in PROGRAM XYSLIDE.
   It contains routines for driving in X & Y to the hardware
   defined centre position.

*************************************************************************
*  version  last update                                                 *
*    1.00        ?                                                     *
*    1.01     19/12/86      Restructure sendline calls                 *
*    1.02     21/12/86      PhOY to Function YPHO, Refy to Function Yref  *
*                           PhOX to Function XPHO, Refx to Function Xref,  *
*                           Resructure X & Y-Finecentre, add error trap.   *
*    1.03     21/12/86      Changed Delay calls                        *
*    1.04     27/01/87      corrections to 1.03                        *
*    1.05     28/01/87      corrections to 1.04                        *
*                                                                       *
*************************************************************************



}



{************************PROCEDURE XCOORD**************************}
{   Assembly language subroutine to get and right justify 10 bit
number from A-D Convertor}

PROCEDURE Xcoord (VAR Pos : hex) ; external ;

{************************PROCEDURE YCOORD**************************}
{    Assembly language routine as above xcoord}

PROCEDURE Ycoord (VAR Pos : hex) ; external ;

{************************PROCEDURE CNT*****************************}

PROCEDURE Cnt ;
Begin
  Count := Count + 1 ;
End;

{************************PROCEDURE YSTEP***************************}

PROCEDURE ystep ;
BEGIN
  ycontrol := ycontrol OR stp ;
  ymotor := ycontrol ;
  ycontrol := ycontrol AND NOT stp ;
  ymotor := ycontrol ;
  delay(4);
  Cnt ;
END ;
```

```
{************************PROCEDURE XSTEP*************************}

PROCEDURE xstep ;
BEGIN
  xcontrol := xcontrol OR stp ;
  xmotor := xcontrol ;
  xcontrol := xcontrol AND NOT stp ;
  xmotor := xcontrol ;
  delay(4) ;
  Cnt ;
END ;

{************************FUNCTION YPH0*****************************}

Function YPh0 : boolean ;
Begin
  IF Stat2 AND Yph0mask  = Yph0mask THEN
    Yph0 := true
   ELSE
    Yph0 := false ;
End ;

{************************FUNCTION XPH0*****************************}

Function XPh0 : boolean ;
Begin
  IF Stat2 AND Xph0mask = Xph0mask THEN
    Xph0 := true
   ELSE
    Xph0 := false ;
End ;


{************************FUNCTION YREF*****************************}

Function Yref : boolean ;
Begin
  IF Stat2 AND Yrefmask = Yrefmask THEN
    Yref := true
   ELSE
    Yref := false ;
End ;

{************************FUNCTION XREF*****************************}

Function Xref : Boolean ;
Begin
  IF Stat2 AND Xrefmask = Xrefmask THEN
    Xref := true
   ELSE
    Xref := false ;
End ;
```

```
{************************PROCEDURE FINDYPH0************************}

PROCEDURE FindYph0 ;
BEGIN
  WHILE NOT YPh0 DO ystep ;
END ;

{************************PROCEDURE FINDXPH0************************}

PROCEDURE FindXph0 ;
BEGIN
  WHILE NOT XPh0 DO xstep ;
END ;

{************************PROCEDURE YFINECENTRE*********************}

PROCEDURE YfineCentre ;
Begin
  ycontrol := ycontrol OR cw ;
  IF NOT YPh0 THEN FindYph0 ;
  Count := 0 ;
  IF NOT Yref THEN
  Begin
    WHILE NOT Yref DO
    begin
      FOR N := 1 to 8 DO ystep ;
      IF Count = Countmax THEN exit ;
    end ;
    IF NOT Yref THEN
    Begin
      ycontrol := ycontrol AND NOT cw ;
      FOR N := 1 TO 200 DO ystep ;
      Count := 0 ;
      WHILE NOT Yref DO
      begin
        FOR N := 1 TO 8 DO ystep ;
        IF Count >= Countmax THEN exit ;
      end ;
    End ;
  End;
END ;

{************************PROCEDURE XFINECENTRE*********************}

PROCEDURE XfineCentre ;
Begin
  xcontrol := xcontrol OR cw ;
  IF NOT Xph0 THEN FindXph0 ;
  Count := 0 ;
  IF NOT Xref THEN
  Begin
    WHILE NOT Xref DO
    begin
      FOR N := 1 to 8 DO xstep ;
      IF Count = Countmax THEN exit ;
    end ;
    IF NOT Xref THEN
    Begin
      xcontrol := xcontrol AND NOT cw ;
```

```
      FOR N := 1 TO 200 DO xstep ;
      Count := 0 ;
      WHILE Xref = false DO
      begin
        FOR N := 1 TO 8 DO xstep ;
        IF Count >= Countmax THEN exit ;
      end ;
    End ;
  End;
END ;

{**********************PROCEDURE CALCMOVE**************************}

PROCEDURE CalcMove ;

VAR
  Distance    : hex ;
  Ypospot     : hex ;
  Xpospot     : hex ;
BEGIN
  Xcoord (Xpospot) ;
  Ycoord (Ypospot) ;
  IF Xpospot = Xcentre THEN
  Begin
    Xsteps := $0 ;
    Xcontrol := Xcontrol AND NOT stepenable ;
  End
  ELSE
    IF Xpospot > Xcentre THEN
    Begin
      Distance := Xpospot - Xcentre ;
      Xsteps := Distance * Calibrate ;
      xcontrol := xcontrol OR cw ;
      xcontrol := xcontrol OR stepenable ;
    End
    ELSE
    Begin
      Distance := Xcentre - Xpospot ;
      Xsteps := Distance * Calibrate ;
      xcontrol := xcontrol AND NOT cw ;
      xcontrol := xcontrol OR Stepenable ;
    End ;
  IF Ypospot = Ycentre THEN
  Begin
    Ysteps := $0 ;
    Ycontrol := Ycontrol AND NOT stepenable ;
  End
  ELSE
    IF Ypospot > Ycentre THEN
    Begin
      Distance := Ypospot - Ycentre ;
      Ysteps := Distance * Calibrate ;
      ycontrol := ycontrol AND NOT cw ;
      ycontrol := ycontrol OR Stepenable ;
    End
    ELSE
    Begin
      DIstance := Ycentre - Ypospot ;
      Ysteps := Distance * Calibrate ;
```

```
      Ycontrol := ycontrol OR cw ;
      Ycontrol := ycontrol OR Stepenable ;
    End ;
END ;

{**********************PROCEDURE CENTRE*****************************}

PROCEDURE Centre ;
Begin
  Calcmove ;
  Initial := true ;
  Slew ;
  Initial := false ;
  Motorson ;
  Xfinecentre ;
  Yfinecentre ;
  IF Stat2 AND #$F = #$F THEN
  Begin
    Sendline(CONCAT(C,L,'X-Y SLIDE CENTRED .',Bell,C,L,L,
                    'Initialization Complete. ',C,L));
    Xposn := 0 ;
    Yposn := 0 ;
    Motorsoff ;
  End
  ELSE
  Begin
    Sendline(CONCAT(C,L,'ERROR - CANNOT FIND CENTRE.',C,L,Bell,
                    'Press RESET on XYSLIDE Microcomputer and if'));
    Sendline(CONCAT(' that does not work,',C,L,
                  ' call a technician.',Bell,C,L));
  End;
End ;
```

```
{*****************INCLUDE FILE UTIL1.SA***********************************

   This module compiles as an include file in PROGRAM XYSLIDE.
   It contains some general utility procedures and functions

*************************************************************************
*  Version  last update                                                 *
*    1.00        ?                                                      *
*    1.01    18/12/86      Restructure of code and sendline statements  *
*                          Systemcheck now a function Syserror          *
*    1.02    19/12/86      Changes to TEST                              *
*    1.03    20/12/86      More changes to TEST                         *
*    1.04    22/12/86      Changes to Syserror                          *
*    1.05    20/01/87      More changes to Syserror                     *
*                                                                       *
*************************************************************************



}



{************************PROCEDURE FIRVEC**************************}
{Assembly language routine to set up interupt vector}

PROCEDURE Firvec ; external ;

{***********************PROCEDURE PRINTPOSITION*********************}

PROCEDURE Printpos ;

VAR
  Xtemp  : integer ;
  Ytemp  : integer ;
  X   : string[6] ;
  Y   : string[6] ;

Begin
  Xtemp := Xposn div 2 ;
  Ytemp := Yposn div 2 ;
  X  := STRING(Xtemp) ;
  Y  := STRING(Ytemp) ;
  Sendline(CONCAT(C,L,'Current position is :  X =  ',X,
                      '    Y =  ',Y,C,L)) ;
End ;

{*************************PROCEDURE HELP****************************}

PROCEDURE Help ;
Begin
  Sendline(CONCAT(C,L,'Commands available are :',C,L)) ;
  Sendline(CONCAT('   MOV     - Move in X & Y to specified destination.',
                  C,L)) ;
  Sendline(CONCAT('   LPOSN   - List current position co-ordinates.',C,L));
  Sendline(CONCAT('   CENTR   - Move to centre of field of view.',C,L));
  Sendline(CONCAT('   VMODE   - Velocity mode.',C,L,
```

```
                    '   VC      - Change velocity mode speed.',C,L));
  Sendline(CONCAT('   VHOLD   - Suspend vmode tracking and enable ',
                    'pushbutton operation.',C,L)) ;
  Sendline(CONCAT('   VSTOP   - Stop velocity mode of operation.',C,L)) ;
  Sendline(CONCAT('   VSPEED  - Examine / Change constant applied for',C,L));
  Sendline(CONCAT('             setting up velocity mode speeds.',C,L));
  Sendline(CONCAT('   SPEED   - Change motor speeds.',C,L));
  Sendline(CONCAT('   GMCEN   - Offset guide mirror to centre position.',C,L));
  Sendline(CONCAT('   GMVIEW  - Offset guide mirror to view position.',C,L));
  Sendline(CONCAT('   RVMIN   - Rear viewing mirror to view position.',C,L));
  Sendline(CONCAT('   RVMOUT  - Rear viewing mirror out of beam.',C,L));
  Sendline(CONCAT('   GRON    - Rear view optics graticule illum. on.',C,L));
  Sendline(CONCAT('   GROFF   - Rear viewing graticule illum. Off.',C,L));
  Sendline(CONCAT('   FUNCSET - Set up Esprit terminal function keys.',C,L));
  Sendline(CONCAT('   HELP    - List this command menu.',C,L)) ;
  Sendline(CONCAT(L,'    Press any key for more.....')) ;
  Charflag   := false ;
  WHILE Charflag = false DO
    Getchar(CH) ;
  Charflag  := false ;
  Sendline(CONCAT(C,L,L,L,L,'   GUIDE   - Start Autoguiding.',C,L)) ;
  Sendline(CONCAT('   AQUIRE  - Search for and lock on to a star.'));
  Sendline(CONCAT(C,L,L,'    Function Keys :',C,L,
                    '    ***************',C,L)) ;
  Sendline(CONCAT('            Certain XYSLIDE/AUTOGUIDER commands',C,L));
  Sendline(CONCAT('    can be initiated from the function keys of',C,L));
  Sendline(CONCAT('    the Esprit III terminal. These are :',C,L)) ;
  Sendline(CONCAT('       CENTR  -  F1',C,L,
                  '       MOV    -  F2',C,L,
                  '       GMCEN  -  F3',C,L));
  Sendline(CONCAT('       GMVIEW -  F4',C,L,
                  '       RVMIN  -  F5',C,L,
                  '       RVMOUT -  F6',C,L));
  Sendline(CONCAT('       HELP   -  F11',C,L));
  Sendline(CONCAT(L,'  NOTE - Do NOT use the Function keys while ',
                  'VMODE is active',C,L,'       as they ')) ;
  Sendline(CONCAT('will not work correctly.',C,L,L,L,
                  '    Press any key to continue.....')) ;
  Charflag   := false ;
  WHILE Charflag = false DO
    Getchar(CH) ;
  Charflag  := false ;
End ;

(************************PROCEDURE TEST*****************************)

PROCEDURE Test ;


VAR
  Choice      : integer ;
  Temp        : byte ;
  Temp1       : byte ;
  N           : integer ;
```

```
PRocedure LISTSTATUS(Var Testio : byte; N : integer);
Begin
  Temp := Testio ;
  Sendline(CONCAT(C,L,'BIT7    6     5     4     3     2     1   BIT0',C,L));
  Sendline(CONCAT(C,L)) ;
  IF N = 1 THEN
    Sendline(CONCAT('YDWN   YUP  XRGT  XLFT  GMPN  RVMP  SLCT  AGMR',C,L))
  ELSE
    Sendline(CONCAT('YPH0  XPH0  YFLT  XFLT  YSLT  XSLT  YREF  XREF',C,L));
  FOR N := 1 TO 8 DO
    Begin
      Temp1 := Temp AND #$80 ;
      IF Temp1 = #$80 THEN
        Sendline(CONCAT('  1   '))
       ELSE
        Sendline(CONCAT('  0   ')) ;
      TEmp := Temp << #$1 ;
    End ;
  Sendline(CONCAT(C,L)) ;
End ;

Procedure ListPotCoord ;

VAR
  Posn     : hex ;
  Position   : string[6] ;

Begin
  Xcoord(Posn) ;
  Position  := STRING(Posn) ;
  Sendline(CONCAT(C,L,'X Pot. value is : ',Position,C,L)) ;
  Ycoord(Posn) ;
  Position  := STRING(Posn) ;
  Sendline(CONCAT(C,L,'Y Pot. value is : ',Position,C,L)) ;
End ;

BEGIN
  Sendline(CONCAT('The following options are available',C,L,
                  '    1 .  Stat1 ',C,L,
                  '    2 .  Stat2 ',C,L)) ;
  Sendline(CONCAT('    3 .  Potentiometers ',C,L,
                  '    4 .  Exit this routine ',C,L,
                  'Enter code of choice : '));
  Data  := '000' ;
  Cmndflag  := false ;
  Charflag  := false ;
  Choice  := 0 ;
  WHILE Choice <> 4 DO
    Begin
      WHILE MesRx = false DO
        CheckVdu ;
      MesRx   := false ;
      Choice := INTEGER(Data) ;
      CASE Choice OF
         1  : Liststatus(Stat1,1);
         2  : Liststatus(Stat2,2);
         3  : ListPotCoord  ;
        End ;
```

```
      IF Choice <> 4 THEN
        Sendline(CONCAT(C,L,'Enter code of choice : ')) ;
    End ;
  Cmndflag  := true ;
  MesRx  := false ;
End ;

{**********************FUNCTION SYSERROR*****************************}

Function Syserror : boolean ;

Var
  Xpot       : hex ;
  Ypot       : hex ;
  Dummy      : byte ;

Begin
  Motorson ;
  Xcoord(Xpot) ;
  Ycoord(Ypot) ;
  Dummy  := Stat2 AND #$F0 ;
  IF Xpot < $20 THEN
    Poterror := true
  ELSE
    IF Ypot < $20 THEN
      Poterror := true ;
  IF Dummy <> #$C0 THEN
    MBerror  := true ;
  IF Poterror THEN
    Syserror  := true
  ELSE
    IF MBerror THEN
      Syserror  := true
    ELSE
      Syserror := false ;
End ;
```

```
{*****************INCLUDE FILE VELOCE.SA**********************************

   This code compiles as an include file in PROGRAM XYSLIDE.
   It contains routines for moving in X & Y in velocity mode,
   i.e. speed controlled by the frequency synthesizers.

*************************************************************************
*  version  last update                                                 *
*    1.00       ?                                                      *
*    1.01    20/12/86      Vmodeflag to VmodeActive and restructure of *
*                          sendline calls                              *
*    1.02    28/01/87      corrections to 1.01                         *
*                                                                       *
*************************************************************************


}




(*********************PROCEDURE VELOCITYSPEED************************)

Procedure Velocityspeed ;

VAR
  Dummy   : string[20] ;

Begin
  Dummy  := STRING(Con74) ;
  Sendline(CONCAT('Current constant is : ',Dummy,C,L,
                  'Enter new Velocity mode constant',C,L));
  Sendline(CONCAT('  (maximum allowed is  0.09) ',C,L,
                  'Enter constant : '));
  Cmndflag := false ;
  WHILE MesRx = false DO
    CheckVdu ;
  MesRx  := false ;
  Con74  := REAL(Data) ;
  Sendchar(L) ;
End ;

{*********************PROCEDURE STOPSYNTH*****************************}

Procedure StopSynth ;
Begin
  Xcontrol := Xcontrol AND NOT Synthenable ;
  Ycontrol := Ycontrol AND NOT Synthenable ;
  xmotor    := xcontrol ;
  Ymotor    := ycontrol ;
End ;
```

```
{*************************PROCEDURE CHECKPOTPOSN*****************}

Procedure CheckPotPosn ;

VAR
  Xpospot    : hex ;
  YposPot    : hex ;
  Xdistance  : hex ;
  Ydistance  : hex ;
  Xdistdec   : longinteger ;
  Ydistdec   : longinteger ;
  Radius     : real ;
  Dummy      : string[15] ;
  Xpos       : string[10] ;
  Ypos       : string[10] ;

Begin
  Xcoord(XposPot) ;
  Ycoord(YposPot) ;
  IF Xpospot = $200 THEN
    Xdistance  := $1
  ELSE
    IF Xpospot < $200 THEN
      Xdistance := $200 - Xpospot
    ELSE
      Xdistance := Xpospot - $200 ;
  IF Ypospot = $200 THEN
    Ydistance  := $1
  ELSE
    IF Ypospot < $200 THEN
      Ydistance := $200 - Ypospot
    ELSE
      Ydistance := Ypospot - $200 ;
  Xdistdec  := LONGINTEGER(Xdistance) ;
  Ydistdec  := LONGINTEGER(Ydistance) ;
  Radius    := SQRT(SQR(Xdistdec) + SQR(Ydistdec)) ;
  IF Radius > 445 THEN
  Begin
    Stopsynth ;
    IF Msgsent = false THEN
    Begin
      Buzzer ;
      Xpos  := STRING(Xpospot) ;
      Ypos  := STRING(Ypospot) ;
      Sendline(CONCAT(C,L,L,Bell,'XYSLIDE IN INVALID POSITION')) ;
      Sendline(CONCAT(C,L,L,'Pot. Readings : X ',Xpos,',  Y ',Ypos)) ;
      Sendline(CONCAT(C,L,L,'    Use VSTOP command to reset it.',C,L,L));
      Msgsent := True ;
    End ;
  End ;
End ;
```

```
{*************************PROCEDURE VSTOP*************************}

Procedure Vstop ;
Begin
  IF VmodeActive THEN
  Begin
    StopSynth ;
    VmodeActive  := false ;
    Msgsent   := false ;
    Centre ;
  End
  ELSE
    Sendline(CONCAT(C,L,'Velocity mode not active.',C,L));
End ;

{***************************PROCEDURE SETUPSYNTH*******************}

Procedure Setupsynth ;

VAR
  Dummy     : real ;
  Xspeed    : real ;
  Yspeed    : real ;
  Xspeedint : longinteger ;
  Yspeedint : longinteger ;
  Xspeedstr : string[10] ;
  Yspeedstr : string[10] ;
  Outloop   : boolean ;
  Byte1     : byte ;
  Byte2     : byte ;
  Byte3     : byte ;
  Error     : boolean ;


(*-------------------------PROCEDURE PACKBCD-------------------------*)

Procedure PackBCD(Var Str : string) ;

VAR
  CH1       : byte ;
  CH2       : byte ;

(* -------------------PROCEDURE PACK-------------------------------*)

Procedure Pack ;
Begin
  CH1    := CH1 AND #$F ;
  Ch2    := Ch2 << #$4 ;
  Ch1    := Ch1 OR Ch2 ;
End ;
```

```
BEGIN
  Ch1    := Str[7] ;
  Ch2    := Str[6] ;
  Pack ;
  Byte1  := Ch1 ;
  Ch1    := Str[5] ;
  Ch2    := Str[4] ;
  Pack ;
  Byte2  := Ch1 ;
  Ch1    := Str[3] ;
  Ch2    := Str[2] ;
  Pack ;
  Byte3  := Ch1 ;
End ;

(*----------------------PROCEDURE LEFT----------------------------*)

Procedure Left ;
Begin
  Xspeedint  := Xspeedint - 1000000 ;
  Xspeedint  := Xspeedint * (-1) ;
  Xcontrol := Xcontrol OR cw ;
End ;

(*----------------------PROCEDURE RIGHT---------------------------*)

Procedure Right ;
Begin
  Xspeedint  := Xspeedint + 1000000 ;
  Xcontrol := Xcontrol AND NOT cw ;
End ;

(*----------------------PROCEDURE UP------------------------------*)

Procedure Up ;
Begin
  Yspeedint  := Yspeedint  + 1000000 ;
  Ycontrol := Ycontrol OR cw ;
End ;

(*----------------------PROCEDURE DOWN----------------------------*)

Procedure Down ;
Begin
  Yspeedint := Yspeedint - 1000000 ;
  Yspeedint  := Yspeedint * (-1) ;
  Ycontrol := Ycontrol AND NOT cw ;
End ;
```

```
{------------------START OF PROCEDURE SETUPSYNTH--------------------}

BEGIN
  Sendline(CONCAT(C,L,'Enter speed in hundreths of mm per second',C,L)) ;
  Sendline(CONCAT('   i.e.  X = 43.56 ',C,L,'         Y = -21',C,L)) ;
  Sendline(CONCAT('         Will move at 0.4356 mm/sec to the right')) ;
  Sendline(CONCAT(C,L,'          and 0.21 mm/sec down.',C,L)) ;
  Error   := true ;
  WHILE Error = true DO
  Begin
    Sendline(CONCAT('Enter X speed (max. 46) : ')) ;
    Cmndflag  := false ;
    WHILE MesRx = false DO
      CheckVdu ;
    MesRx  := false ;
    Xspeed  := REAL(Data) ;
    Sendline(CONCAT(C,L));
    Dummy  := SQR(Xspeed) ;
    Dummy  := SQRT(Dummy) ;      (* Remove sign *)
    IF Dummy > 46.1 THEN
      Error := true
    ELSE
      Error := false ;
  End ;
  Error   := true ;
  WHILE Error = true DO
  Begin
    Sendline(CONCAT('Enter Y speed (max. 46) : ')) ;
    Cmndflag  := false ;
    WHILE MesRx = false DO
      CheckVdu ;
    MesRx  := false ;
    Yspeed  := REAL(Data) ;
    Sendline(CONCAT(C,L)) ;
    Dummy  := SQR(Yspeed) ;
    Dummy  := SQRT(Dummy) ;      (* Remove sign *)
    IF Dummy > 46.1 THEN
      Error := true
    ELSE
      Error := false ;
  End ;
  Sendline(CONCAT(C,L)) ;
  Xspeed   := Xspeed * Con74 * 1000000 ;
  Yspeed   := Yspeed * Con74 * 1000000 ;
  Xspeedint  := LONGINTEGER(Xspeed) ;
  Yspeedint  := LONGINTEGER(Yspeed) ;
  IF Xspeedint < 0 THEN
    Left
   ELSE
    Right ;
  IF Yspeedint < 0 THEN
    Down
   ELSE
    Up ;
  Xdummy := Xcontrol ;
  Ydummy := Ycontrol ;
  Xspeedstr  := STRING(Xspeedint) ;
  Yspeedstr  := STRING(Yspeedint) ;
  PackBCD(Yspeedstr) ;
```

```
  YsynthLSB  := Byte1 ;
  YsynthMidB := Byte2 ;
  YsynthMSB  := Byte3 ;
  PackBCD(Xspeedstr) ;
  XsynthLSB  := Byte1 ;
  XsynthMidB := Byte2 ;
  XsynthMSB  := Byte3 ;
End ;

{************************PROCEDURE GETSYNTHGO************************}

Procedure GetSynthGo ;
Begin
  Sendline(CONCAT(C,L,'Press any key to start velocity mode track :'));
  Charflag  := false ;
  WHILE Charflag = false DO
    Getchar(CH) ;
  Charflag  := false ;
  Sendline(CONCAT(C,L,L));
  xcontrol  := Xcontrol OR Synthenable ;
  ycontrol  := Ycontrol OR Synthenable ;
  xmotor    := xcontrol ;
  Ymotor    := ycontrol ;
End ;

{************************PROCEDURE TRACK*****************************}

Procedure Track ;

Begin
  IF VmodeActive  THEN
      Sendline(CONCAT(C,L,'Velocity mode already active',Bell,C,L))
  ELSE
  Begin
    VmodeActive := true ;
    Printpos ;
    Xposn   := 0 ;
    Yposn   := 0 ;          {To allow pushbutton use in VHOLD}
    Motorson ;
    Sendline(CONCAT(C,L,'Press RETURN to apply previous speed ',
                        'settings, or any other key',C,L));
    Sendline(CONCAT('to enter new speeds.....')) ;
    WHILE Charflag = false DO
      Getchar(CH) ;
    Charflag := false ;
    CH  := CH AND #$7F ;
    IF CH <> #13 THEN
      SetUpSynth
    ELSE
    Begin
      Xcontrol := Xdummy ;
      Ycontrol := Ydummy ;
    End ;
    GetsynthGo ;
    Sendline(CONCAT(C,L,'When Velocity mode is stopped - by ')) ;
    SendLine(CONCAT('command VSTOP - the X - Y mechanism',C,L,
                    'is automatically Centred .',C,L));
  End ;
End ;
```

```
{**********************PROCEDURE VCHANGE*****************************}

Procedure Vchange ;
Begin
  IF VmodeActive THEN
  Begin
    SetUpSynth ;
    Xmotor  := Xcontrol ;
    Ymotor  := Ycontrol ;
  End
  ELSE
    Sendline(CONCAT(C,L,'Velocity mode not active - do VMODE ',
                        'command first',C,L));
End ;

{***************************PROCEDURE VHOLD************************}

Procedure Vhold ;

Var
  Xmotorstatus  : byte ;
  Ymotorstatus  : byte ;

Begin
  IF VmodeActive THEN
  Begin
    Xmotorstatus  := xcontrol ;
    Ymotorstatus  := Ycontrol ;
    Xcontrol  := Xcontrol AND NOT Synthenable ;
    Xmotor  := Xcontrol ;
    Ycontrol  := Ycontrol AND NOT Synthenable ;
    Ymotor  := Ycontrol ;
    Sendline(CONCAT(C,L,L,'  Press any key to continue VMODE track :'));
    Charflag  := false ;
    WHILE Charflag = false DO
    Begin
      Getchar(CH) ;
      Pushbuttons ;
    End ;
    Charflag  := false ;
    Sendline(CONCAT(C,L,L));
    Xcontrol := Xmotorstatus ;
    Xmotor  := Xcontrol ;
    Ycontrol := Ymotorstatus ;
    Ymotor  := Ycontrol ;
  End
  ELSE
    Sendline(CONCAT(C,L,'Velocity Mode not active.',Bell,C,L));
End ;
```

```
{*****************INCLUDE FILE HANDSET.SA*********************************

    This code compiles as an include file into PROGRAM XYSLIDE.
    It contains routines for moving in X & Y in response to the
    pushbuttons on the handset.

*************************************************************************
*  version  last update                                                 *
*   1.00         ?                                                     *
*   1.01      19/12/86      Resructure sendline calls                  *
*   1.02      21/12/86      Handsetflag to PBactive                    *
*   1.03      28/01/87      corrections to 1.02                        *
*                                                                       *
*************************************************************************


}



{************************PROCEDURE PUSHBUTTONS*********************}

PROCEDURE PUSHBUTTONS ;

VAR
  Posnerror  : boolean ;
  Temp       : byte ;
  Temphex    : hex ;
  Xposnreal  : real ;
  Yposnreal  : real ;
  Checkpos   : real ;
  Msgsent    : boolean ;
  Rightenable : boolean ;
  Leftenable : boolean ;
  UpEnable   : boolean ;
  DownEnable : boolean ;

(*----------------------PROCEDURE STOP-------------------------*)

PROCEDURE STOP ;
begin
  Xcontrol := Xcontrol AND NOT Stepenable ;
  Ycontrol := Ycontrol AND NOT Stepenable ;
  Goingleft := false ;
  Goingright := false ;
  Goingup := false ;
  Goingdown := false ;
End ;
```

```
(*----------------------PROCEDURE RIGHT-------------------------------*)

PROCEDURE RIGHT ;

Procedure Subright ;
Begin
  Temp := Stat1 AND chr($C0) ;
  IF Temp = chr(0) THEN
  Begin
    Ycontrol := Ycontrol AND NOT Stepenable ;
    Goingup := false ;
    Goingdown := false ;
  End ;
  IF Goingright = false THEN
  Begin
    Goingright := true ;
    Xcontrol := Xcontrol AND NOT cw ;
    Xcontrol := Xcontrol OR stepenable ;
    Xmotor   := xcontrol ;
    IF Xtemp < 0 THEN
      Xtempcnt := ABS(Xtemp) ;
    IF Xtemp > 0 THEN
      Xtempcnt := 40 - Xtemp ;
    IF Xtemp = 0 THEN
      Xtempcnt := 40 ;
    Xstpcnt := HEX(Xtempcnt) ;
    Xchkstp := CHR(Xstpcnt) ;
    Switch := true ;
  End ;
End ;

Begin
  IF Rightenable = true THEN
    Subright
  ELSE
  Begin
    Xcontrol := Xcontrol AND NOT Stepenable ;
    Xmotor   := Xcontrol ;
  End ;
End ;

(*-----------------------PROCEDURE LEFT-------------------------*)

PROCEDURE LEFT ;

Procedure Subleft ;
Begin
  Temp := Stat1 AND chr($C0) ;
  IF Temp = chr(0) THEN
  Begin
    Ycontrol := Ycontrol AND NOT Stepenable ;
    Goingup := false ;
    Goingdown := false ;
  End ;
  IF Goingleft = false THEN
  Begin
    Goingleft := true ;
    Xcontrol := Xcontrol OR cw ;
    Xcontrol := Xcontrol OR stepenable ;
```

They are not used at present.

```
    Xmotor   := xcontrol ;
    IF Xtemp < 0 THEN
      Xtempcnt := 40 + Xtemp ;
    IF Xtemp > 0 THEN
      Xtempcnt := Xtemp ;
    IF Xtemp = 0 THEN
      Xtempcnt := 40 ;
    Xstpcnt := HEX(Xtempcnt) ;
    Xchkstp := CHR(Xstpcnt) ;
    Switch := true ;
  End ;
End ;

Begin
  IF Leftenable = true THEN
    Subleft
  ELSE
  Begin
    Xcontrol := Xcontrol AND NOT Stepenable ;
    Xmotor   := Xcontrol ;
  End ;
End ;

(*-------------------------PROCEDURE UP-------------------------*)

PROCEDURE UP ;

Procedure Subup ;
Begin
  Temp := Statl AND chr($30) ;
  IF Temp = chr(0) THEN
  Begin
    Xcontrol := Xcontrol AND NOT Stepenable ;
    GoingLeft := false ;
    Goingright := false ;
  End ;
  IF Goingup = false THEN
  Begin
    Goingup := true ;
    Ycontrol := Ycontrol OR cw ;
    Ycontrol := Ycontrol OR stepenable ;
    Ymotor   := Ycontrol ;
    IF Ytemp < 0 THEN
      Ytempcnt := ABS(Ytemp) ;
    IF Ytemp > 0 THEN
      Ytempcnt := 40 - Ytemp ;
    IF Ytemp = 0 THEN
      Ytempcnt := 40 ;
    Ystpcnt := HEX(Ytempcnt) ;
    Ychkstp := CHR(Ystpcnt) ;
    Switch := true ;
  End ;
End ;
```

```
Begin
  IF Upenable = true THEN
    SubUp
  ELSE
  Begin
    Ycontrol := Ycontrol AND NOT Stepenable ;
    Ymotor   := Ycontrol ;
  End ;
End ;

(*-------------------------PROCEDURE DOWN-------------------------*)
PROCEDURE DOWN ;

Procedure SubDown ;
Begin
  Temp:= Stat1 AND chr($30) ;
  IF Temp = chr(0) THEN
  Begin
    Xcontrol := Xcontrol AND NOT Stepenable ;
    GoingLeft := false ;
    GoingRight := false ;
  End ;
  IF Goingdown = false THEN
  Begin
    Goingdown := true ;
    Ycontrol := Ycontrol AND NOT cw ;
    Ycontrol := Ycontrol OR stepenable ;
    Ymotor   := Ycontrol ;
    IF Ytemp < 0 THEN
      Ytempcnt := 40 + Ytemp ;
    IF Ytemp > 0 THEN
      Ytempcnt := Ytemp ;
    IF Ytemp = 0 THEN
      Ytempcnt := 40 ;
    Ystpcnt := HEX(Ytempcnt) ;
    Ychkstp := CHR(Ystpcnt) ;
    Switch := true ;
  End ;
End ;

Begin
  IF Downenable = true THEN
    SubDown
  ELSE
  Begin
    Ycontrol := Ycontrol AND NOT Stepenable ;
    Ymotor   := Ycontrol ;
  End ;
End ;

{----------------------PROCEDURE UPANDLEFT-------------------------}

PROCEDURE UpandLeft ;
Begin
  Up ;
  Left ;
  Goingdown := false ;
  Goingright := false ;
End ;
```

```
{-----------------------PROCEDURE UPANDRIGHT-----------------------}

Procedure UpandRight ;
Begin
  Up ;
  Right ;
  Goingdown := false ;
  Goingleft := false ;
End ;

{-----------------------PROCEDURE DOWNANDLEFT----------------------}

PROCEDURE DownandLeft ;
Begin
  Down ;
  Left ;
  Goingup := false ;
  Goingright := false ;
End ;

{-----------------------PROCEDURE DOWNANDRIGHT---------------------}

PROCEDURE DownandRight ;
Begin
  Down ;
  Right ;
  GoingUp := false ;
  Goingleft := false ;
End ;

{-----------------------PROCEDURE INVALID--------------------------}

PROCEDURE Invalid ;
Begin
  Stop ;
  Buzzer ;
End ;

{----------------------PROCEDURE CHECKPOSN-------------------------}

PROCEDURE CHECKPOSN ;
Begin
  Xposnreal := REAL(Xposn) ;
  Yposnreal := REAL(Yposn) ;
  Xposnreal := Xposnreal / 2 ;
  Yposnreal := Yposnreal / 2 ;
  Checkpos := SQR(Xposnreal) + SQR(Yposnreal) ;
  IF Checkpos > Limit THEN
  Begin
    IF NOT Msgsent THEN
    Begin
      Stop ;
      Sendline(CONCAT(C,L,'X-Y SLIDE IN INVALID POSITION.',Bell,C,L,L)) ;
      Msgsent  := true ;
    End ;
    Buzzer ;
```

```
    IF Xposn > 0 THEN
      Rightenable  := false
     ELSE
      Leftenable   := false ;
    IF Yposn > 0 THEN
      Upenable   := false
     ELSE
      Downenable   := false ;
  End
  ELSE
  Begin
    Msgsent      := false ;
    Rightenable  := true ;
    Leftenable   := true ;
    Upenable     := true ;
    Downenable   := true ;
  End ;
End ;

{----------------------MAIN PROCEDURE----------------------------}

BEGIN
  IF (Statl AND #$F0) <> #0 THEN
  Begin
    Setimer  := false ;
    IF NOT PBactive THEN
    Begin
      PBactive  := true ;
      Motorson ;
      Speed := Hsetspeed ;
      Setspeed (Speed) ;
      Countercontrol := Countercontrol AND NOT Firqreset ;
      Msgsent  := false ;
      Rightenable  := true ;
      Leftenable   := true ;
      Upenable     := true ;
      Downenable   := true ;
    End ;
    CheckPosn ;
    Temp := Statl AND chr($F0) ;
    CASE Temp OF
       #$0  : Stop ;
      #$10  : Left ;
      #$20  : Right ;
      #$30  : Invalid ;
      #$40  : Up ;
      #$50  : UpandLeft ;
      #$60  : UpandRight ;
      #$70  : Invalid ;
      #$80  : Down ;
      #$90  : DownandLeft ;
      #$A0  : DownandRight ;
      #$B0..#$F0  : Invalid ;
    End ;
    ClrFir ;
  End
  ELSE
```

```
  Begin
    IF PBactive THEN
    Begin
      MskFir ;
      Goingleft  := false ;
      Goingright := false ;
      GoingUp    := false ;
      GoingDown  := false ;
      IF Setimer = false THEN
        Setimer  := true
      ELSE
      Begin
        Timer  := Timer + 1 ;
        IF Timer = 1000 THEN
        Begin
          Setimer  := false ;
          Timer  := 0 ;
          PBactive  := false ;
          Motorsoff ;
          Switch  := false ;
        End ;
      End ;
    End ;
  End ;
End ;
```

```
{*****************INCLUDE FILE VDUDRIVE.SA*********************************

   This code compiles as an include file in PROGRAM XYSLIDE. It
   contains routines for handling Terminal input and output.

*************************************************************************
*  version  last update                                                 *
*    1.00       ?                                                      *
*    1.01    28/01/87         restructure sendline calls               *
*                                                                       *
*************************************************************************



}


{************************PROCEDURE SENDCHAR************************}

Procedure Sendchar(CH : char) ;
begin
  WHILE (Vdusts AND TxRdy) <> TxRdy DO
  begin
  end ;
  VduData  := CH ;
End ;

{************************PROCEDURE SENDLINE*****************************}

Procedure Sendline(Line : string[80]) ;
Begin
  For I := 1 TO LENGTH(line) DO
  Begin
    CH  := Line[I] ;
    Sendchar(CH) ;
  End ;
End ;

{************************PROCEDURE GETCHAR*****************************}

Procedure Getchar(Var CH : char) ;
begin
  IF (Vdusts AND RxRdy) = Rxrdy THEN
  Begin
    CH  := VduData ;
    Charflag  := true ;
    Sendchar(CH) ;
  End ;
End ;
```

```
{**********************PROCEDURE CHECKVDU*************************}
{Checks terminal input, and puts character on stack if present}

Procedure CheckVdu ;

{---------------------PROCEDURE PUTCHINBUF----------------------}

Procedure PutCHinBuf ;

Procedure Delete ;
Begin
  Sendline(CONCAT(#$20,#$08));
  Ptr  := MsgInBuf[0] ;
  Ptr  := Ptr - #1 ;
  MsgInBuf[0]  := Ptr ;
End ;

Begin
  IF CH = #$08 THEN
    Delete
  ELSE
  Begin
    Ptr  := MsgInBuf[0] ;
    Ptr  := Ptr + #1 ;
    MsgInBuf[Ptr]  := CH ;
    MsgInBuf[0]  := Ptr ;
  End ;
End ;

{----------------------START OF PROCEDURE CHECKVDU-----------------}

Begin
  Getchar(CH) ;
  IF Charflag = true THEN
  Begin
    Charflag := false ;
    Ch   :=  CH AND #$7F ;
    IF CH <> #13 THEN
      PutCHinBuf
    ELSE
    Begin
      MesRx  := true ;
      IF Cmndflag THEN
        Cmnd  := MsgInBuf
      ELSE
        Data  := MsgInBuf ;
      CH  := #10 ;
      Sendchar(CH) ;
      Ptr  := #0 ;
      MsgInBuf[0]  := Ptr ;
    End ;
  End ;
End ;
```

```
{************************PROCEDURE SETUPFUNCKEYS*******************}

Procedure SetUpFuncKeys ;

CONST
  Esc = #$1B ;
  A   = #$7C ;
  CR  = #$0D ;
  EM = #$19 ;

BEGIN
  Sendline(CONCAT(Esc,A,'1','1','CENTR',CR,EM,
                  Esc,A,'2','1','MOV',CR,EM,
                  Esc,A,'3','1','GMCEN',CR,EM,
                  Esc,A,'4','1','GMVIEW',CR,EM));
  Sendline(CONCAT(Esc,A,'5','1','RVMIN',CR,EM,
                  Esc,A,'6','1','RVMOUT',CR,EM,
                  Esc,A,'7','1','GUIDE',CR,EM,
                  Esc,A,'8','1','AQUIRE',CR,EM,
                  Esc,A,';','1','HELP',CR,EM));
END ;
```

```
{*****************INCLUDE FILE AQBOX.SA***********************************

   This code compiles as an include file in PROGRAM XYSLIDE.
   It contains routines for controlling the Acquisition box
   mirrors positions.

*************************************************************************
*  version  last update                                                 *
*    1.00       ?                                                      *
*    1.01    21/12/86     Restructure sendline calls                   *
*    2.00    21/12/86     checkselstat to function Computereselected   *
*    2.01    21/12/86     CheckGMposition to Function GMinBeam         *
*    2.02    21/12/86     CheckRVMposition to Function RVMinBeam       *
*    2.03    21/12/86     Changed Delay calls                          *
*    2.04    27/01/86     corrections to 2.03                          *
*    2.05    28/01/86     corrections to 2.04                          *
*                                                                       *
*************************************************************************


}



{*********************FUNCTION COMPUTERSELECTED*******************}

FUNCTION Computerselected : boolean ;
Begin
  IF Statl AND Selstatmask = Selstatmask THEN
    Computerselected := true
  ELSE
    Computerselected := false;
End ;

{*********************FUNCTION GMINBEAM*****************************}

Function GMinBeam : boolean ;
Begin
  IF Statl AND GMCenmask = GMcenMask THEN
    GMinBeam := false
  ELSE
    GMinBeam := true ;
End ;

{*********************FUNCTION RVMINBEAM****************************}

Function RVMinBeam : boolean ;
Begin
  IF Statl AND RVMoutmask  = RVMoutmask THEN
    RVMinBeam := false
  ELSE
    RVMinBeam := true ;
End ;
```

```
{**********************PROCEDURE ERRMESGE1**************************}

PROCEDURE Errmesge1 ;
Begin
  Sendline(CONCAT('CANNOT CONTROL AQUISITION BOX',Bell,C,L,'Check Compu'));
  Sendline(CONCAT('ter/Manual select switch on Aquisition box Panel',C,L));
End ;

{**********************PROCEDURE ERRMESGE2**************************}

PROCEDURE Errmesge2 ;
Begin
  Sendline(CONCAT('MIRROR NOT MOVING - Check gas pressure',Bell,C,L)) ;
End ;

{**********************PROCEDURE GRATON*****************************}

PROCEDURE Graton ;
Begin
  IF ComputerSelected THEN
  Begin
    IF RVMinBeam THEN
      Sendline(CONCAT('Rear View Mirror not in beam - cannot',
                      'switch graticule on.',Bell,C,L))
    ELSE
    Begin
      Aqbox := Aqbox OR Gratcontrol ;
      Sendline(CONCAT('Graticule illumination is ON. ',C,L));
    End ;
  End
  ELSE
    Errmesge1 ;
End ;

{**********************PROCEDURE GRATOFF****************************}

PROCEDURE Gratoff ;
Begin
  IF ComputerSelected THEN
  Begin
    Aqbox := Aqbox AND NOT Gratcontrol ;
    Sendline(CONCAT('Rear View Eyepiece graticule illumination OFF',C,L));
  End
  ELSE
    ErrMesge1;
End ;

{**********************PROCEDURE BUZZER*****************************}

PROCEDURE Buzzer ;
Begin
   Aqbox := Aqbox OR Buzz ;
   Aqbox := Aqbox AND NOT Buzz ;
End ;
```

```
{********************PROCEDURE GMVIEW*******************************}

PROCEDURE GMview ;
Begin
  IF ComputerSelected THEN
  Begin
    IF GMinBeam THEN
      Sendline(CONCAT('Guide mirror already in view position',Bell,C,L))
    ELSE
    Begin
      Aqbox := Aqbox OR Gmposcontrol ;
      Delay(30000) ;
      IF GMinBeam THEN
        Sendline(CONCAT('Guider mirror in view position'))
      ELSE
        Sendline(CONCAT('Guider Mirror not moving - Check gas pressure',
                        Bell,C,L));
    End ;
  End
  ELSE
    Errmesgel ;
End ;

{*********************PROCEDURE GMCENTRE*****************************}

PROCEDURE GMCentre ;
Begin
  IF ComputerSelected THEN
  Begin
    IF GMinBeam THEN
    Begin
      Aqbox := Aqbox AND NOT GMposcontrol ;
      Delay(30000) ;
      IF GMinBeam THEN
        Errmesge2
      ELSE
        Sendline(CONCAT('Guider mirror Centre',C,L)) ;
    End
    ELSE
      Sendline(CONCAT('Guide mirror already Centred',Bell,C,L));
  End
  ELSE
    Errmesgel ;
End ;
```

```
{**********************PROCEDURE RVMOUTBEAM*************************}

PROCEDURE RVMoutBeam ;
Begin
  IF ComputerSelected THEN
  Begin
    IF RVMinBeam THEN
    Begin
      Aqbox := Aqbox AND NOT RVMposControl ;
      Delay(30000) ;
      IF RVMinBeam THEN
        Errmesge2
      ELSE
        Sendline(CONCAT('Rear View Mirror out of Beam',C,L)) ;
    End
    ELSE
      Sendline(CONCAT('Rear view mirror already out of Beam',Bell,C,L));
  End
  ELSE
    Errmesge1 ;
End ;

{**********************PROCEDURE RVMINTOBEAM************************}

PROCEDURE RVMintoBeam ;
Begin
  IF ComputerSelected THEN
  Begin
    IF RVMinBeam THEN
      Sendline(CONCAT('Rear view mirror already in beam',C,L,Bell))
    ELSE
    Begin
      Aqbox := Aqbox OR RVMposcontrol ;
      Delay(30000) ;
      IF RVMinBeam THEN
        Sendline(CONCAT('Rear view mirror in beam',C,L))
      ELSE
        Errmesge2 ;
    End ;
  End
  ELSE
    Errmesge1 ;
End ;
```

```
**********ASSEMBLY SOURCE CODE POTPOSN.SA**********

*  This file contains assembly language routines
*   for reading the 10 bit ADC and manipulating the
*   data into a form acceptable for the PASCAL.
*
************************************************************
*  version  last update                                    *
*   1.00        ?                                          *
*                                                          *
************************************************************
*
*
*
*SUBROUTINE TO GET Y-POT COORDINATES
*
       NAM    POTPOSN
*
YCOORD PSHU   A,B,X,Y       SAVE REGISTERS
       LDY    6,U           GET YPOS ADDRESS OFF STACK
       LDA    ADC2          GET MS BYTE
       LDB    ADC2          GET LS BYTE
       LDX    #$6
LOOP   LSRA                 ROTATE 16 BITS UNTIL
       RORB                 RIGHT JUSTIFIED
       LEAX   -1,X
       BNE    LOOP
       STD    0,Y           UPDATE YPOS VARIABLE
       PULU   A,B,X,Y       RESTORE REGISTERS
       LEAU   2,U           REMOVE YPOS ADDRESS FROM STACK
       RTS                  RETURN
*
*SUBROUTINE TO GET X-POT COORDINATES
*
XCOORD PSHU   A,B,X,Y       SAVE REGISTERS
       LDY    6,U           GET ADDRESS OF XPOS OFF STACK
       LDA    ADC1          GET M.S. BYTE
       LDB    ADC1          GET L.S. BYTE
       LDX    #$6
LOOP1  LSRA                 ROTATE 16 BITS UNTIL
       RORB                 RIGHT JUSTIFIED
       LEAX   -1,X
       BNE    LOOP1
       STD    0,Y           UPDATE XPOS VARIABLE
       PULU   A,B,X,Y       RESTORE REGISTERS
       LEAU   2,U           REMOVE XPOS ADDRESS FROM STACK
       RTS                  RETURN
*
ADC1   EQU    $E040
ADC2   EQU    $E041
*
       XDEF   XCOORD
       XDEF   YCOORD
*
       END
```

```
*****************ASSEMBLY SOURCE CODE INTRPT.SA*****************************
*
*
*  File for linking into XYSLIDE program at link/load
*    time . It contains routines for setting and
*    clearing the FIRQ mask bit, and for setting up
*    the FIRQ vectors.
*
************************************************************
*  version  last update                                    *
*   1.00         ?                                         *
*   1.01     04/12/86    corrected FIRQ masking           *
*   1.02     21/12/86    Mskirq to MskFir, also ClrFir    *
*                                                          *
************************************************************
*
*
*ROUTINES TO SET AND CLEAR INTERUPT MASKS
*
*
       NAM    INTRPT
*
MSKFIR ORCC   #$40     MASK OUT FIRQ
       RTS
CLRFIR ANDCC  #$BF     ENABLE FIRQ
       RTS
*
************************************************************
*
*ROUTINE TO SET UP FIRQ VECTOR
*
FIRVEC PSHU   A,X      SAVE REGISTERS
       LDX    #INTADD  GET FIRQ. RTN. START ADDRESS
       LDA    #$7E     OPCODE FOR JMP(EXT)
       STA    $0000
       STX    $0001
       LDX    #$0
       STX    $FFF6    SET UP FIRQ VECTOR
       PULU   A,X      RESTORE REGISTERS
       RTS
*
       XREF   INTADD
*
       XDEF   MSKFIR
       XDEF   CLRFIR
       XDEF   FIRVEC
*
       END
```

```
*****************ASSEMBLY SOURCE CODE FIRQ1.SA*****************************
*
*FIRQ ROUTINE FOR MOVING STEPPING MOTORS OF XYSLIDE
*
*
************************************************************
*
*  04/12/86   1.01    Top of stack TSTACK added
*  28/01/87   1.02    TSTACK = $5FF
*  28/01/87   1.03    TSTACK = $4FF
*   5/02/87   1.04    TIDY UP COMMENTS
*   6/02/87   1.05    PUT IN TEST CODE TO CHECK SLOTS
*   6/02/87   1.06    CORRECTIONS TO 1.05
*
************************************************************
*
*NOTE !! TSTACK MUST BE CHANGED IF PASCAL
*   LEVEL DATASTACK ALLOCATION CHANGES.
*
************************************************************
*
       NAM    FIRQRTNE
*
************************************************************
*
TSTACK EQU    $4FF     PASCAL DATASTACK START
*
************************************************************
*
AQBOX  EQU    $E082
ICNTRL EQU    $E032
STAT2  EQU    $E081
XMOTR  EQU    $E000
YMOTR  EQU    $E008
BASE   EQU    TSTACK-$16   ADDR. OF GLOBAL STACK START
*
************************************************************
*
       XREF   XSLTCH
       XREF   YSLTCH
       XREF   SWITCH
       XREF   PBUTON
       XREF   INITIA
       XREF   INIT
       XREF   XCONTR
       XREF   YCONTR
       XREF   XCHKST
       XREF   YCHKST
       XREF   XMEROR
       XREF   YMEROR
       XREF   XPOSN
       XREF   YPOSN
       XREF   XDEST
       XREF   YDEST
*
       XDEF   INTADD
       XDEF   OUT
*
************************************************************
```

```
*
INTADD PSHU   A,B,X,Y  SAVE WORKING REGISTERS
       LDY    BASE     GET GLOBAL BASE REGISTER
       LDA    ICNTRL   GET TIMER CONTROL BYTE
       ORA    #$04     SET TIMER ONE(FIRQ) RESET, I.E.
       STA    ICNTRL   CLEAR INTR. AT SOURCE.
*
       LDA    INITIA,Y IS INITIALIZE IN PROGRESS?
       LBNE   INIT     YES, DO IT,
       LDA    SWITCH,Y NO , MOVING FROM PUSHBUTTONS?
       LBNE   PBUTON   YES,USE PBUTON PATH
*
* Start of doing X motor step
*
       LDA    XCONTR,Y GET XMOTOR STATUS AND
       PSHU   A        SAVE IT FOR LATER
       ANDA   #$80     ISOLATE STEP ENABLE BIT
       LBEQ   NOXSTP   X STEP REQUIRED ?
       PULU   A        YES , GET X STATUS TO STEP
       ORA    #$20
       STA    XMOTR    STEP = 1
       ANDA   #$DF
       NOP             STRETCH STEP PULSE
       STA    XMOTR    STEP = 0, DONE
*
       DEC    XCHKST,Y DECR.MODULO 40 COUNTER
       BNE    NXPHCH   IF N.E. ZERO, NOCHECK,
       LDB    STAT2    IF EQ. ZERO, CHECK PH0
       ANDB   #$40     IS TRUE
       BEQ    XERROR   FLAG ERROR IF NOT.
       CLR    XMEROR,Y ELSE CLR ERROR AND
       LDB    #$28     RESET MOD 40 COUNTER,
       STB    XCHKST,Y
       LDB    #$03     AND SET SLOTCHECK COUNTER
       STB    XSLTCH,Y
NXPHCH LDB    XSLTCH,Y GET SLOTCHECK COUNTER
       BEQ    NOXCHK   IF = 0, NO CHECK.
*
       LDB    AQBOX    GET CURRENT AQBOX STATUS
       ORB    #$04     AND SET BIT2 FOR SCOPE TESTS
       STB    AQBOX    OF SLOT
*
       LDB    STAT2    ELSE CHECK SLOT
       ANDB   #$04     SLOT TRUE?
       BNE    CLRXER   YES,
*
       LDB    AQBOX    GET AQBOX STATUS AND
       ANDB   #$B      RESET BIT 2
       STB    AQBOX
*
       DEC    XSLTCH,Y NO, DECREMENT SLOTCHECK COUNTER
       BNE    NOXCHK   IF N.E. ZERO, TRY NEXT TIME
XERROR LDB    #$FF     ELSE SET ERROR FLAG
       STB    XMEROR,Y
       BRA    NOXCHK   NOW UPDATE POSITION COUNTER
*
CLRXER CLR    XMEROR,Y CLEAR ERROR FLAG AND
       CLR    XSLTCH,Y SLOT CHECK FLAG.
*
```

```
        LDB     AQBOX     GET AQBOX STATUS AND
        ANDB    #$B       RESET BIT 2
        STB     AQBOX
*
*
NOXCHK  LDX     XPOSN,Y   NOW UPDATE POSITION COUNTER.
        ANDA    #$08      WHAT DIRECTION?
        BEQ     RIGHT
        LEAX    -1,X      IF LEFT, DECR POSITION,
        BRA     DESTX
RIGHT   LEAX    1,X       IF RIGHT, INCR POSITION.
*
DESTX   CMPX    XDEST,Y   AT DESTINATION?
        BNE     MOREX     NO , MORE STEPS TO DO.
        ANDA    #$7F      YES , DISABLE MORE X STEPS
        STA     XCONTR,Y  AND SAVE.
MOREX   STX     XPOSN,Y   SAVE STEP COUNT
        BRA     YSTEP     AND CHECK Y MOTOR
NOXSTP  PULU    A         ADJUST USER STACK
*
*************************************************************
*
* Start of doing Y step.
*
YSTEP   LDA     YCONTR,Y  GET YMOTOR STATUS
        PSHU    A
        ANDA    #$80      ISOLATE STEP ENABLE BIT
        LBEQ    NOYSTP    Y STEP REQUIRED, IF NO , EXIT
        PULU    A         YES , GET STATUS TO STEP
        ORA     #$20
        STA     YMOTR     STEP = 1
        ANDA    #$DF
        NOP               STRETCH STEP PULSE
        STA     YMOTR     STEP = 0
*
        DEC     YCHKST,Y  DECR.MODULO 40 COUNTER
        BNE     NYPHCH    IF N.E. ZERO, NOCHECK,
        LDB     STAT2     IF EQ. ZERO, CHECK PH0
        ANDB    #$80      IS TRUE
        BEQ     YERROR    FLAG ERROR IF NOT.
        CLR     YMEROR,Y  ELSE CLR ERROR AND
        LDB     #$28      RESET MOD 40 COUNTER,
        STB     YCHKST,Y
        LDB     #$02      AND SET SLOTCHECK COUNTER
        STB     YSLTCH,Y
NYPHCH  LDB     YSLTCH,Y  GET SLOTCHECK COUNTER
        BEQ     NOYCHK    IF = 0, NO CHECK.
*
        LDB     AQBOX     GET CURRENT AQBOX STATUS
        ORB     #$0E      AND SET BIT0 FOR SCOPE TESTS
        STB     AQBOX     OF SLOT
*
        LDB     STAT2     ELSE CHECK SLOT
        ANDB    #$08      SLOT TRUE?
        BNE     CLRYER    YES , CLEAR ERROR FLAG
*
        LDB     AQBOX     GET AQBOX STATUS AND
        ANDB    #$1       RESET BIT 0
        STB     AQBOX
```

```
*
        DEC     YSLTCH,Y NO, DECREMENT SLOTCHECK COUNTER
        BNE     NOYCHK   IF N.E. ZERO, TRY NEXT TIME
YERROR  LDB     #$FF     ELSE SET ERROR FLAG
        STB     YMEROR,Y
        BRA     NOYCHK   NOW UPDATE POSITION COUNTER
*
CLRYER  CLR     YMEROR,Y CLEAR ERROR FLAG AND
        CLR     YSLTCH,Y SLOT CHECK FLAG.
*
        LDB     AQBOX    GET AQBOX STATUS AND
        ANDB    #$1      RESET BIT 0
        STB     AQBOX
*
*
NOYCHK  LDX     YPOSN,Y  NOW UPDATE POSITION COUNTER
        ANDA    #$08     WHAT DIRECTION ?
        BEQ     DOWN
        LEAX    1,X      IF UP, INCR POSITION
        BRA     DESTY
DOWN    LEAX    -1,X     IF DOWN,DECR POSITION
*
DESTY   CMPX    YDEST,Y  AT DESTINATION ?
        BNE     MOREY    NO , MORE STEPS TO DO.
        ANDA    #$7F     YES , DISABLE MORE STEPS
        STA     YCONTR,Y AND SAVE
MOREY   STX     YPOSN,Y  SAVE STEP COUNT
        BRA     OUT
NOYSTP  PULU    A
        BRA     OUT
*
OUT     LDA     ICNTRL    GET TIMER CONTROL BYTE
        ANDA    #$FB      ENABLE INTERUPTS
        STA     ICNTRL
        PULU    A,B,X,Y  RESTORE ORIGINAL CONTENTS
        RTI
*************************************************************
*
        END
```

```
*****************ASSEMBLY SOURCE CODE FIRQ2.SA*****************************
*
* This is part of the FIRQ routine. It assembles
* separately for linking at Link/Load time.
* It contains the routines for Initial Slew,
* i.e. for moving in X & Y without error checking.
*
*************************************************************
*  version  last update                                     *
*    1.00       ?                                          *
*                                                          *
*************************************************************
*
*
*
*
*FIRQ ROUTINE FOR INITIALIZE SLEW
*
       NAM    FIRQ2
*
INIT   LDA    XCONTR,Y GET XMOTOR STATUS
       PSHU   A
       ANDA   #$80     ISOLATE STEP ENABLE BIT
       BEQ    NOXSTP   NO X STEP REQUIRED, Y ?
       PULU   A        GET X STATUS TO STEP
       ORA    #$20
       STA    XMOTR    STEP = 1
       ANDA   #$DF
       NOP             STRETCH STEP PULSE
       STA    XMOTR    STEP = 0, DONE
NOXCHK LDX    XSTEPS,Y NOW COUNT
       LEAX   -1,X     THE STEP AND CHECK
       BNE    MOREX    IF MORE STEPS TO DO.
       ANDA   #$7F     DISABLE MORE X STEPS
       STA    XCONTR,Y AND RESTORE TO STACK
MOREX  STX    XSTEPS,Y SAVE STEP COUNT
       BRA    YSTEP    AND CHECK Y MOTOR
NOXSTP PULU   A        ADJUST USER STACK
YSTEP  LDA    YCONTR,Y GET YMOTOR STATUS
       PSHU   A
       ANDA   #$80     ISOLATE STEP ENABLE
       BEQ    NOYSTP   NO Y STEP REQUIRED, OUT
       PULU   A
       ORA    #$20
       STA    YMOTR    STEP = 1
       ANDA   #$DF
       NOP             STRETCH STEP PULSE
       STA    YMOTR    STEP = 0
NOYCHK LDX    YSTEPS,Y COUNT THE
       LEAX   -1,X     STEP AND CHECK
       BNE    MOREY    IF MORE STEPS TO DO.
       ANDA   #$7F     DISABLE MORE STEPS
       STA    YCONTR,Y AND RESTORE TO STACK.
MOREY  STX    YSTEPS,Y  SAVE STEP COUNT
       LBRA   OUT
NOYSTP PULU   A
       LBRA   OUT
```

```
*
XMOTR   EQU     $E000
YMOTR   EQU     $E008
*
        XDEF    INIT
*
        XREF    OUT
        XREF    XSTEPS
        XREF    YSTEPS
        XREF    XCONTR
        XREF    YCONTR
*
        END
```

```
*****************ASSEMBLY SOURCE CODE FIRQ3.SA*****************************
*
* This code is part of the FIRQ routine . It contains
* code for moving in X & Y in response to the pushbuttons
* i.e. There is no reference to a destination position.
*
*************************************************************
*  version  last update                                     *
*    1.00       ?                                          *
*                                                          *
*************************************************************
*
*
*
*
*
PBUTON LDA    XCONTR,Y GET XMOTOR STATUS
       PSHU   A
       ANDA   #$80     ISOLATE STEP ENABLE BIT
       BEQ    NOXSTP   NO X STEP REQUIRED, Y ?
       PULU   A        GET X STATUS TO STEP

       ORA    #$20
       STA    XMOTR    STEP = 1
       ANDA   #$DF
       NOP             STRETCH STEP PULSE
       STA    XMOTR    STEP = 0, DONE

       DEC    XCHKST,Y DECR.MODULO 40 COUNTER
       BNE    NOXCHK   IF N. E. ZERO, NO CHECK
       LDB    STAT2    IF EQ. ZERO, CHECK
       ANDB   #$44     POSITION BY ISOLATING
       CMPB   #$44     XPHO & XSLOT. IF
       BNE    XERROR   NOT TRUE,ERROR CONDITION
       CLR    XMEROR,Y IF TRUE,CLR ERROR FLAG.
       LDB    #$28     RESET MODULO 40
       STB    XCHKST,Y COUNTER
       BRA    NOXCHK

XERROR LDB    #$FF     SET ERROR FLAG TRUE
       STB    XMEROR,Y

NOXCHK LDX    XPOSN,Y  NOW CHECK POSITION
       ANDA   #$08     WHAT DIRECTION?
       BEQ    RIGHT
       LEAX   -1,X     IF LEFT, DECR POSITION
       BRA    DESTX    AND CHECK
RIGHT  LEAX   1,X      IF RIGHT, INCR POSITION
DESTX  STX    XPOSN,Y  SAVE STEP COUNT
       BRA    YSTEP    AND CHECK Y MOTOR

NOXSTP PULU   A        ADJUST USER STACK

YSTEP  LDA    YCONTR,Y GET YMOTOR STATUS
       PSHU   A
       ANDA   #$80     ISOLATE STEP ENABLE
       BEQ    NOYSTP   NO Y STEP REQUIRED, OUT
       PULU   A
```

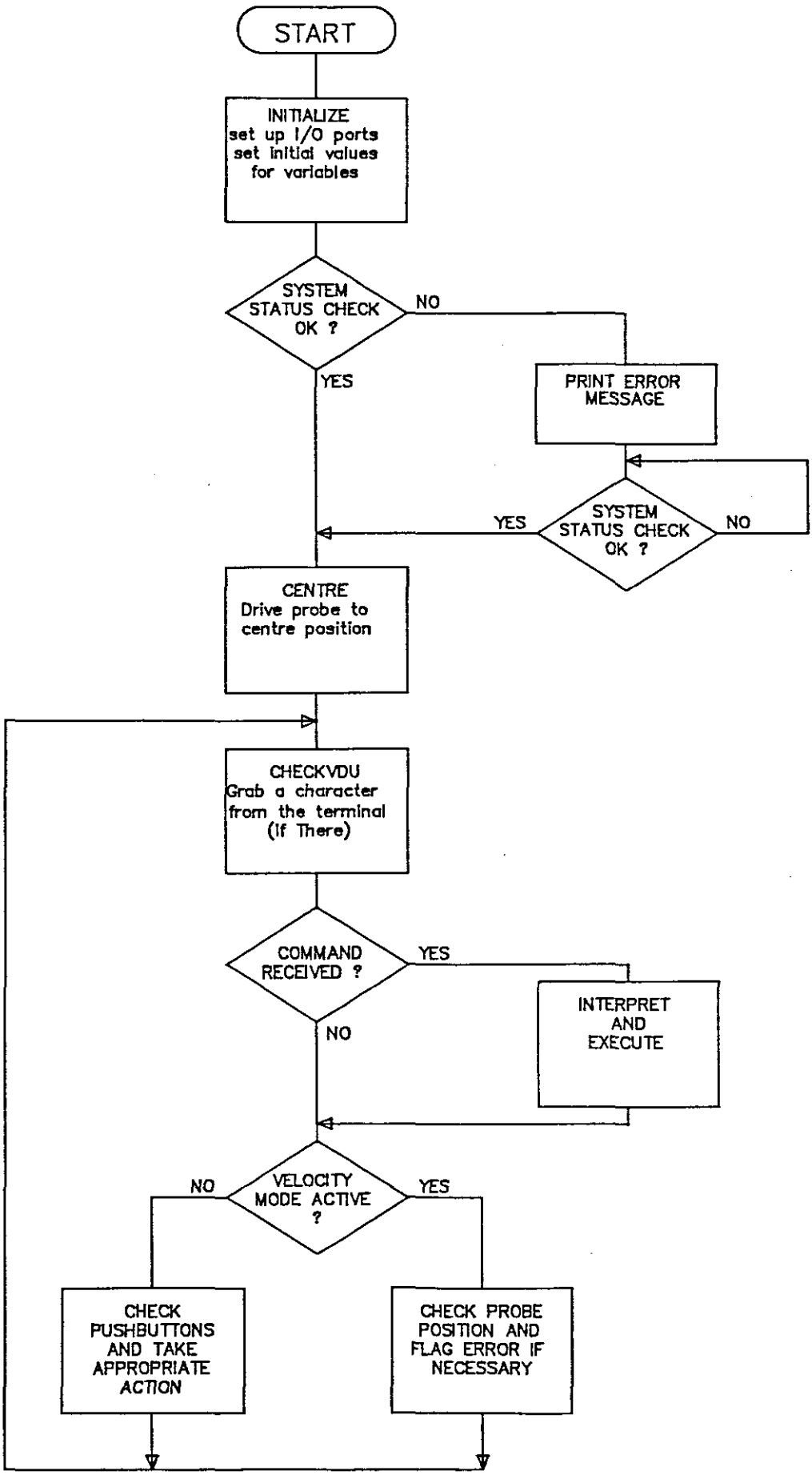
```
        ORA     #$20
        STA     YMOTR     STEP = 1
        ANDA    #$DF
        NOP               STRETCH STEP PULSE
        STA     YMOTR     STEP = 0

        DEC     YCHKST,Y DECR.MODULO 40 COUNTER
        BNE     NOYCHK    IF N. E. ZERO, NO CHECK
        LDB     STAT2     IF EQ. ZERO, CHECK
        ANDB    #$88      POSITION BY ISOLATING
        CMPB    #$88      YPHO & XSLOT. IF
        BNE     YERROR    NOT TRUE,ERROR CONDITION
        CLR     YMEROR,Y IF TRUE,CLR ERROR FLAG.
        LDB     #$28      RESET MODULO 40
        STB     YCHKST,Y COUNTER
        BRA     NOYCHK

YERROR LDB      #$FF      SET ERROR FLAG TRUE
        STB     YMEROR,Y

NOYCHK LDX      YPOSN,Y   NOW CHECK POSITION
        ANDA    #$08      WHAT DIRECTION ?
        BEQ     DOWN
        LEAX    1,X       IF UP, INCR POSITION
        BRA     DESTY     AND CHECK
DOWN    LEAX    -1,X      IF DOWN,DECR POSITION
DESTY   STX     YPOSN,Y    SAVE STEP COUNT

        LBRA    OUT
NOYSTP  PULU    A
        LBRA    OUT
*
STAT2   EQU     $E081
XMOTR   EQU     $E000
YMOTR   EQU     $E008
*
        XREF    XCONTR
        XREF    YCONTR
        XREF    XCHKST
        XREF    YCHKST
        XREF    XMEROR
        XREF    YMEROR
        XREF    XPOSN
        XREF    YPOSN
        XREF    OUT
*
        XDEF    PBUTON
*
        END
```

```
******************ASSEMBLY SOURCE CODE VECTOR.SA*****************************
*
*  This file contains code for PROMMING the
*  6809  vectors at top of memory. It is
*  assembled with option ABS.
*
************************************************************
*  version  last update                                    *
*    1.00       ?                                         *
*    1.01    20/01/86     IRQ vector added                *
*                                                         *
************************************************************
*
*
*
*SET UP VECTOR MAP OF XYSLIDE
*VER4 JAN 86
*
      NAM   VECT1
      OPT   ABS
*
      ORG   $F000
*
      LDA   #$00        DUMMY STATEMENTS
      LDA   #$FF
*
      ORG   $FFF2       START OF VECTOR MAP
*
      FDB   $8000
      FDB   $8000
      FDB   $0000       LOCATION OF FIRQ ROUTINE START
      FDB   $0003       LOCATION OF IRQ ROUTINE START
      FDB   $8000
      FDB   $0006       LOCATION OF NMI ROUTINE START
      FDB   $8000
*
      END
```

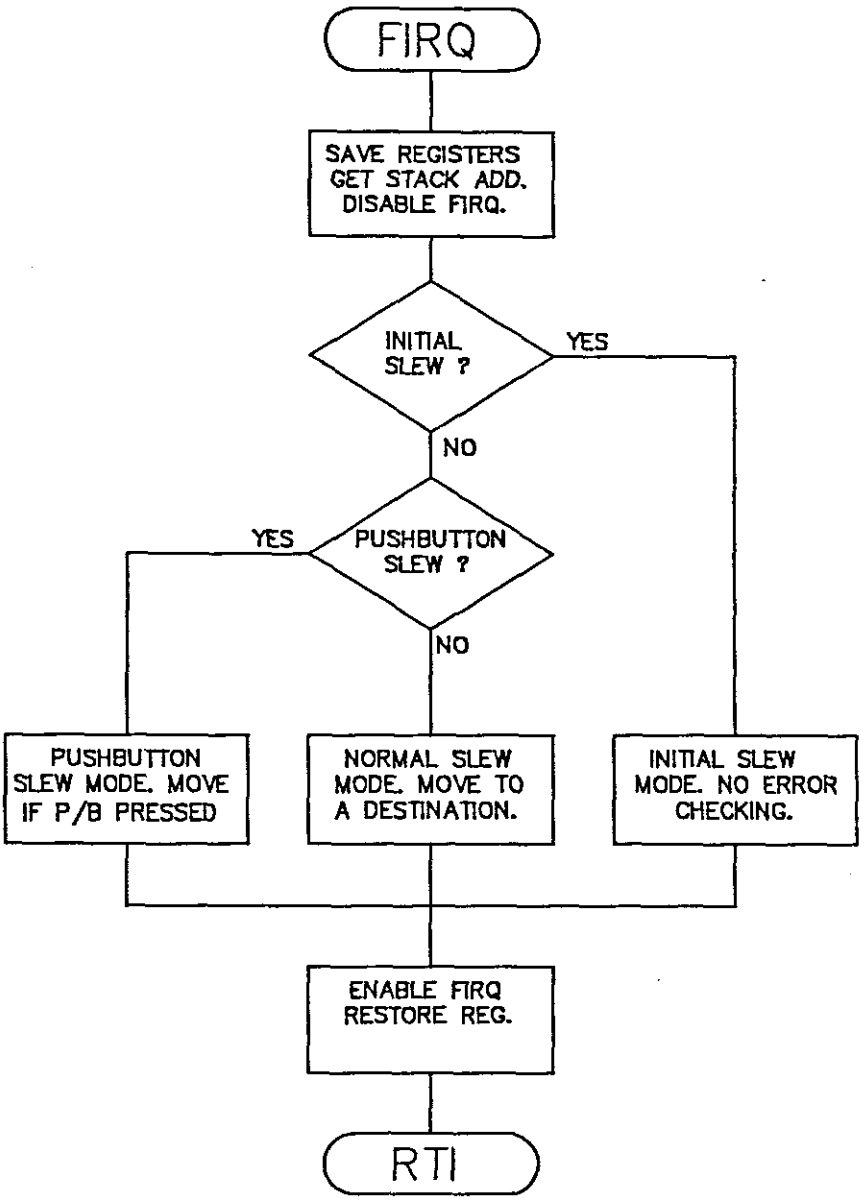
6. Flowcharts.

Flowcharts are shown for the major program blocks and the fast interrupt routine. The flowcharts are not totally explicit - they are drawn to show the important routes and tasks performed in the program. To see the details within each flow read the appropriate listing.

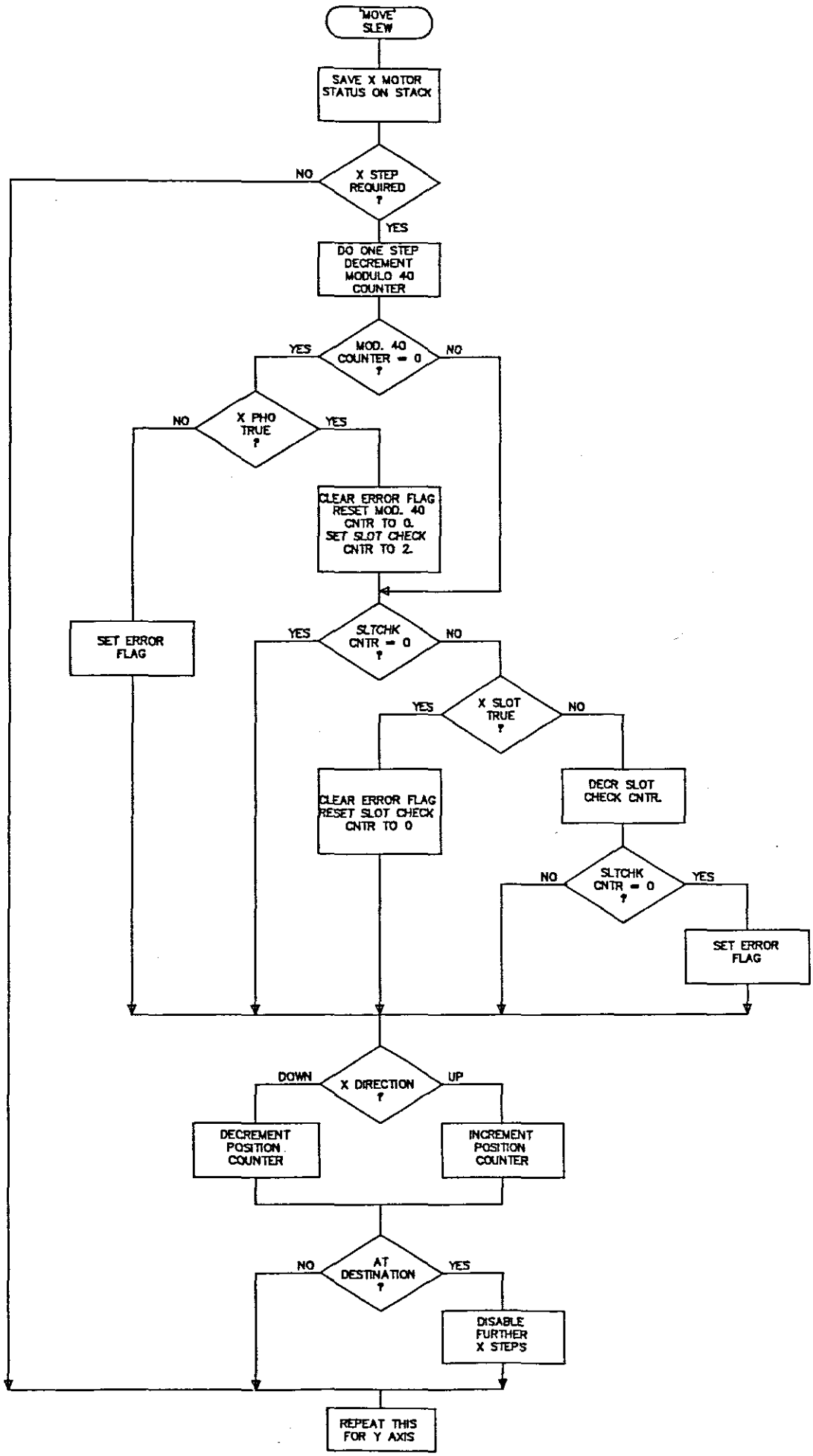
XYSLIDE PROGRAM FLOWCHARTS
MAIN PROGRAM LOOP



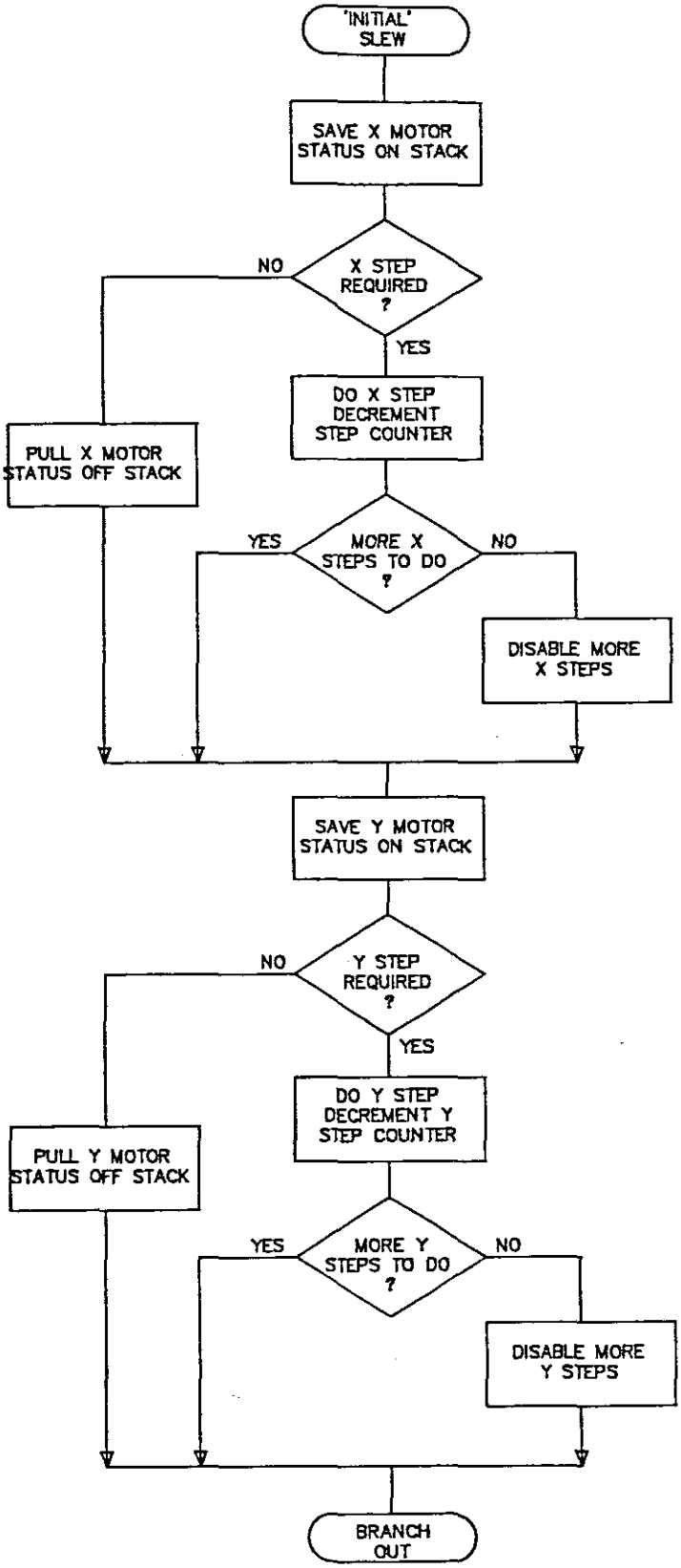
FAST INTERRUPT ROUTINE FLOWCHART
(ASSEMBLER CODE)



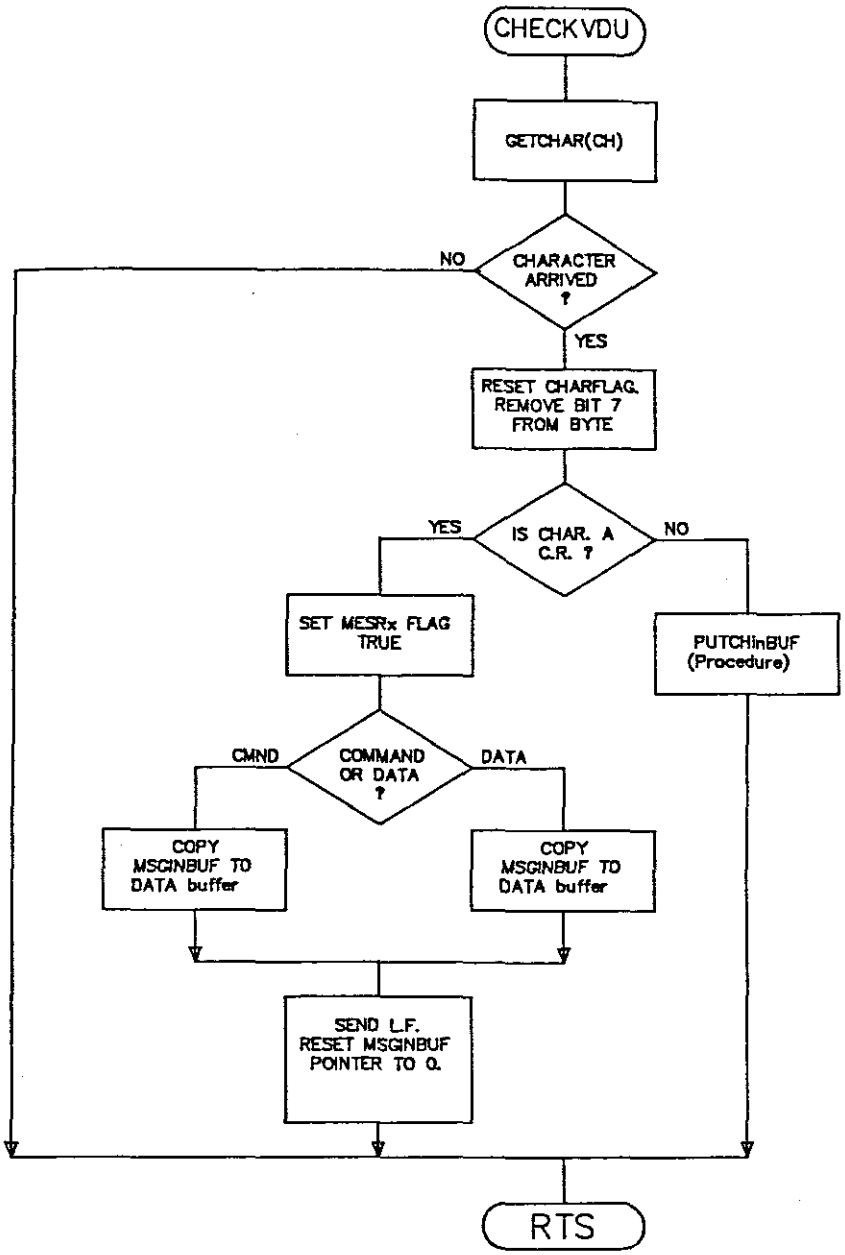
FAST INTERRUPT ROUTINE FLOWCHART
'NORMAL' SLEW FUNCTION
(ASSEMBLER CODE)

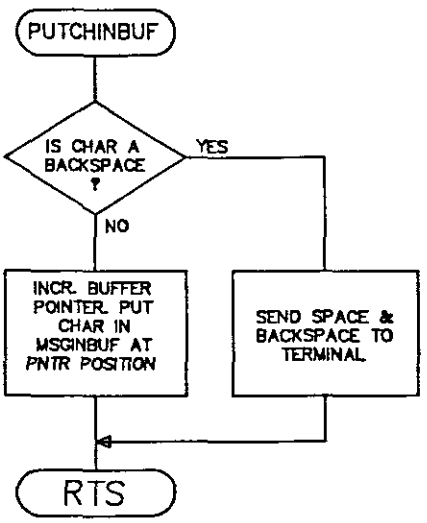


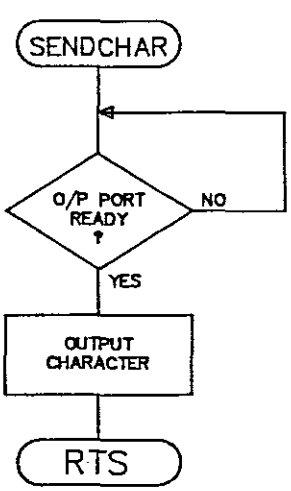
FAST INTERRUPT ROUTINE FLOWCHART
'INITIAL' SLEW
(ASSEMBLER CODE)

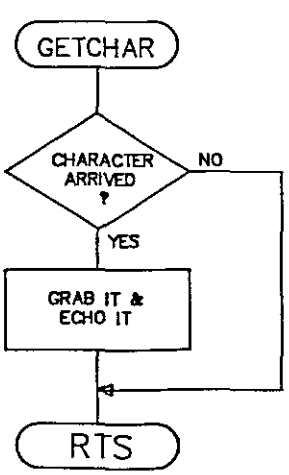


VDU DRIVER FLOWCHART

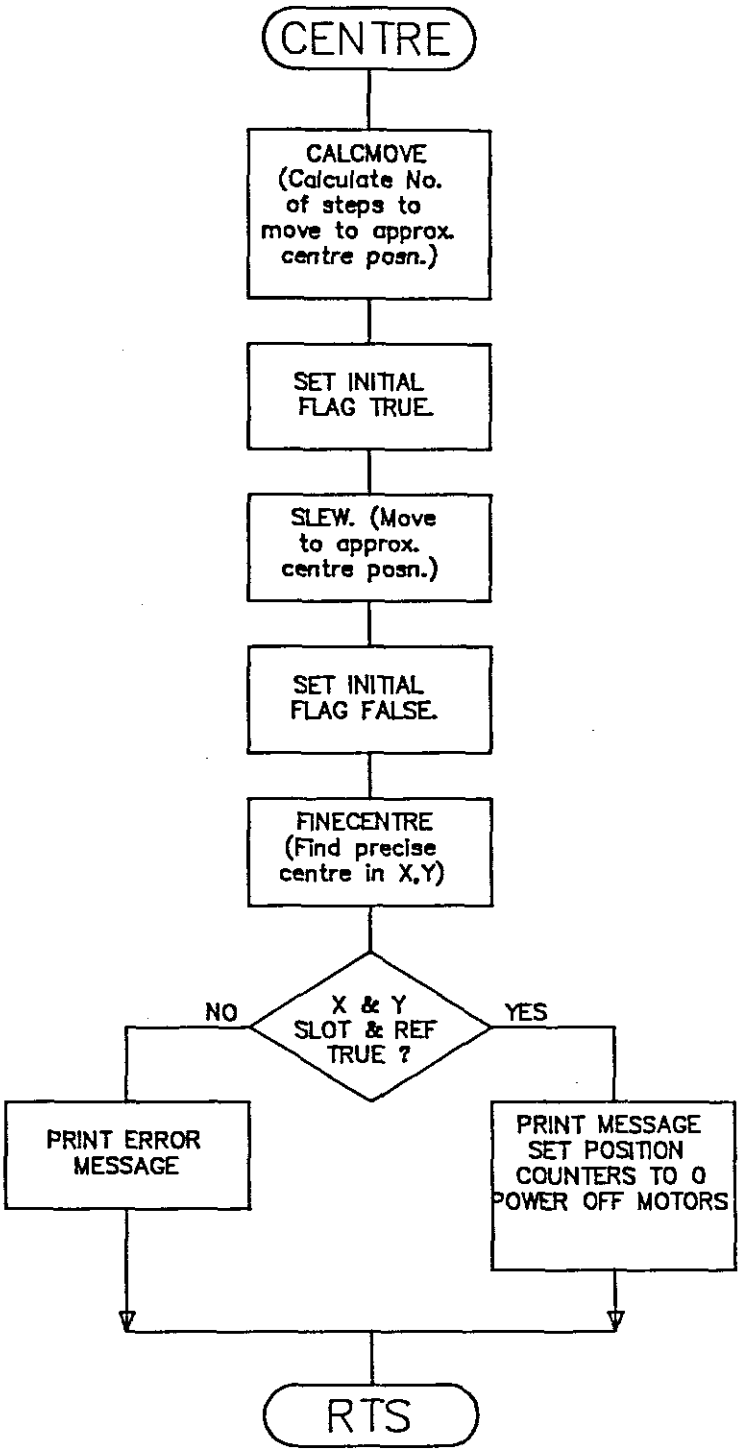




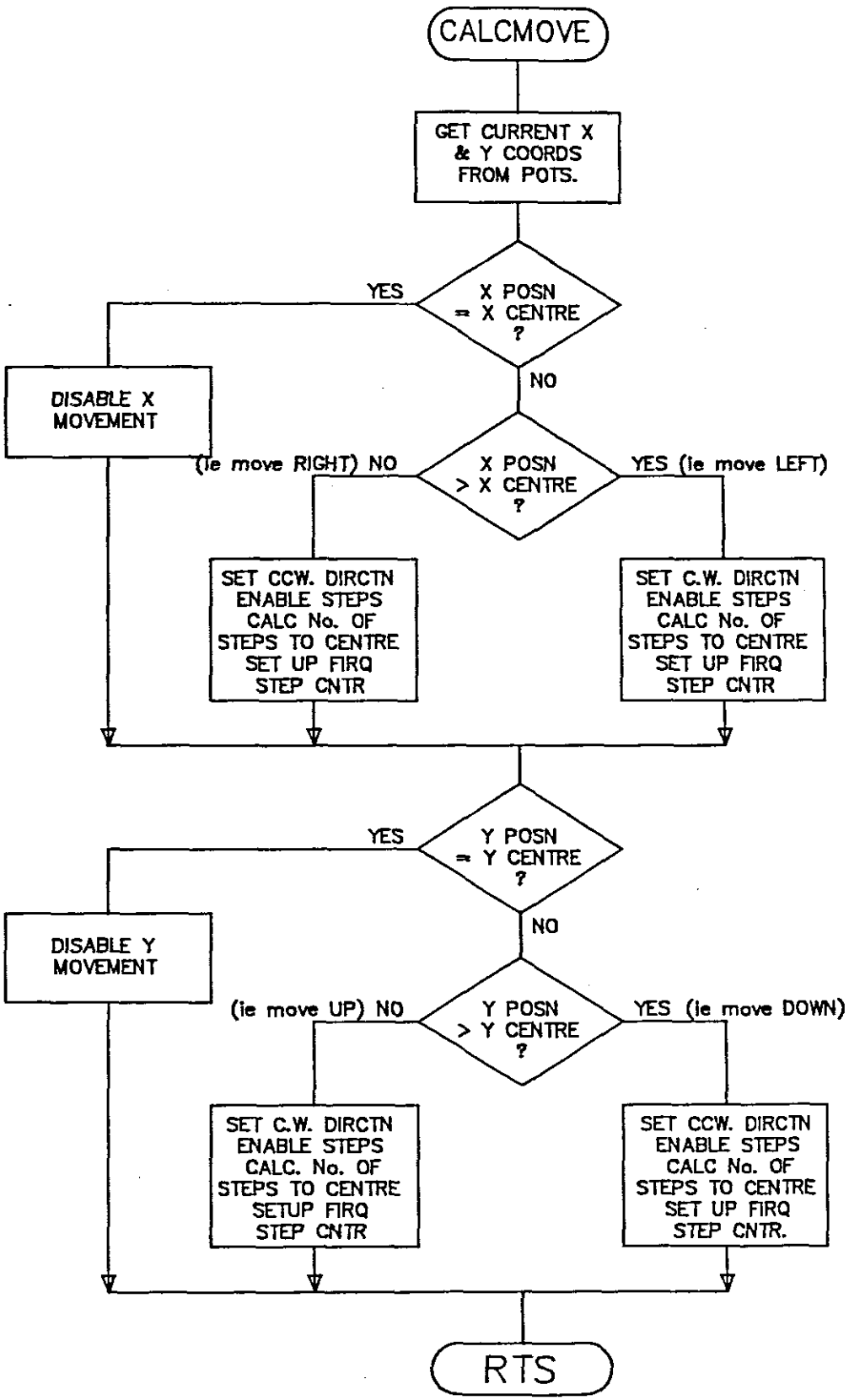




CENTRE FLOWCHART



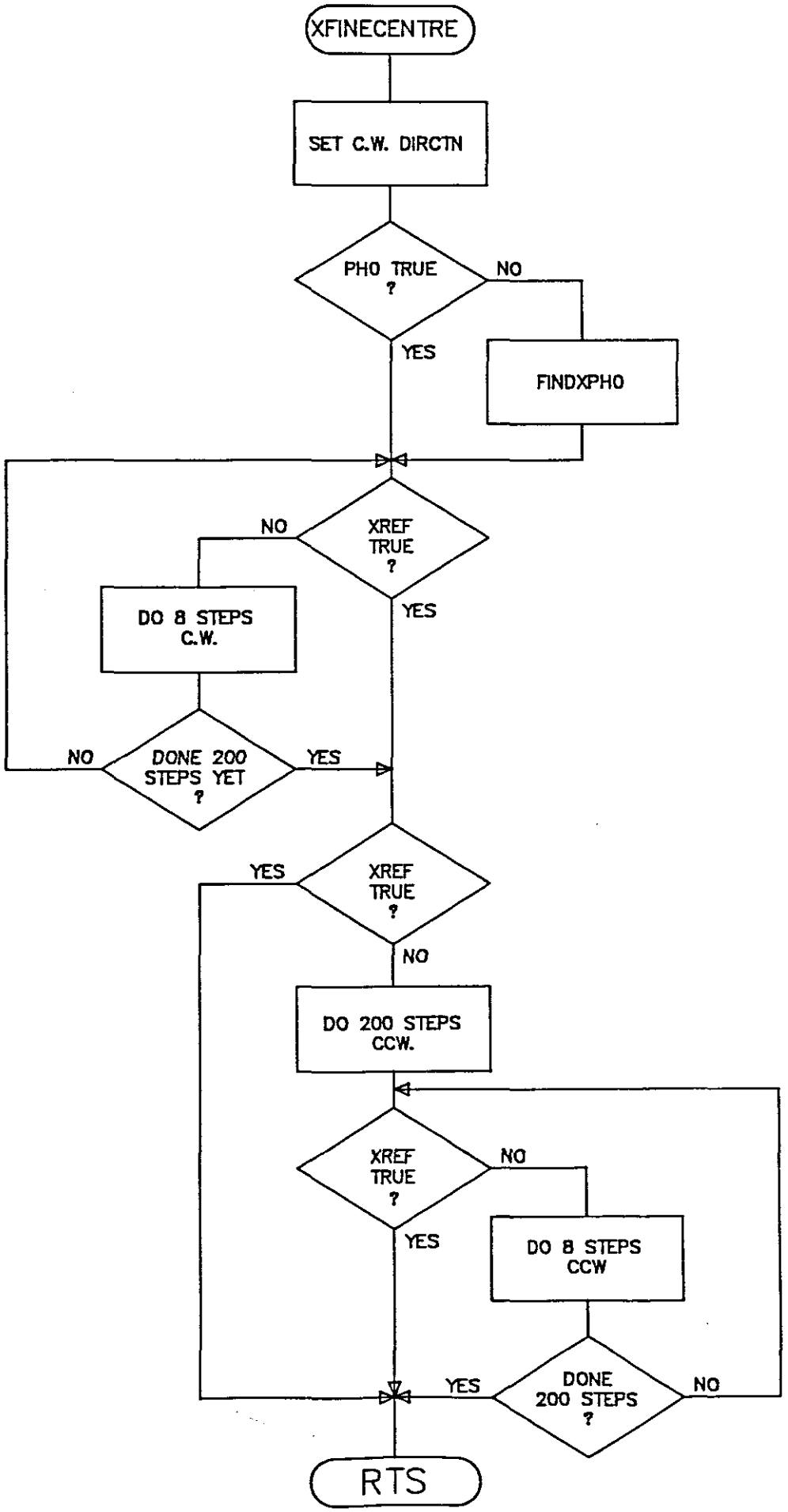
CALCMOVE FLOWCHART
SUB–PROCEDURE OF 'CENTRE'



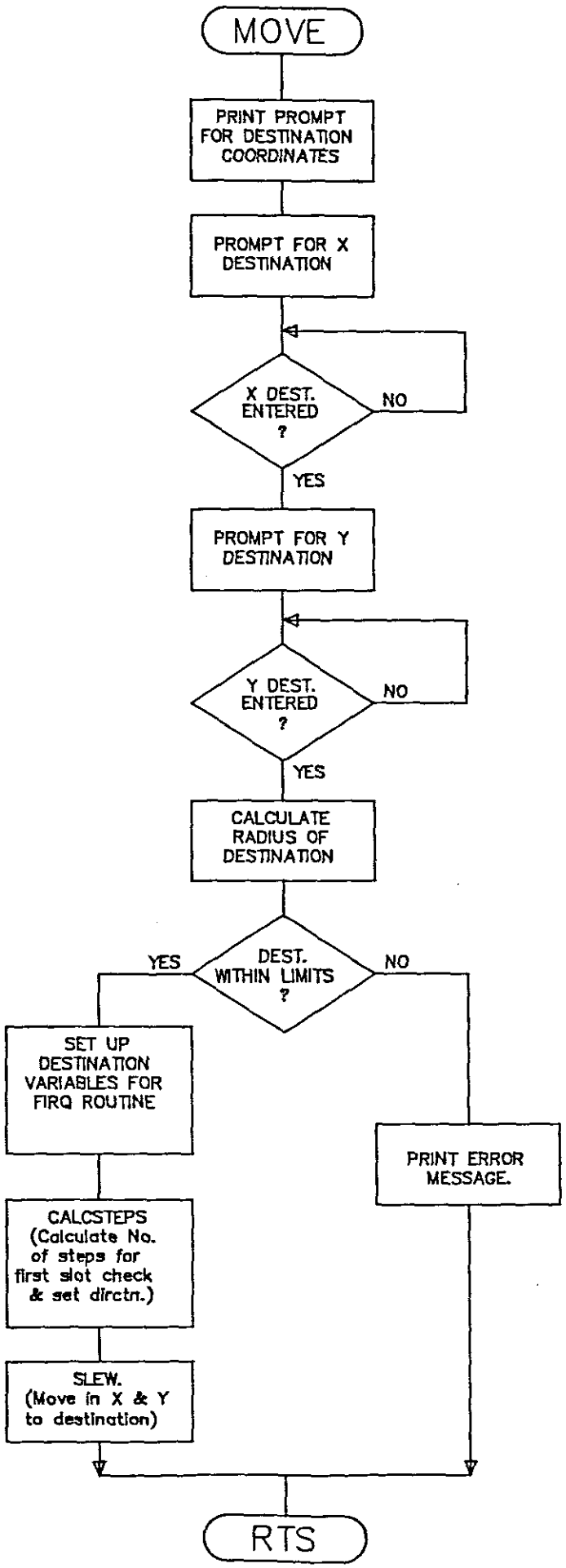
FINECENTRE FLOWCHART

SUB–PROCEDURE OF 'CENTRE'

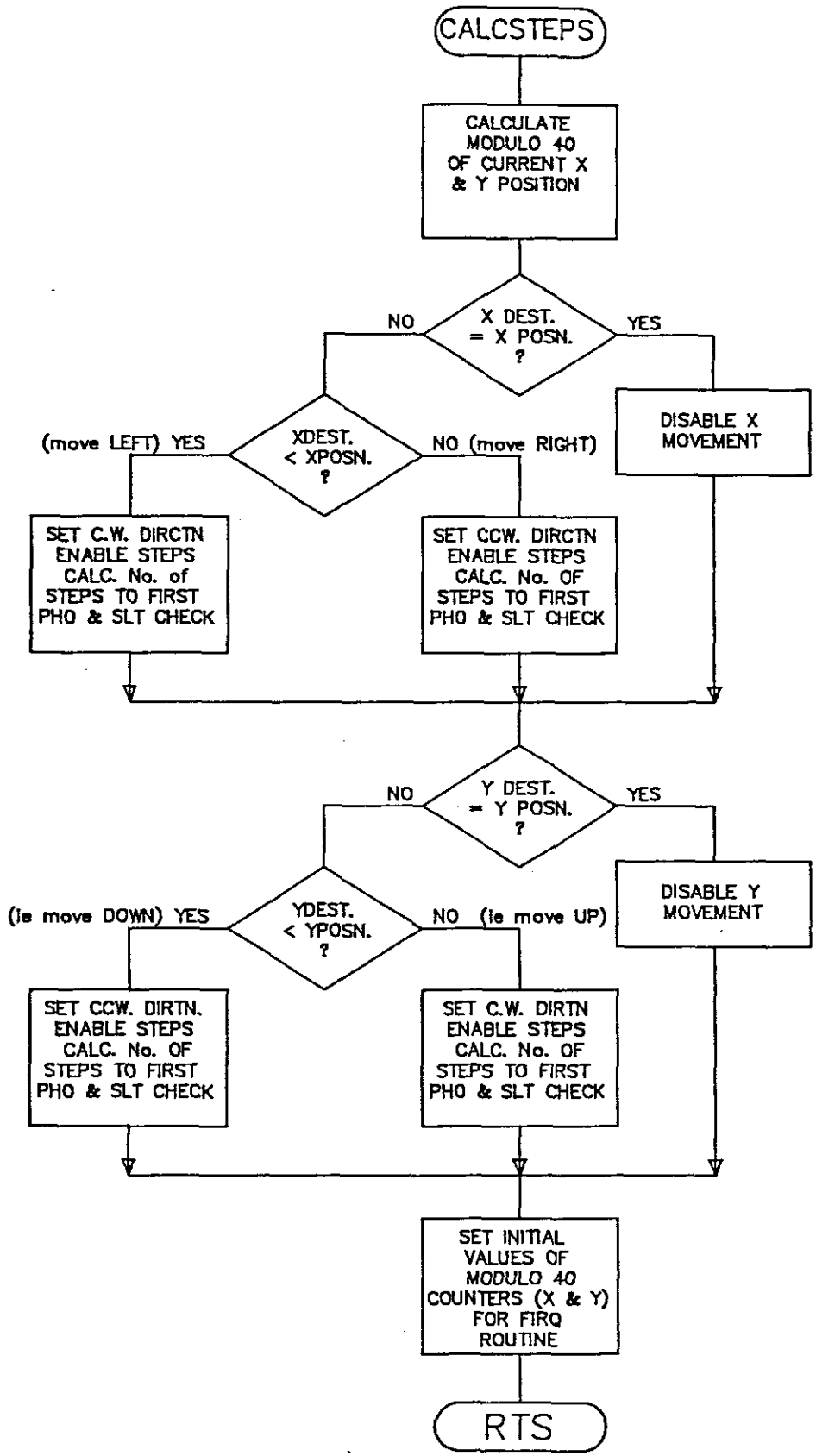
NOTE : XFINECENTRE SHOWN
YFINECENTRE IS IDENTICAL



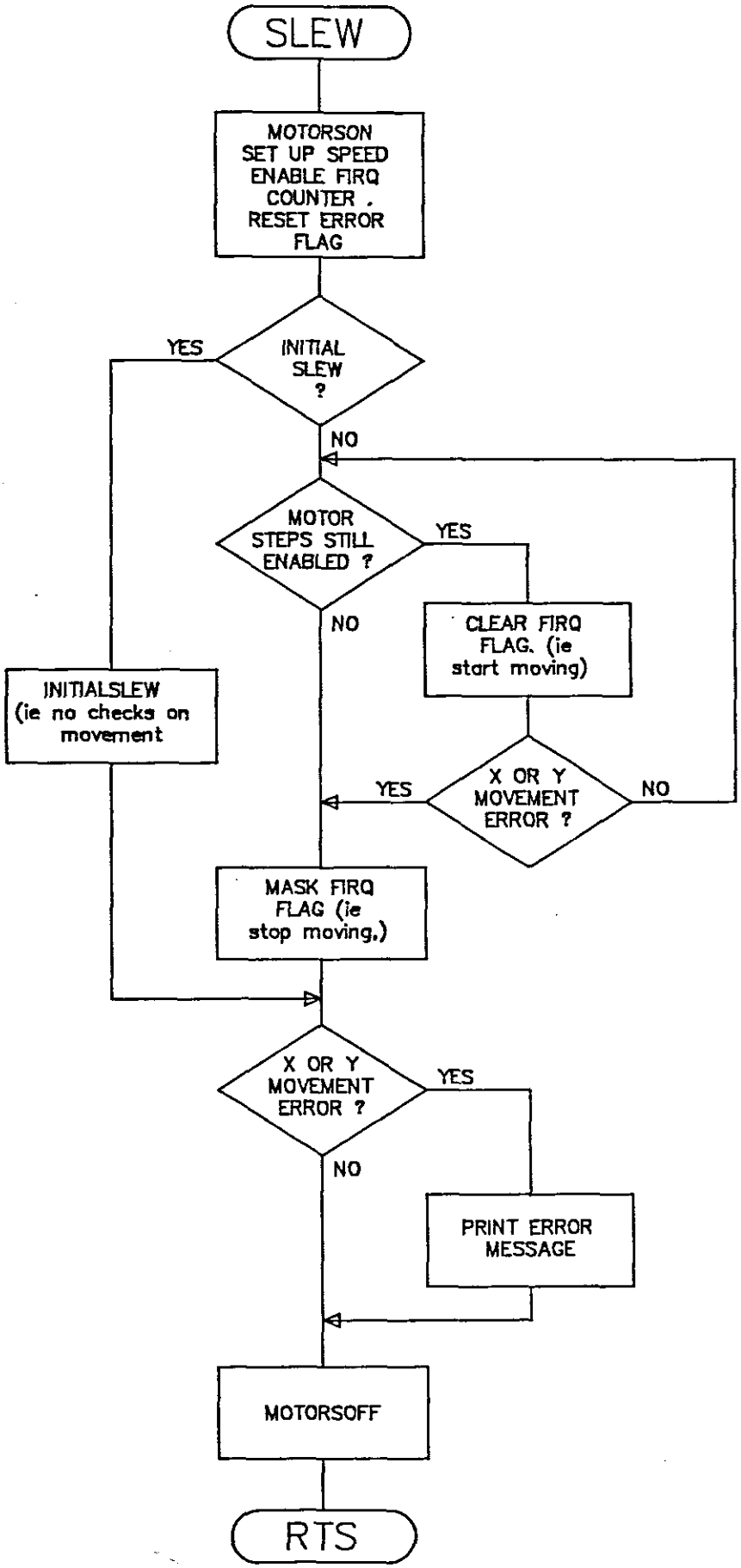
MOVE–to–POSITION FLOWCHART



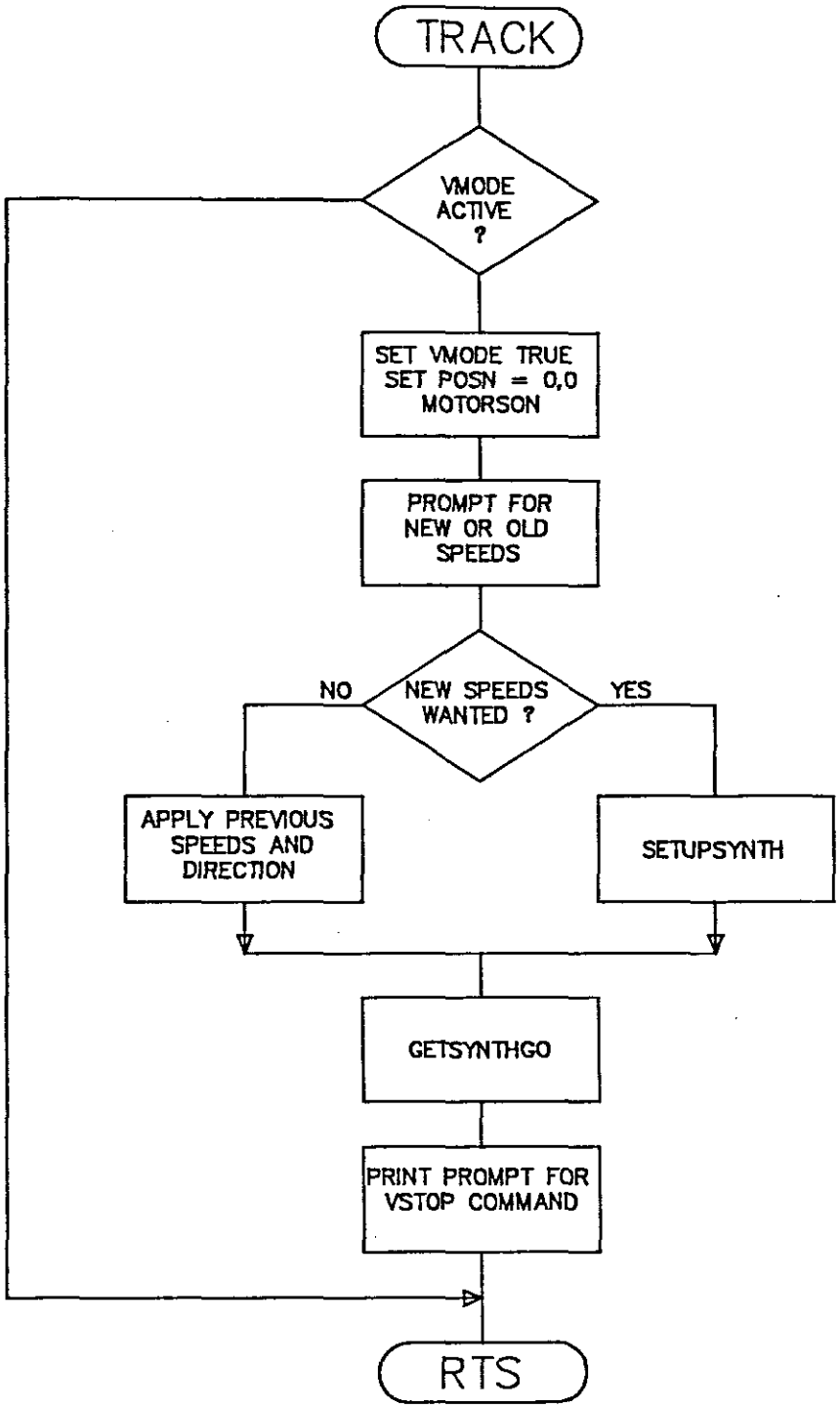
CALCSTEPS FLOWCHART
SUB PROCEDURE OF 'MOVE'



SLEW ROUTINE FLOWCHART

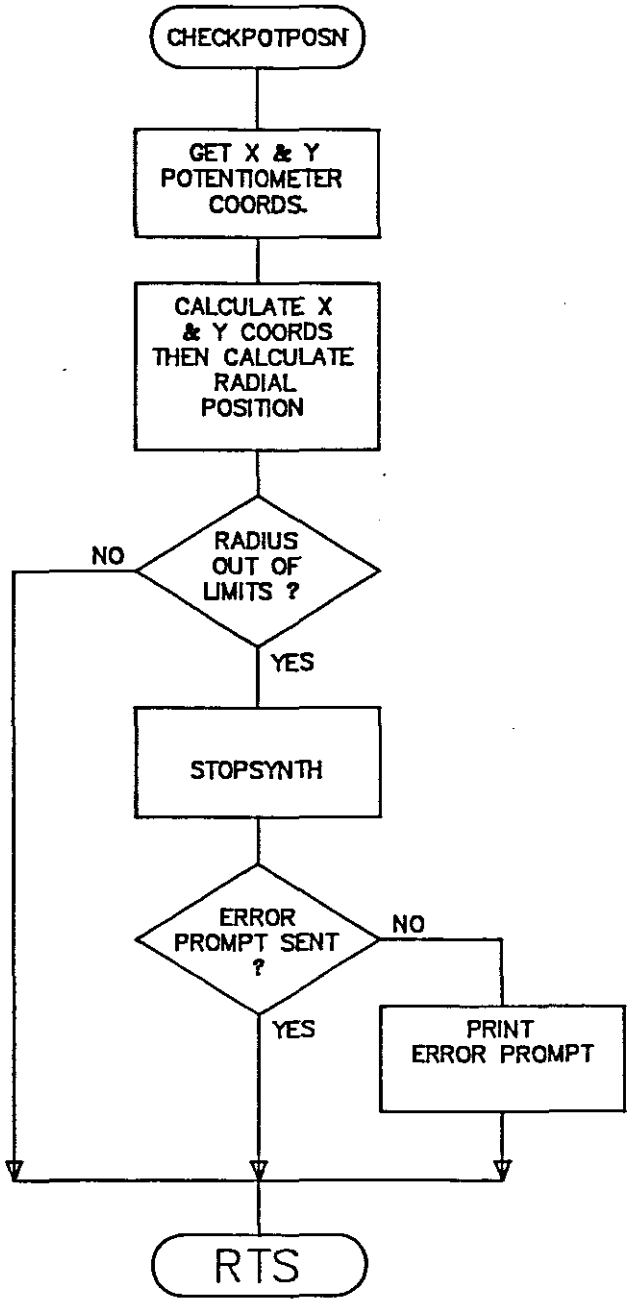


VELOCITY MODE SET UP FLOWCHART

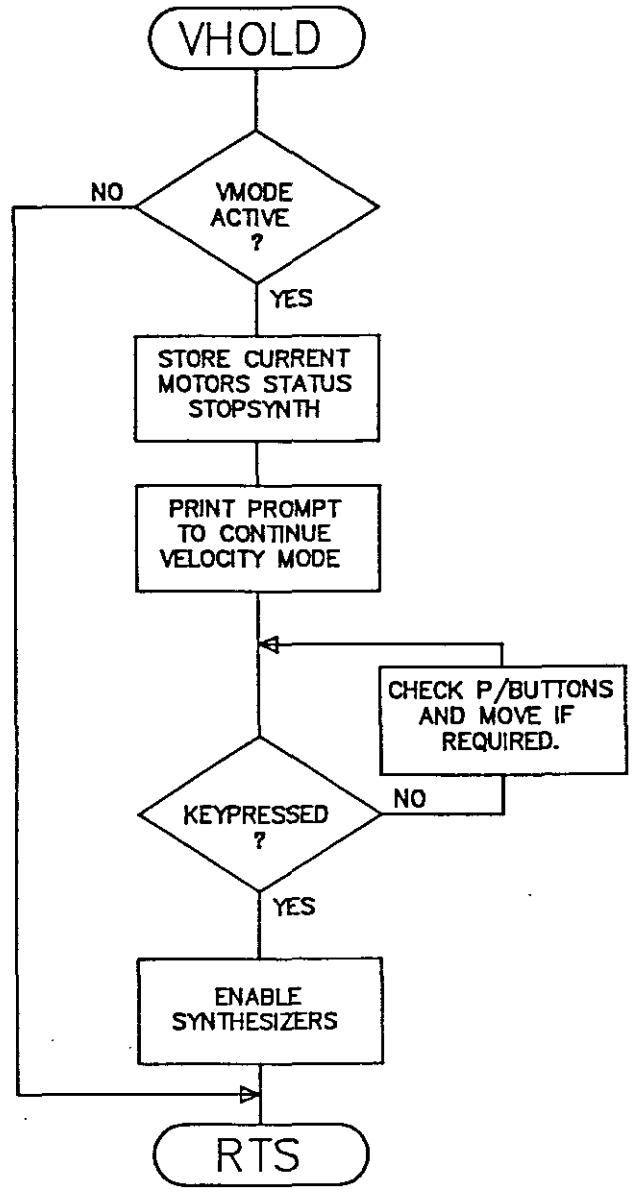


VELOCITY MODE FLOWCHARTS

CHECK PROBE POSITION FROM POTENTIOMETERS

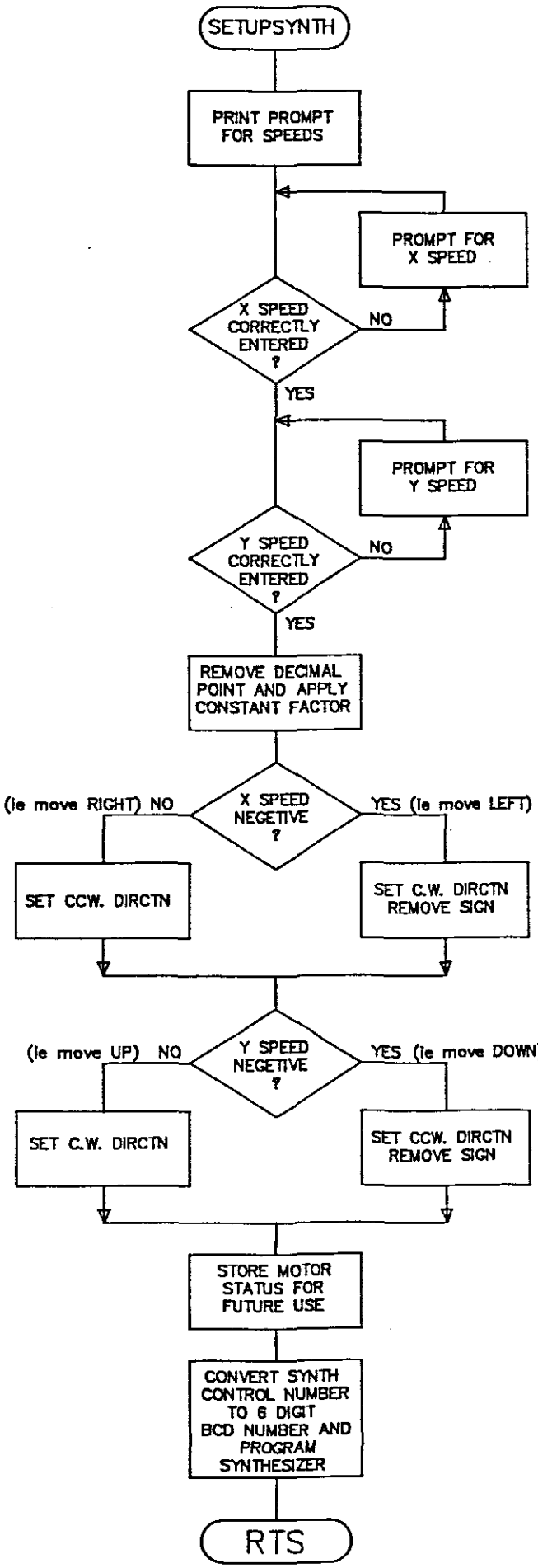


SUSPEND V-MODE OPERATION

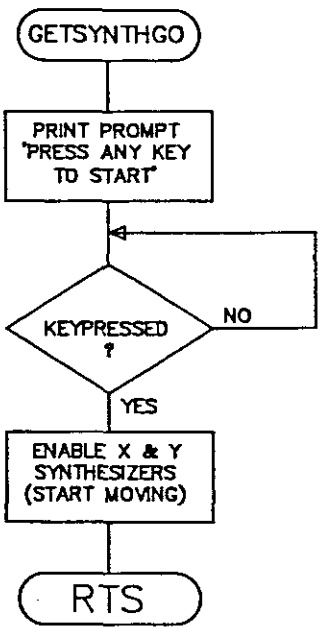


VELOCITY MODE PROCEDURES

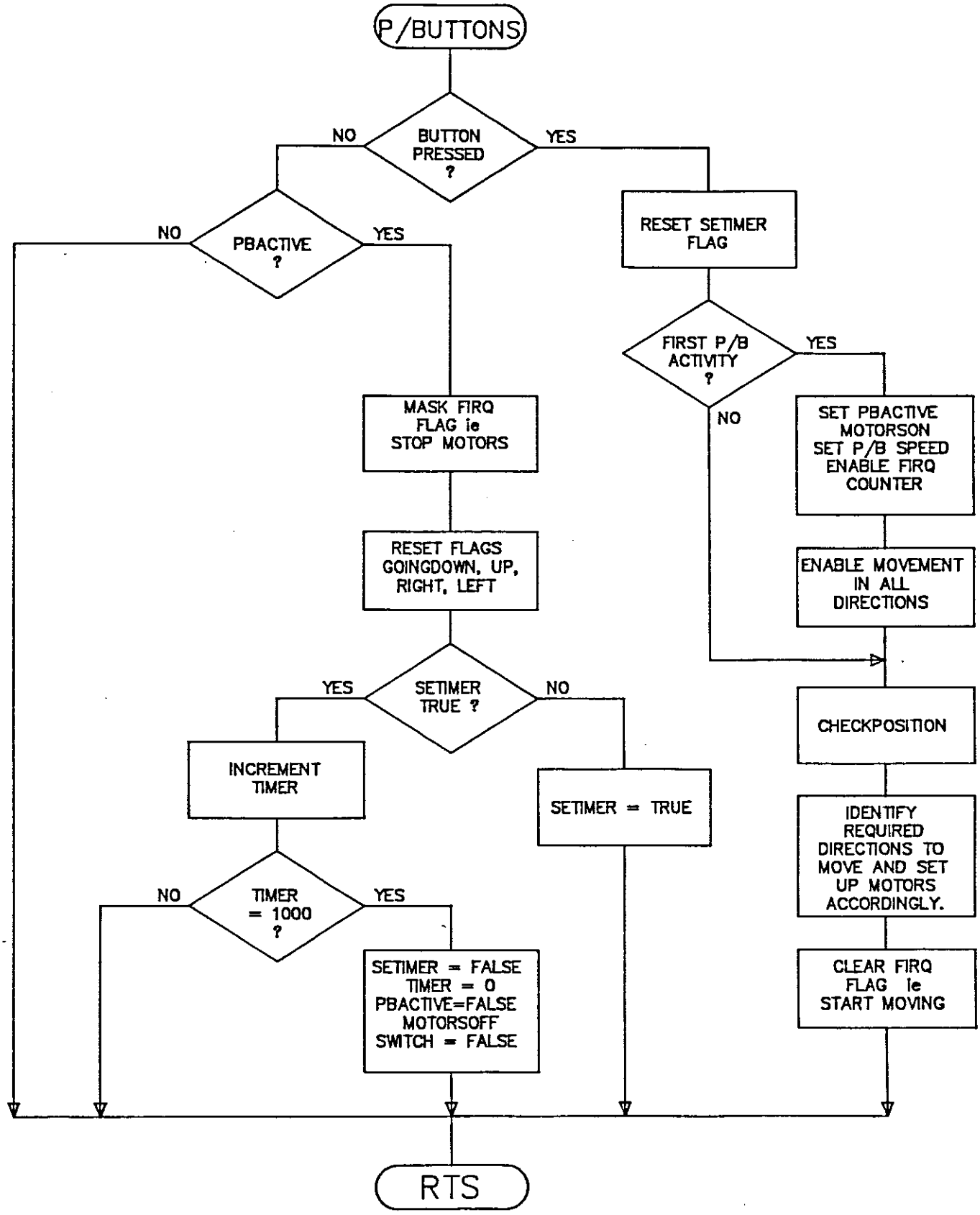
SETTING UP FREQUENCY SYNTHESIZERS



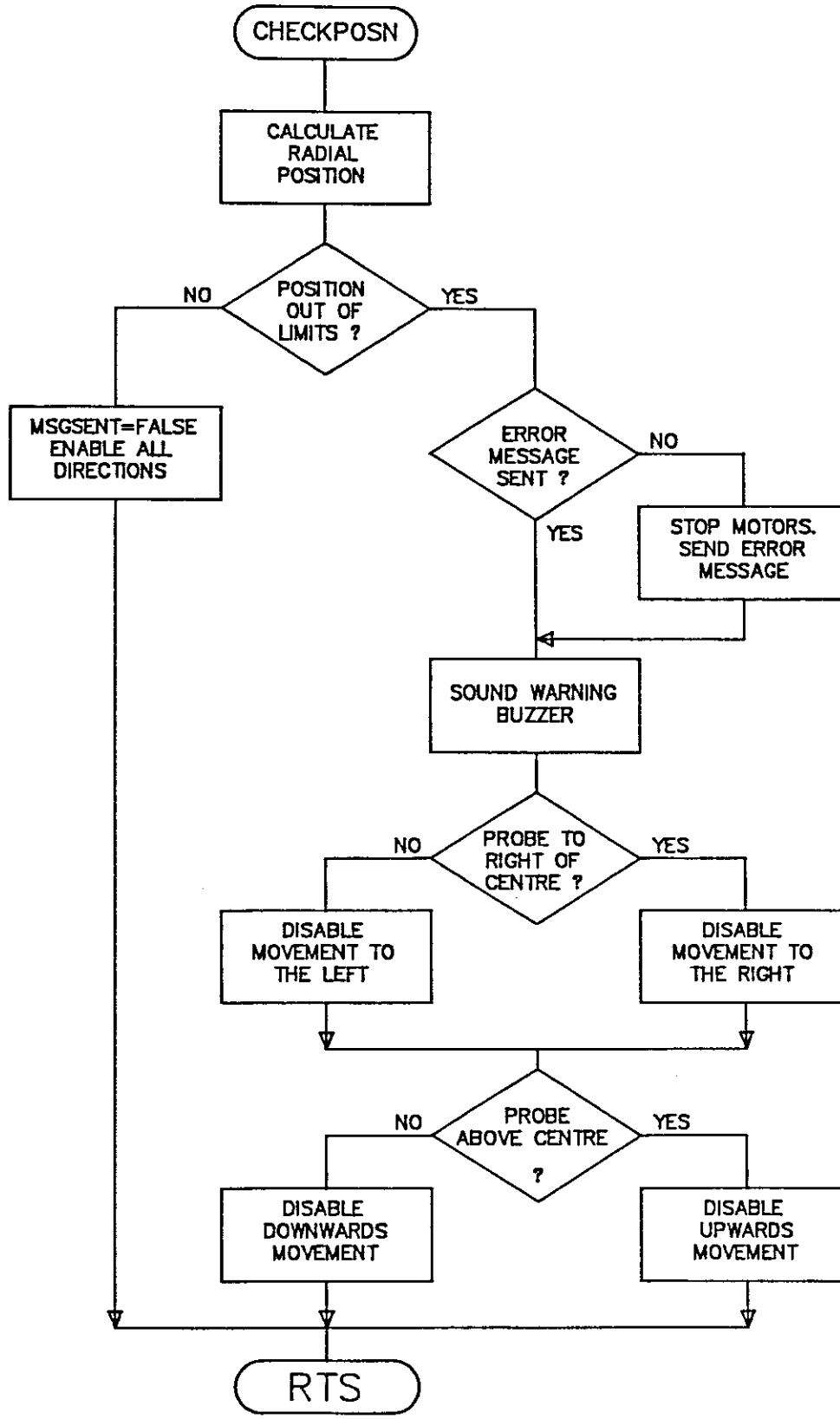
START V-MODE



PUSHBUTTONS FLOWCHART

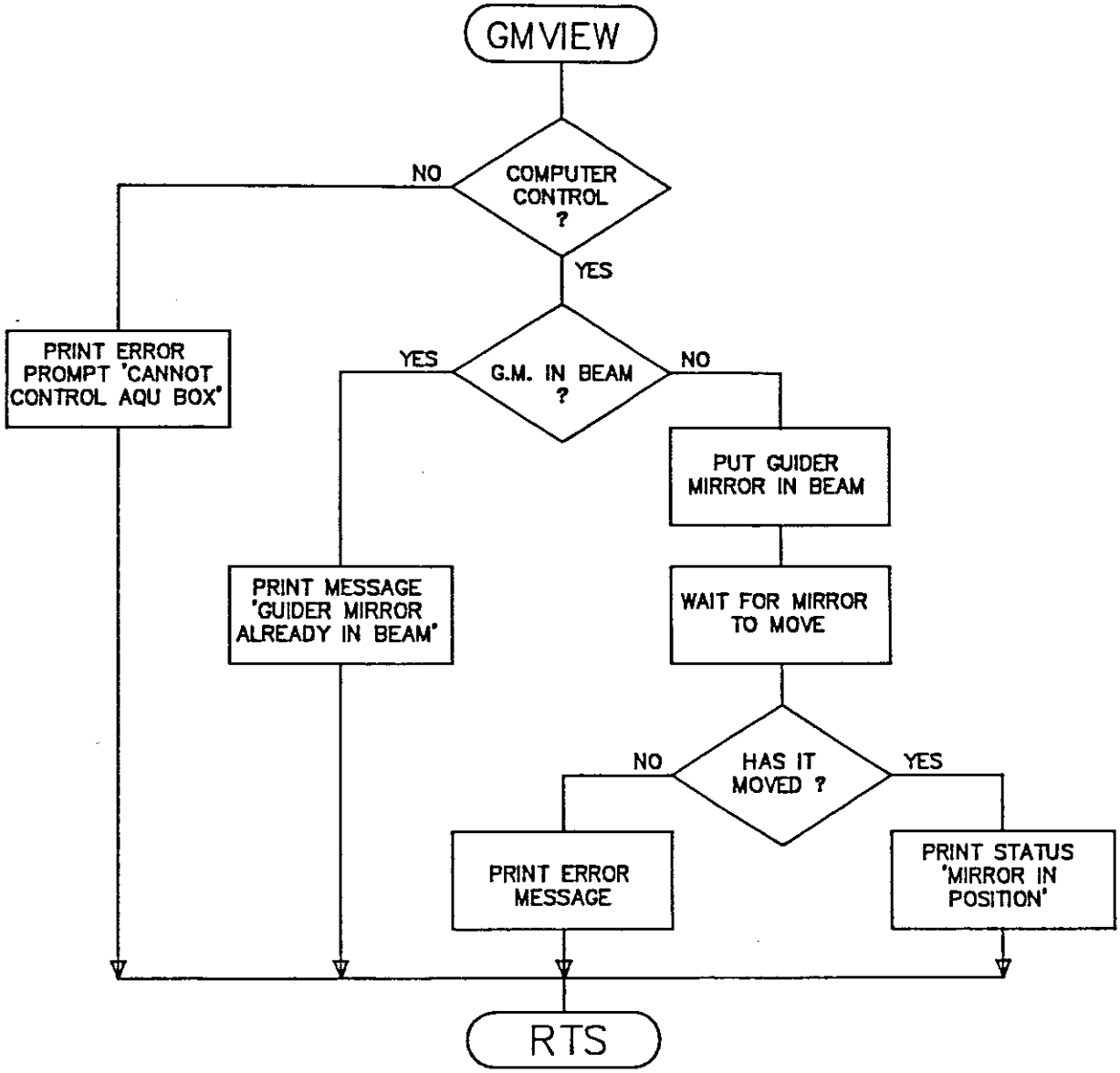


CHECK POSITION FLOWCHART (IN PUSHBUTTON ROUTINE)



MIRROR POSITION CONTROL FLOWCHART

(GMVIEW SHOWN. GMCEN, RVMIN, RVMOUT
HAVE IDENTICAL STRUCTURE)

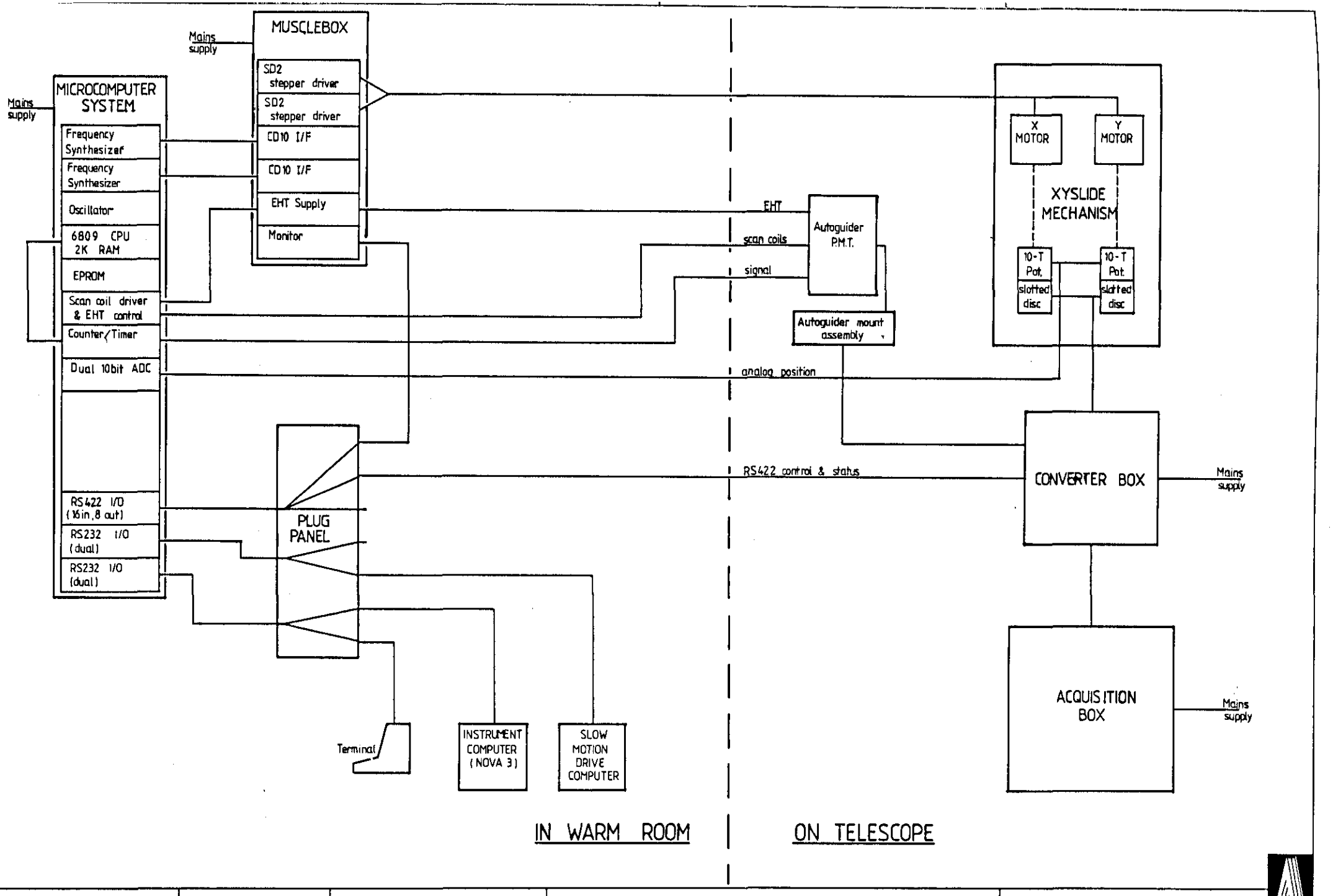


Appendix D.

Automated X–Y Slides
Circuit Diagrams

The following drawings are included, in order of appearance :

Drawing No.	Title
E3–0264	XYslide System Block Diagram.
E3–0283	XYslide Status/Control Wiring Diagram.
E3–0281	XYslide Mechanism Wiring Diagram.
E3–0285	XYslide Musclebox Backplane Wiring Diagram.
E4–0279	XYslide Musclebox Monitor Circuit Diagram.
E3–0162	XYslide Musclebox CD10 I/F Circuit Diagram.
E3–0275	Converter Box Circuit Diagram, Sht. 1.
E3–0276	Converter Box Circuit Diagram, Sht. 2.
E3–0271	SABUS Dual 10–bit ADC Circuit Diagram.
E4–0273	SABUS RS422/TTL I/O Interface Cct. Diagram.
701/1	Basic Electronics G.P. Wirewrap I/F Cct. Diagram.
E3–0125–01	SABUS Oscillator Card Circuit Diagram.
E3–0127	SABUS Frequency Synthesizer Circuit diagram.
E3–0259	Acquisition Box Wiring Diagram.
E3–0260	Acquisition Box Control Circuit diagram.



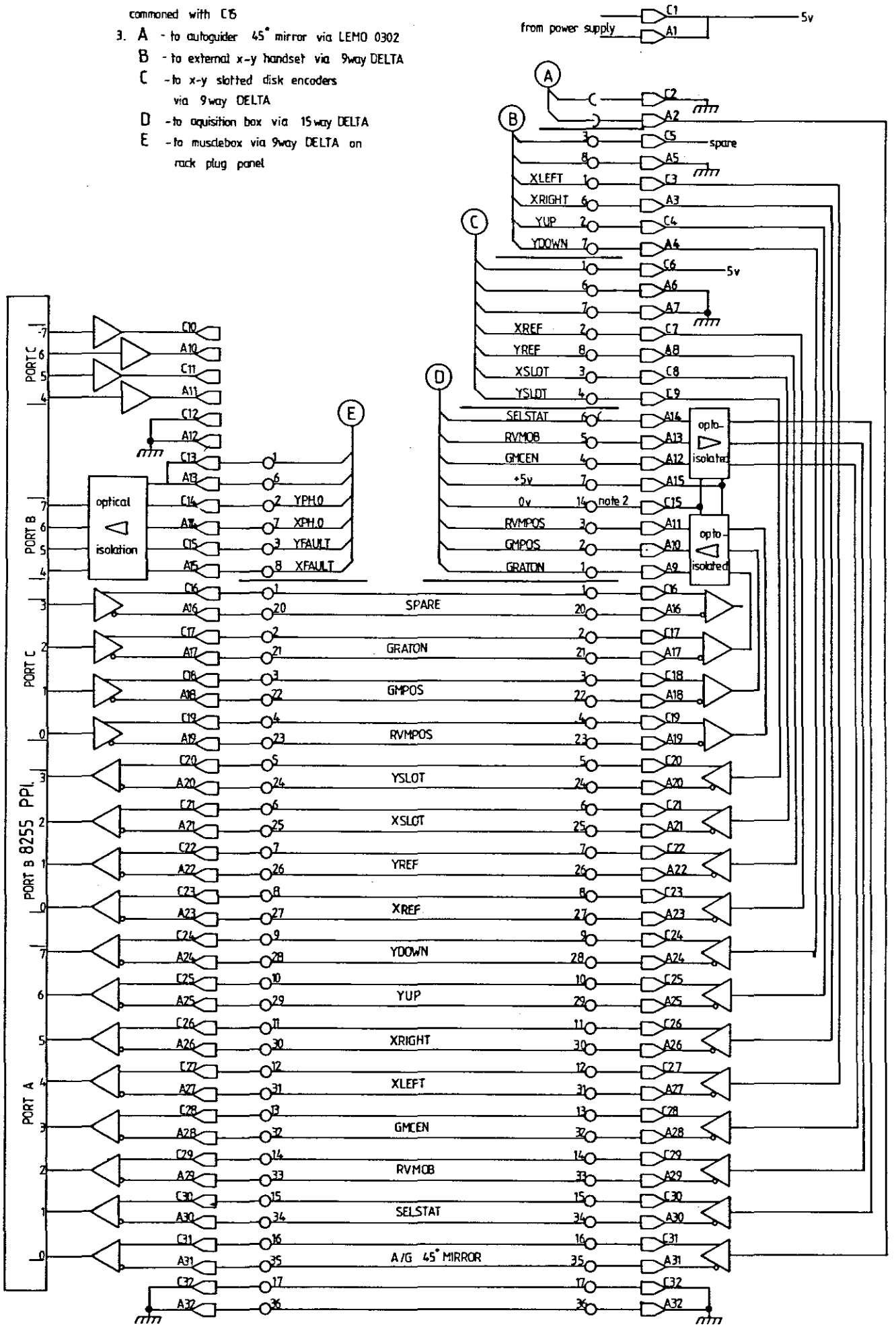
DATE 5-3-85	DRAWN D.C.	DESIGNED D. CARTER	TITLE XYSLIDE AUTOGUIDER. SYSTEM BLOCK DIAGRAM	CSIR-SAAO E3-0264

ozalid 84-325 02-84

NOTES 1. Connections to AC connectors by flat cable

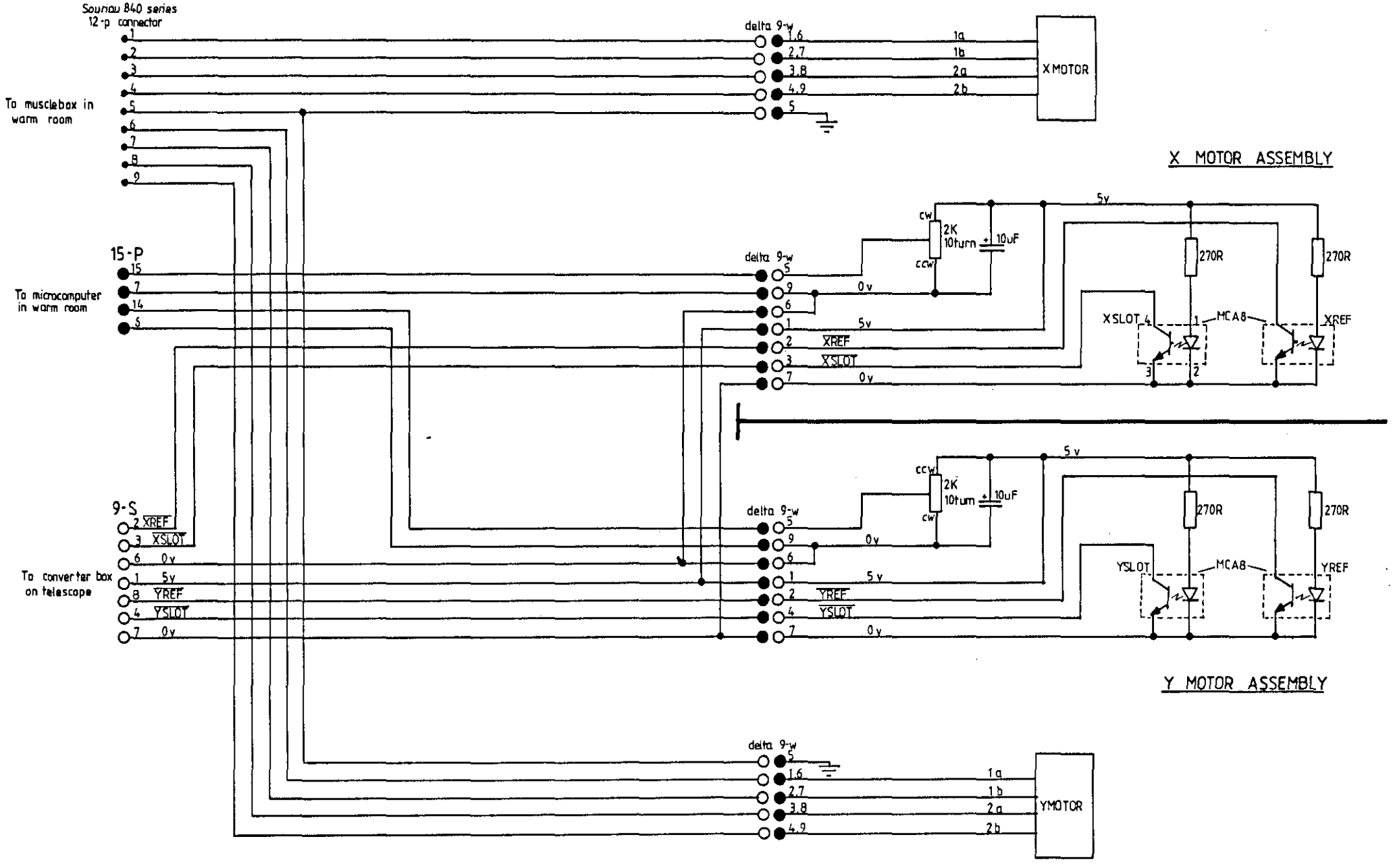
2. Pins C10, C11, C12, C13, & C14 are commoned with C15

3. A - to autoguider 45° mirror via LEMO 0302
   B - to external x-y handset via 9way DELTA
   C - to x-y slotted disk encoders via 9way DELTA
   D - to aquisition box via 15way DELTA
   E - to musclebox via 9way DELTA on rack plug panel



CSIR-SAAO	E3-0283
TITLE	X-Y SLIDES/AUTOGUIDER STATUS WIRING
DESIGNED	D.CARTER
DRAWN	D.C.
DATE	21-3-85

Ibid 84-325 03-84



DATE 17-4-85	DRAWN D.C.	DESIGNED D. CARTER	TITLE XYSLIDE X-Y MECHANISM WIRING DIAGRAM	CSIR-SAAO E3-0281

ozalid 84-325 02-84

PL1 Y-AXIS CD10 I/F

PIN No.	TO/FROM	FUNCTION
AC1	PL 2,3,6/AC1 ; P.S	+5v
A20	PL4/C30	ENERGISE
C20	PL4/C20 PL3/C10	FAULT
C22	PL4/C22 ; PL3/C12	PHASE ZERO
AC26	PL4/C26	DIRECTION
AC28	PL4/C28	CLK
AC16,18, 32	PL2,3,4,5/AC16,18 ; PL2,3,6/AC32	0 v

PL2 X-AXIS CD10 I/F

PIN No.	TO/FROM	FUNCTION
AC1	PL1,3/AC1 P.S.	+5v
A20	PL5/C30	ENERGISE
C20	PL5/C20 PL3/C14	FAULT
C22	PL5/C22 PL3/C15	PHASE ZERO
AC26	PL5/C26	DIRECTION
AC28	PL5/C28	CLK
AC16,18,32	PL1,3,4,5/AC16,18 PL1,3,6/AC32	0v

PL3 MONITOR

PIN No	TO/FROM	FUNCTION
AC1	PL1,2/AC1 P.S.	+5v
C10	PL1/C20 PL4/C20	FAULT Y
C12	PL1,4/C22	PHASE ZERO Y
C14	PL2,5/C20	FAULT X
C15	PL2,5/C22	PHASE ZERO X
C2	PL4/AC2	PH2A
C3	PL4/AC4	PH2B
C4	PL4/AC6	PH1B
C5	PL4AC8	PH1A
C6	PL5/AC2	PH2A
C7	PL5/AC4	PH2B
C8	PL5/AC6	PH1B
C9	PL5/AC8	PH1A
AC16,18,32	PL1,2,4,5/AC16,18 PL1,2,6/AC32	0v

SK1

PIN No	FROM	FUNCTN.	
1	PL5/AC8	PH1A	X
2	PL5/AC6	PH1B	X
3	PL5/AC2	PH2A	X
4	PL5/AC4	PH2B	X
5	CHASSIS	GROUND	
6	PL4/AC8	PH1A	Y
7	PL4/AC6	PH1B	Y
8	PL4/AC2	PH2A	Y
9	PL4/AC4	PH2B	Y

SK2 X-MOTOR

PIN No.	FROM	FUNCTN
1,6	PL5/AC8	PH 1A
2,7	PL5/AC6	PH1B
3,8	PL5/AC2	PH 2A
4,9	PL5/AC4	PH 2B
5	CHASSIS	GROUND

PL4 Y-AXIS SD2

PIN No.	TO/FROM	FUNCION
AC2	PL3/C2 SK1/8 SK3/3,8	PH2A
AC4	PL3/C3 SK1/9 SK3/4,9	PH 2B
AC6	PL3/C4 SK1/7 SK3/2,7	PH 1B
AC8	PL3/C5 SK1/6 SK3/1,6	PH 1A
A12	PL5/A12 TR1-18v	LOGIC SUPPLY
A14	PL5/A14 TR1-18v	
C12	PL5/C12 TR1-26v	MOTOR SUPPLY
C14	PL5/C14 TR1-26v	
AC16,18	PL1,2,3,5/AC16,18 PL1,2,3,6/AC32	0 v
C20	PL3/C10 PL1/C20	FAULT
C22	PL3/C12 PL1/C22	PHASE ZERO
C26	PL1/AC26	DIRECTION
C28	PL1/AC28	CLK
C30	PL1/A20	ENERGIZE
A30	PL5/A30	SYNC.

PL5 X-AXIS SD 2

PIN No.	TO/ FROM	FUNCTION
AC2	PL3/C6 SK1/3 SK2/3,8	PH 2A
AC4	PL3/C7 SK1/4 SK2/4,9	PH 2B
AC6	PL3/C8 SK1/2 SK2/2,7	PH 1B
AC8	PL3/C9 SK1/1 SK2/1,6	PH 1A
A12	PL4/A12 TR1- 18v	LOGIC SUPPLY
A14	PL4/A14 TR1- 18v	
C12	PL4/C12 TR1- 18v	MOTOR SUPPLY
C14	PL4/C14 TR1- 26v	
AC16, 18	PL1,2,3,4/AC16,18 PL1,2,3,6/AC32	0v
C20	PL3/C14 PL2/C20	FAULT
C22	PL3/C15 PL2/C22	PHASE ZERO
C26	PL2/AC26	DIRECTION
C28	PL2/AC28	CLK
C30	PL2/AC20	ENERGISE
A30	PL4/A30	SYNC

PL6 E.H.T. SUPPLY

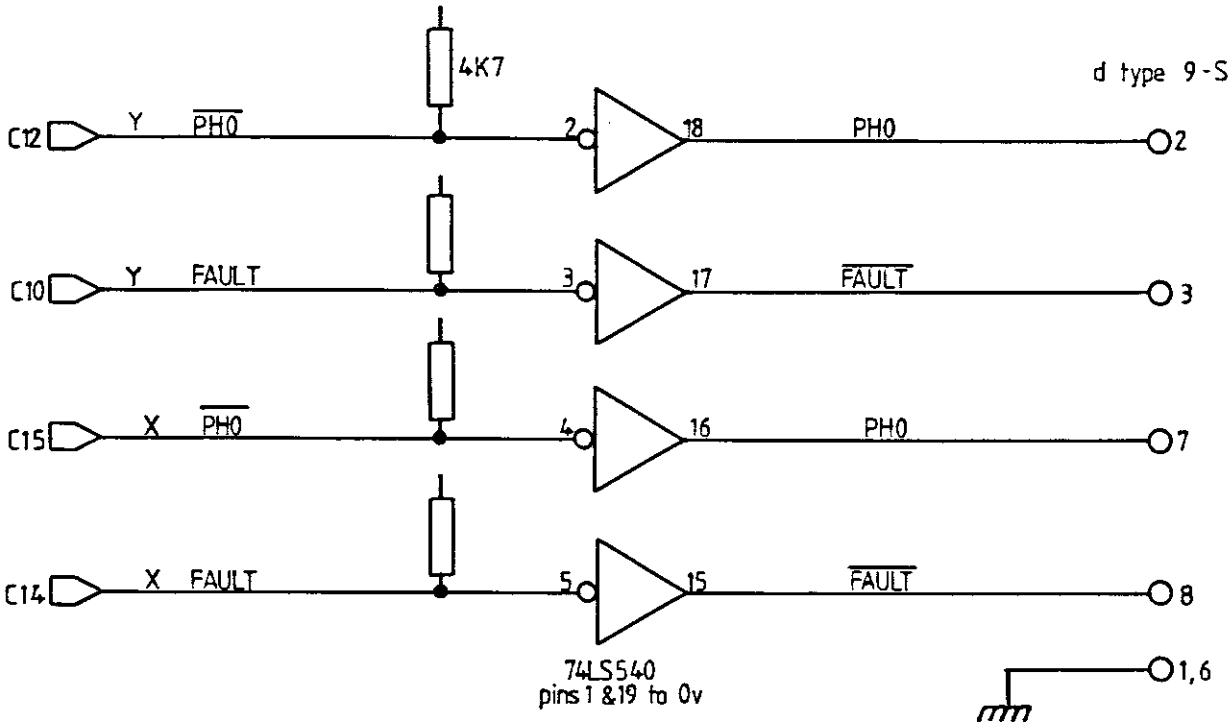
PIN No.	TO/FROM	FUNCTION
AC16	24 v Power supply	+24 v
AC32	PL1,2,3/AC32 PL1,2,3,4,5/AC16,18	0v

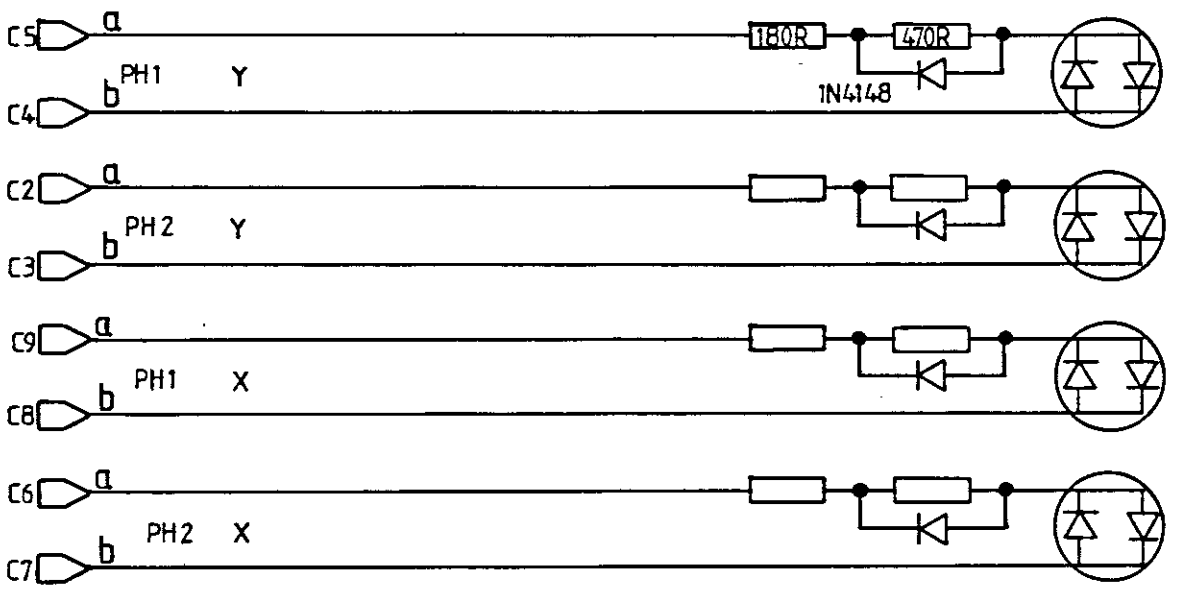
SK 3 Y-MOTOR

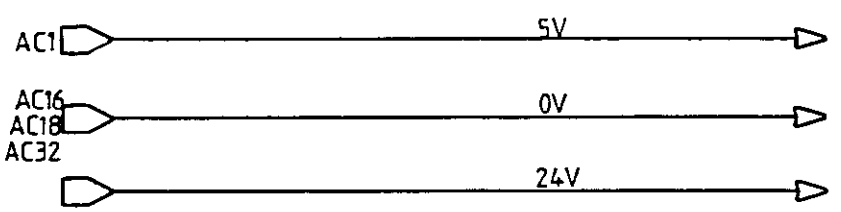
PIN No.	FROM	FUNCTN.
1,6	PL4/AC8	PH1A
2,7	PL4/AC6	PH1B
3,8	PL4/AC2	PH2A
4,9	PL4/AC4	PH2B
5	CHASSIS	GROUND

DATE 5-7-85	DRAWN D.C.	DESIGNED D. CARTER	TITLE XYSLIDE MUSCLEBOX BACKPLANE WIRING	CSIR-SAAO E3-0285

ozalid 84-325 02-84







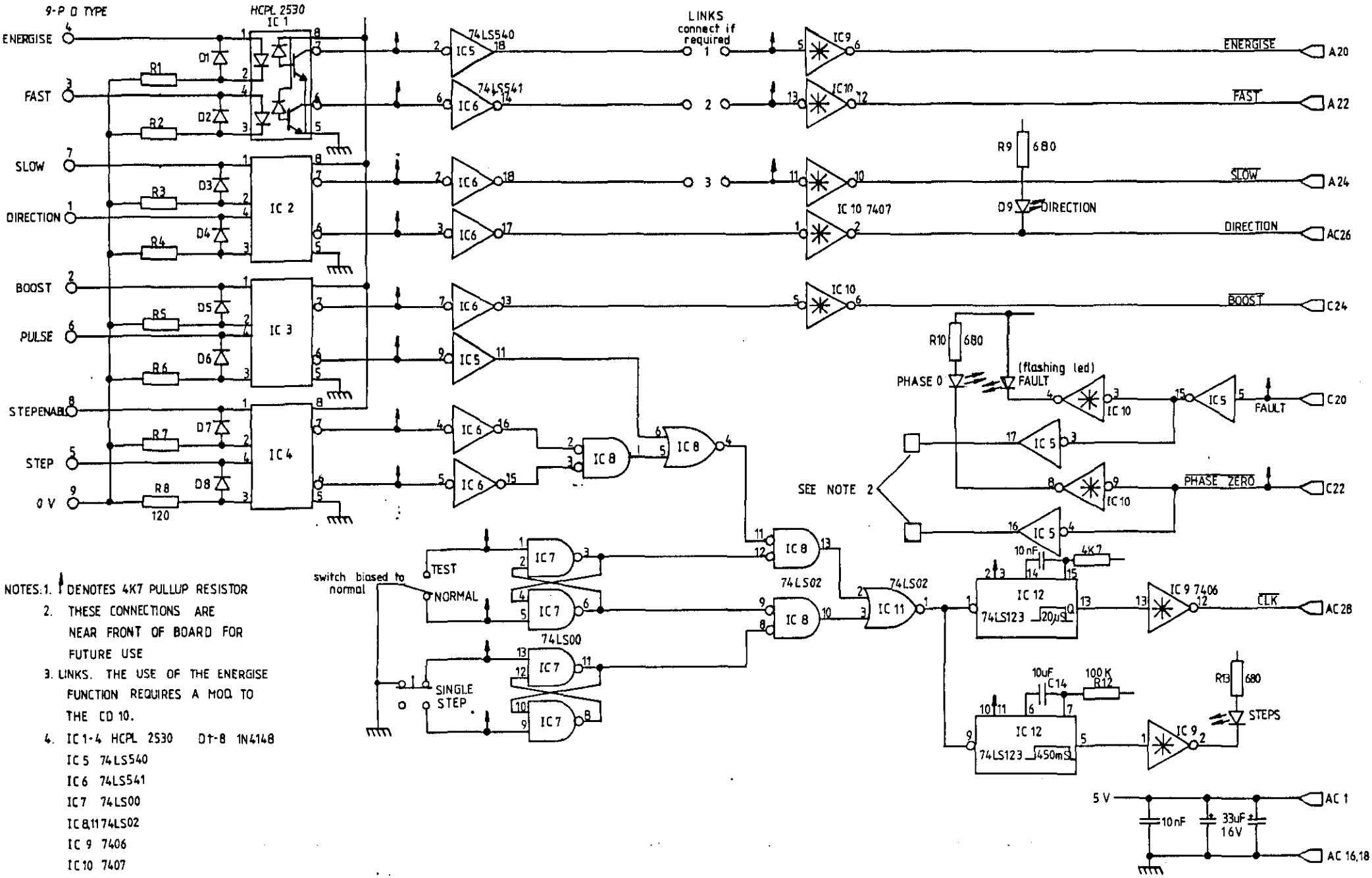
Red/green led's

this lead to B side of phase

2mm test points on front panel

DESIGNED D. CARTER		TITLE XYSLIDES MUSCLEBOX MONITOR CARD	CSIR-SAAO
DATE 21-3-86	DRAWN D.C.		E4-0279

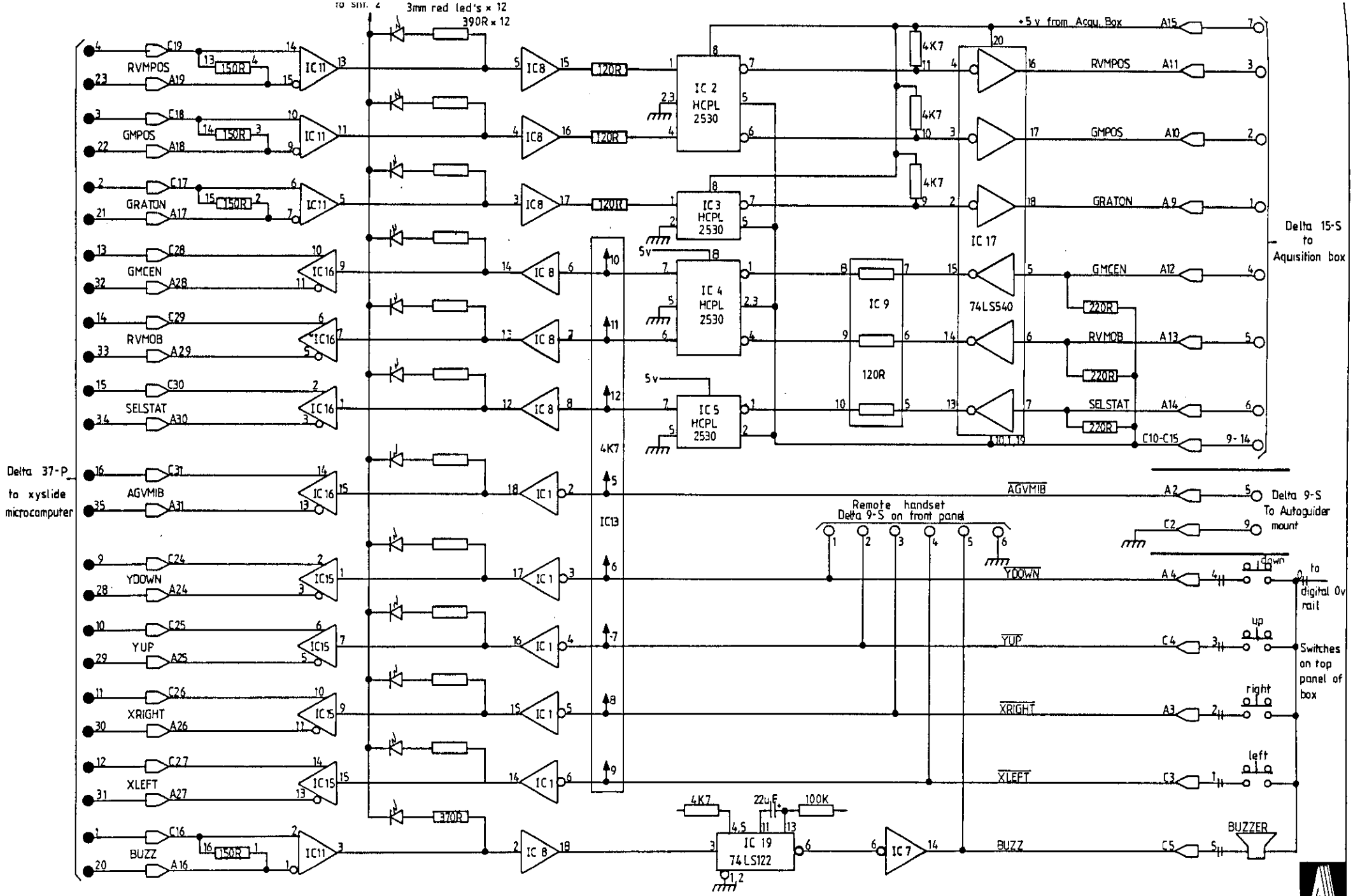
ozalid 84-326 02-84



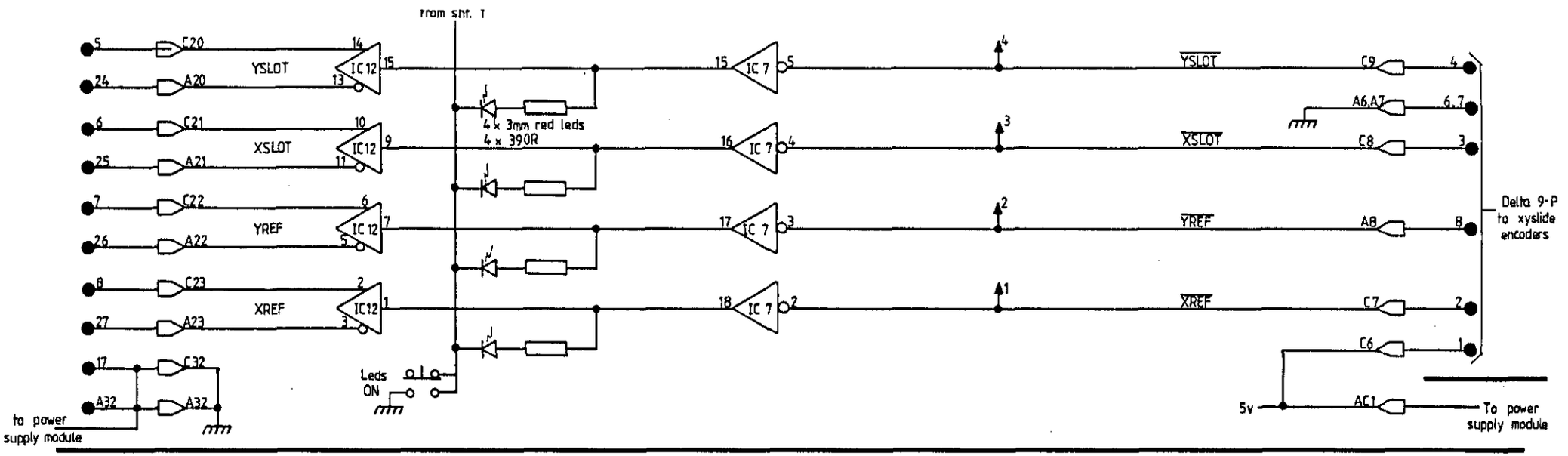
NOTES: 1. ⊓ DENOTES 4K7 PULLUP RESISTOR
2. THESE CONNECTIONS ARE NEAR FRONT OF BOARD FOR FUTURE USE
3. LINKS. THE USE OF THE ENERGISE FUNCTION REQUIRES A MOD TO THE CD 10.
4. IC 1-4 HCPL 2530 D1-8 1N4148
IC 5 74LS540
IC 6 74LS541
IC 7 74LS00
IC 8,11 74LS02
IC 9 7406
IC 10 7407
IC 12 74LS123

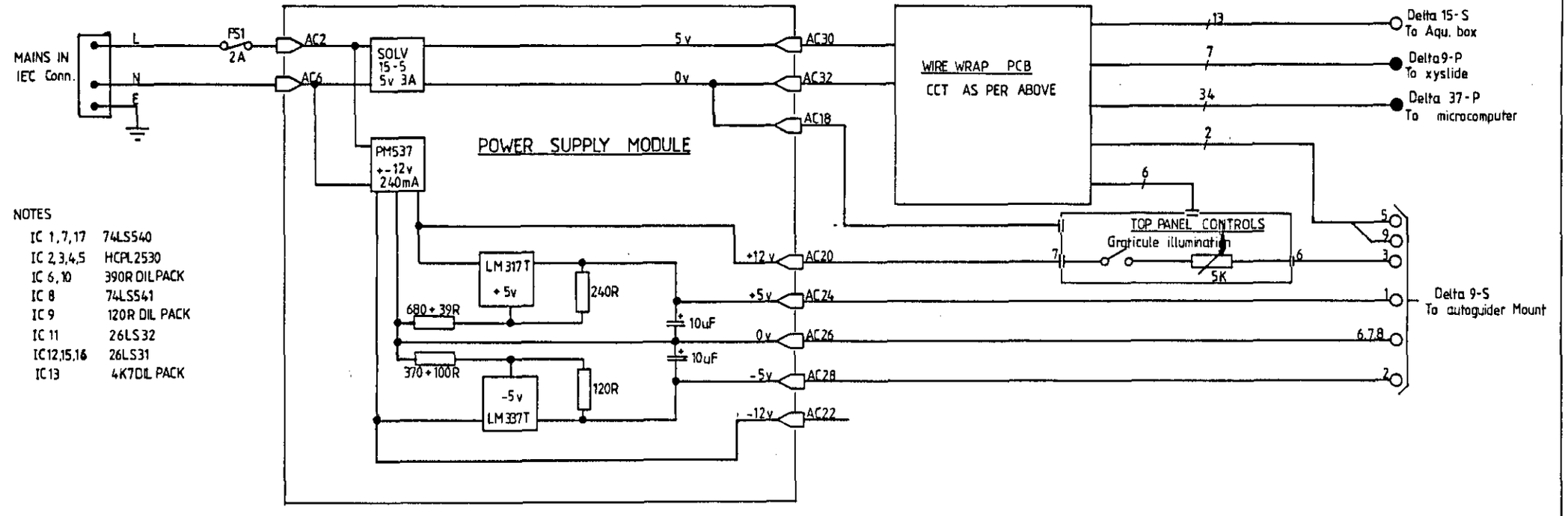
DATE 4-2-83	DRAWN D.C.	DESIGNED D CARTER	TITLE CD 10 INTERFACE	CSIR-SAAO E3-0162

Ozalid 81/1072



DATE 9-3-85	DRAWN D. C.	DESIGNED D. CARTER	TITLE XYSLIDE RS422-TTL CONVERTER BOX. SH. 1 of 2	CSIR-SAAO E3-0275



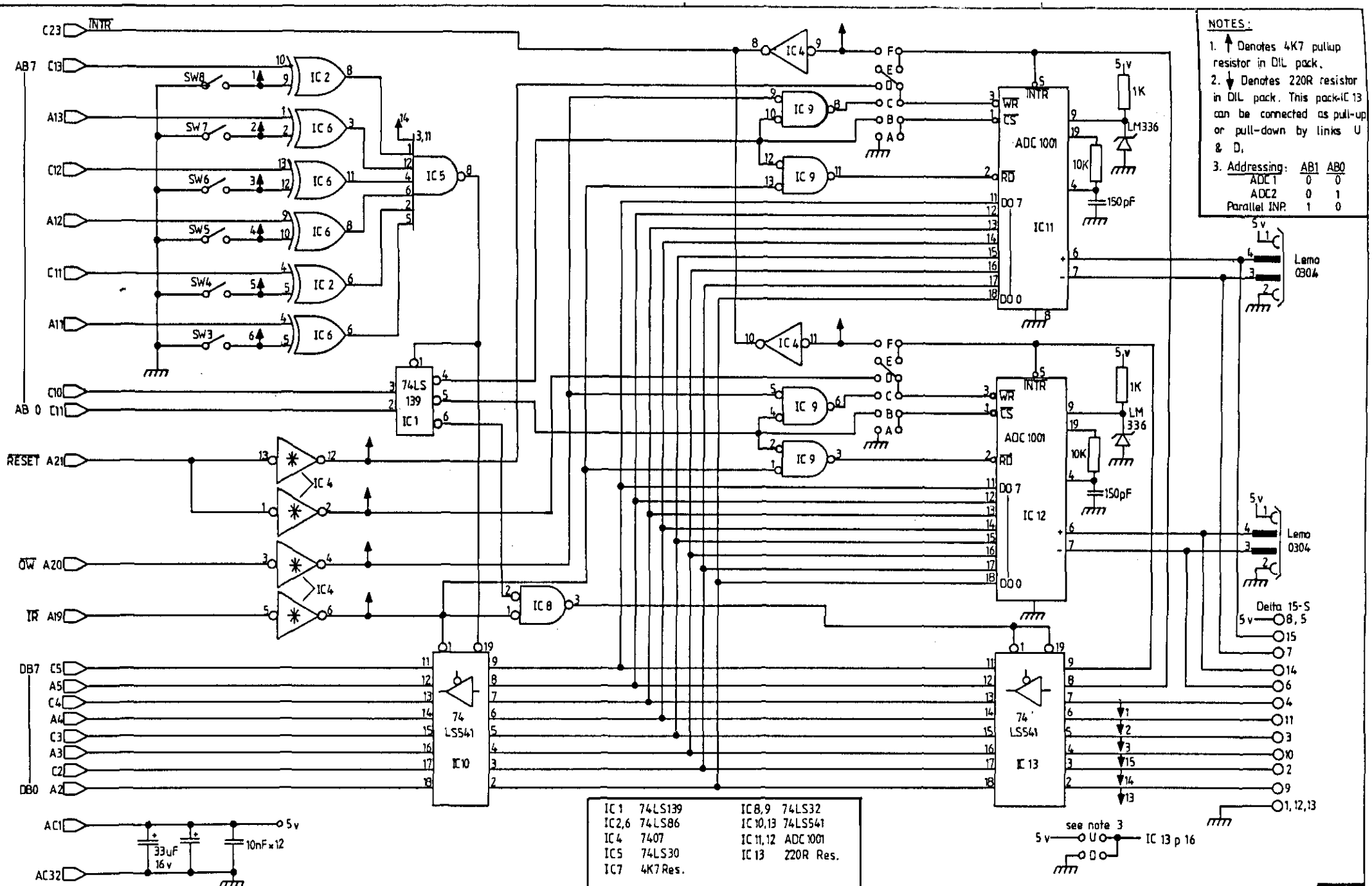


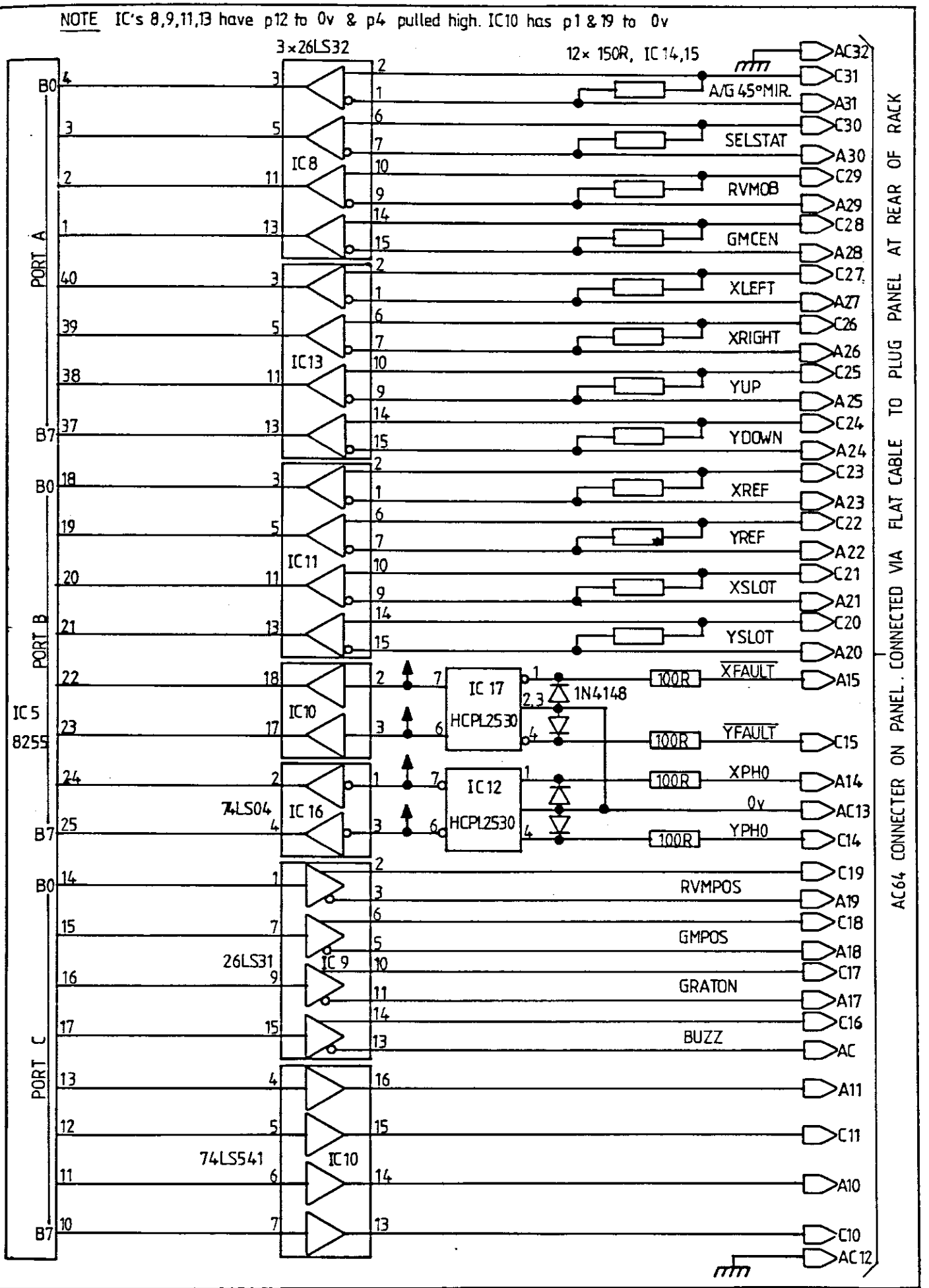
NOTES

IC 1,7,17	74LS540
IC 2,3,4,5	HCPL2530
IC 6,10	390R DIL PACK
IC 8	74LS541
IC 9	120R DIL PACK
IC 11	26LS32
IC12,15,16	26LS31
IC13	4K7DIL PACK

DATE	DRAWN	DESIGNED	TITLE		
12-3-85	D.C.	D. CARTER.	XYSLIDE RS422-TTL CONVERTER BOX	SH. 2 of 2	CSIR-SAAO E3-0276

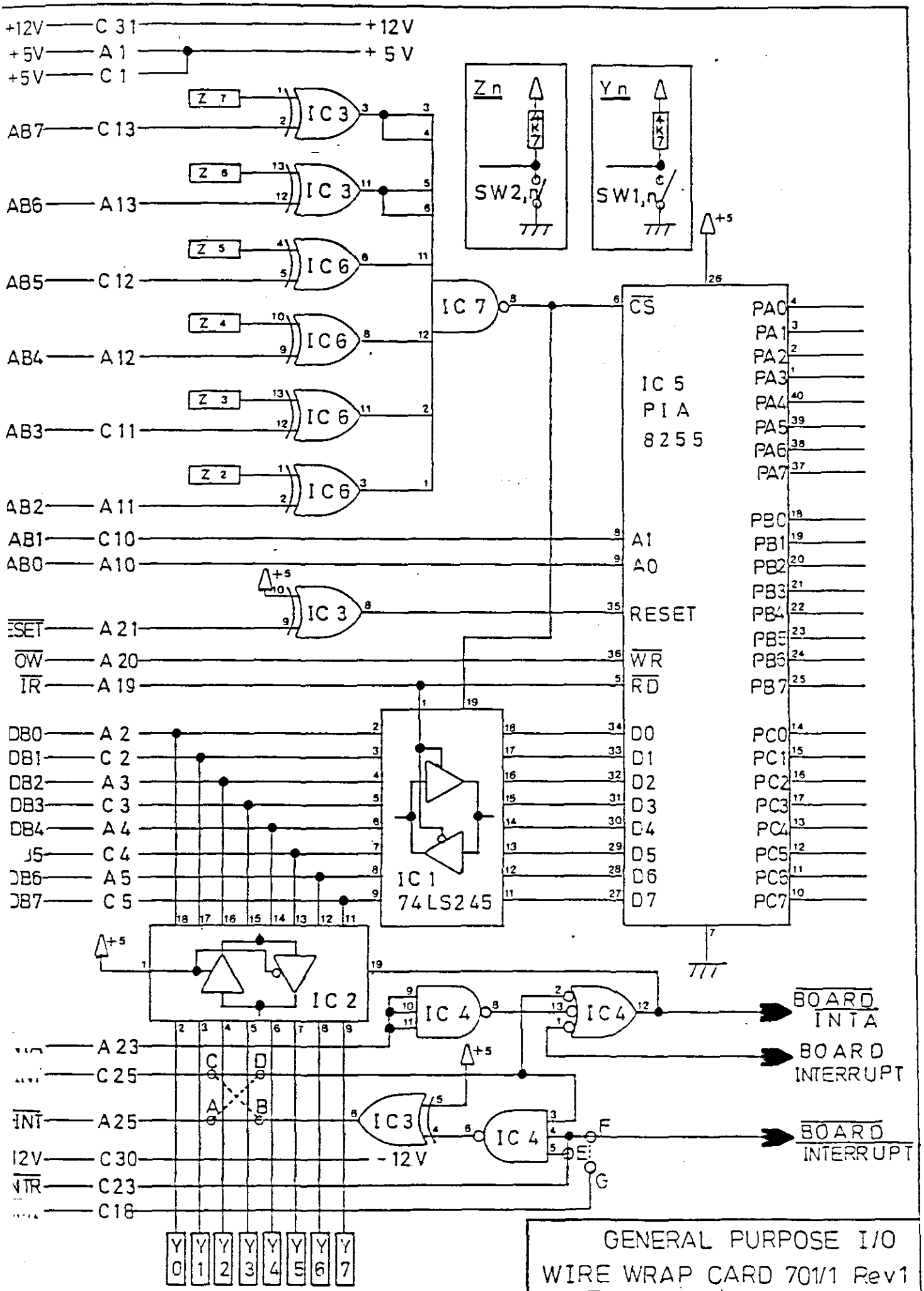
ozalid 84-325 02-84


NOTES:
1. ↑ Denotes 4K7 pullup resistor in DIL pack.
2. ↓ Denotes 220R resistor in DIL pack. This pack-IC 13 can be connected as pull-up or pull-down by links U & D.
3. Addressing: AB1 AB0
ADC1 0 0
ADC2 0 1
Parallel INP. 1 0
INTR
SW8
SW7
SW6
SW5
SW4
SW3
IC 2
IC 6
IC 5
74LS 139 IC 1
IC 4
IC 8
IC 9
ADC 1001
IC 11
IC 12
74 LS541 IC 10
74 LS541 IC 13
1K
10K
150pF
LM336
Lemo 0304
Delta 15-S
33uF 16v
10nF x12
IC 1 74LS139
IC 2,6 74LS86
IC 4 7407
IC 5 74LS30
IC 7 4K7 Res.
IC 8,9 74LS32
IC 10,13 74LS541
IC 11,12 ADC 1001
IC 13 220R Res.
see note 3
IC 13 p 16
DATE 19-3-85
DRAWN D.C.
DESIGNED D. CARTER
TITLE SABUS DUAL A-D CONVERTER (10-bit)
CSIR-SAAO E3-0271



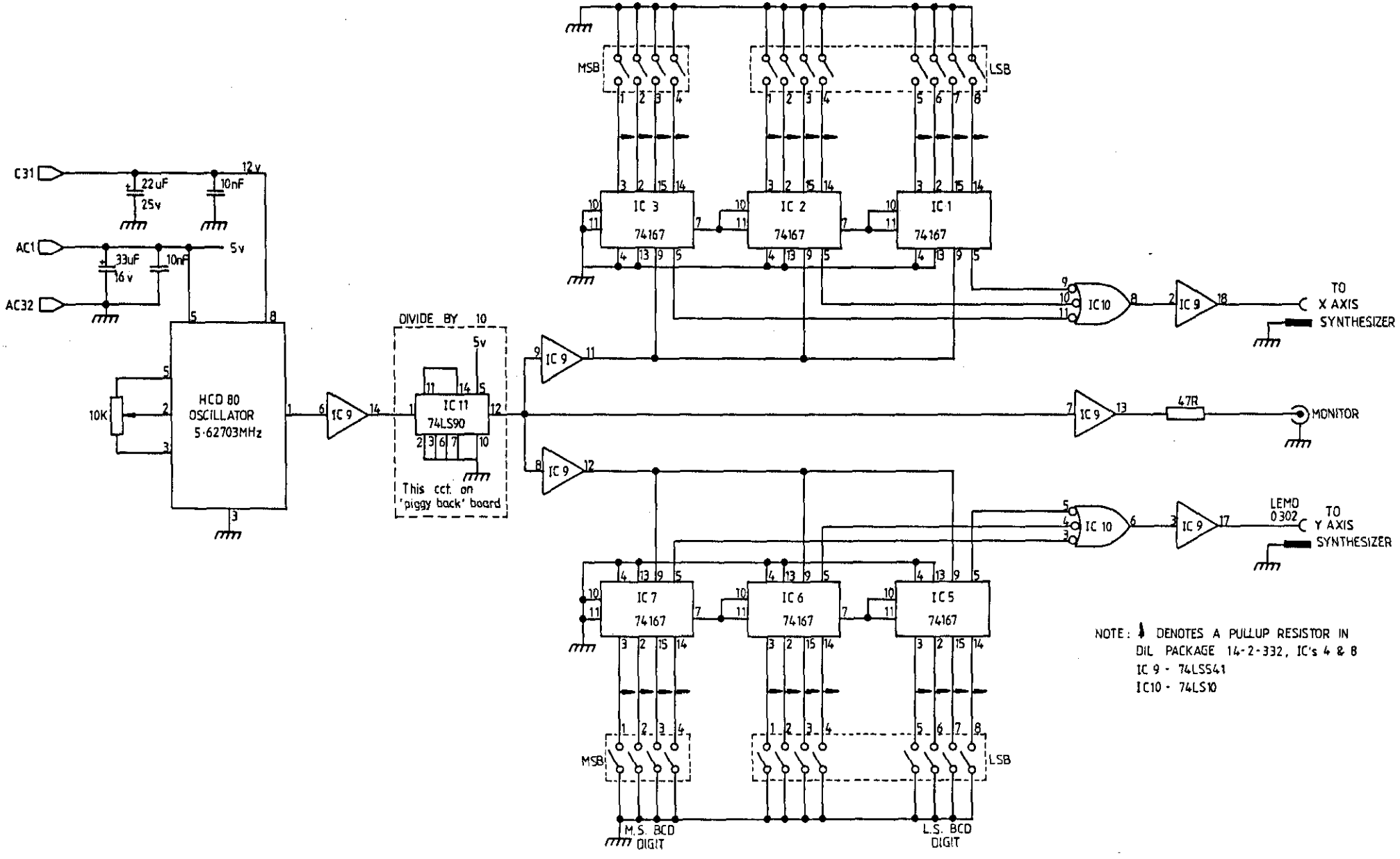
DESIGNED D CARTER	TITLE SABUS RS422 I/O INTERFACE FOR XYSLIDE. BUILT ON BASIC ELECTRONICS GENERAL PURPOSE WIRE WRAP CARD 701/1	CSIR-SAAO	
DATE 19-3-85 DRAWN D.C		E4-0273	

ozalid 84-326 02-84



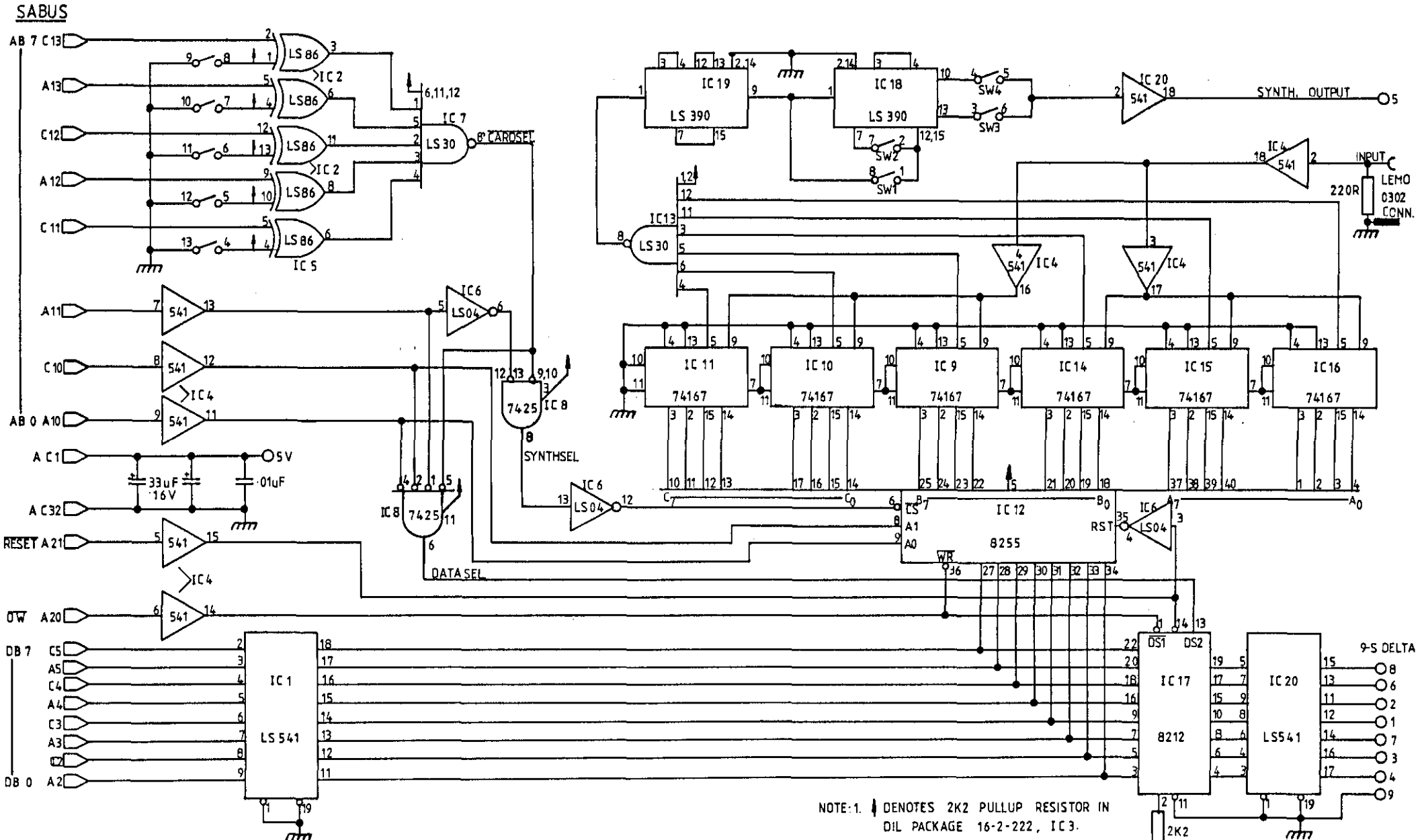
GENERAL PURPOSE I/O
WIRE WRAP CARD 701/1 Rev1

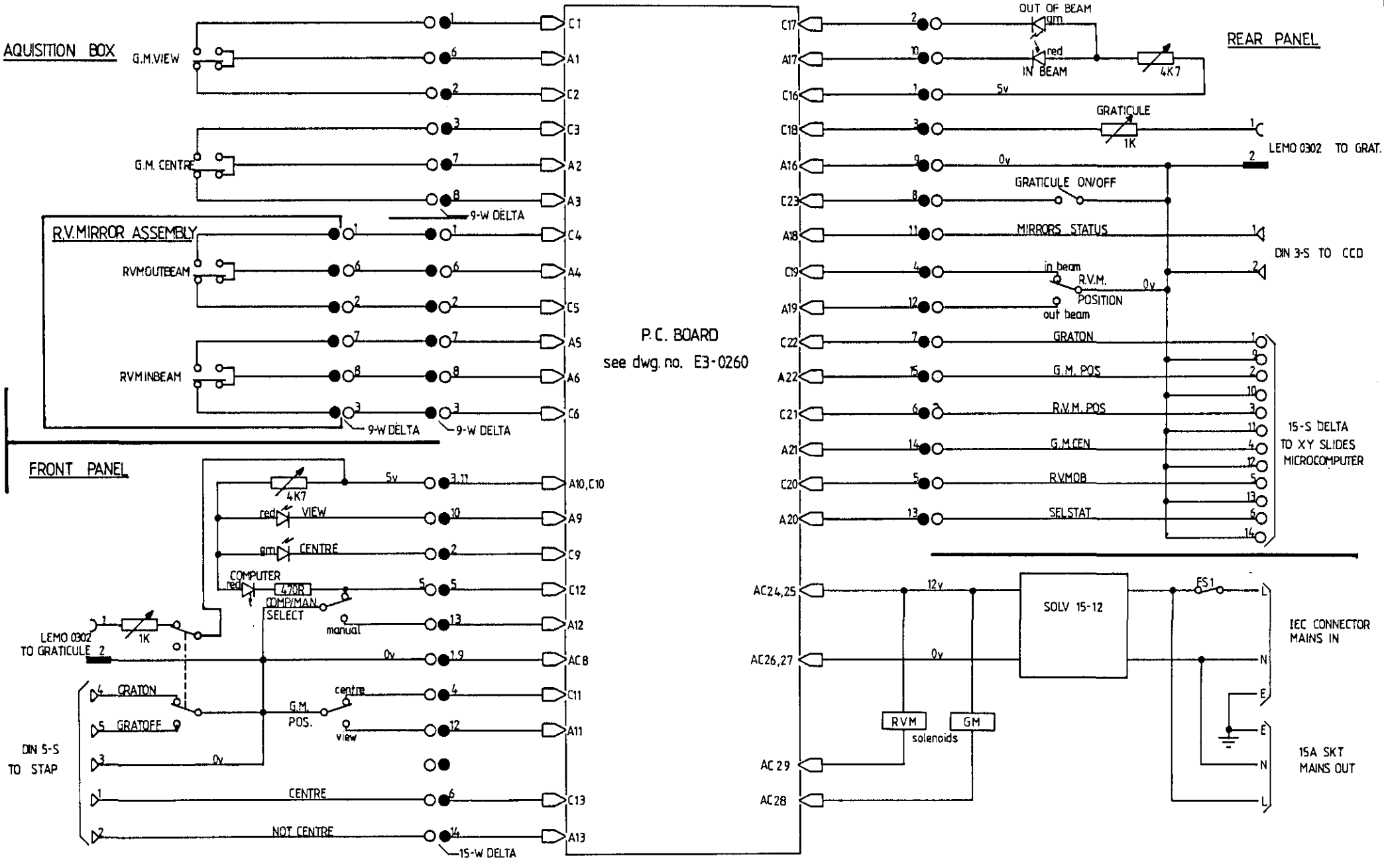
MAGGS INDUSTRIAL MICROS



NOTE: ⬇ DENOTES A PULLUP RESISTOR IN DIL PACKAGE 14-2-332, IC's 4 & 8
IC 9 - 74LS541
IC10 - 74LS10

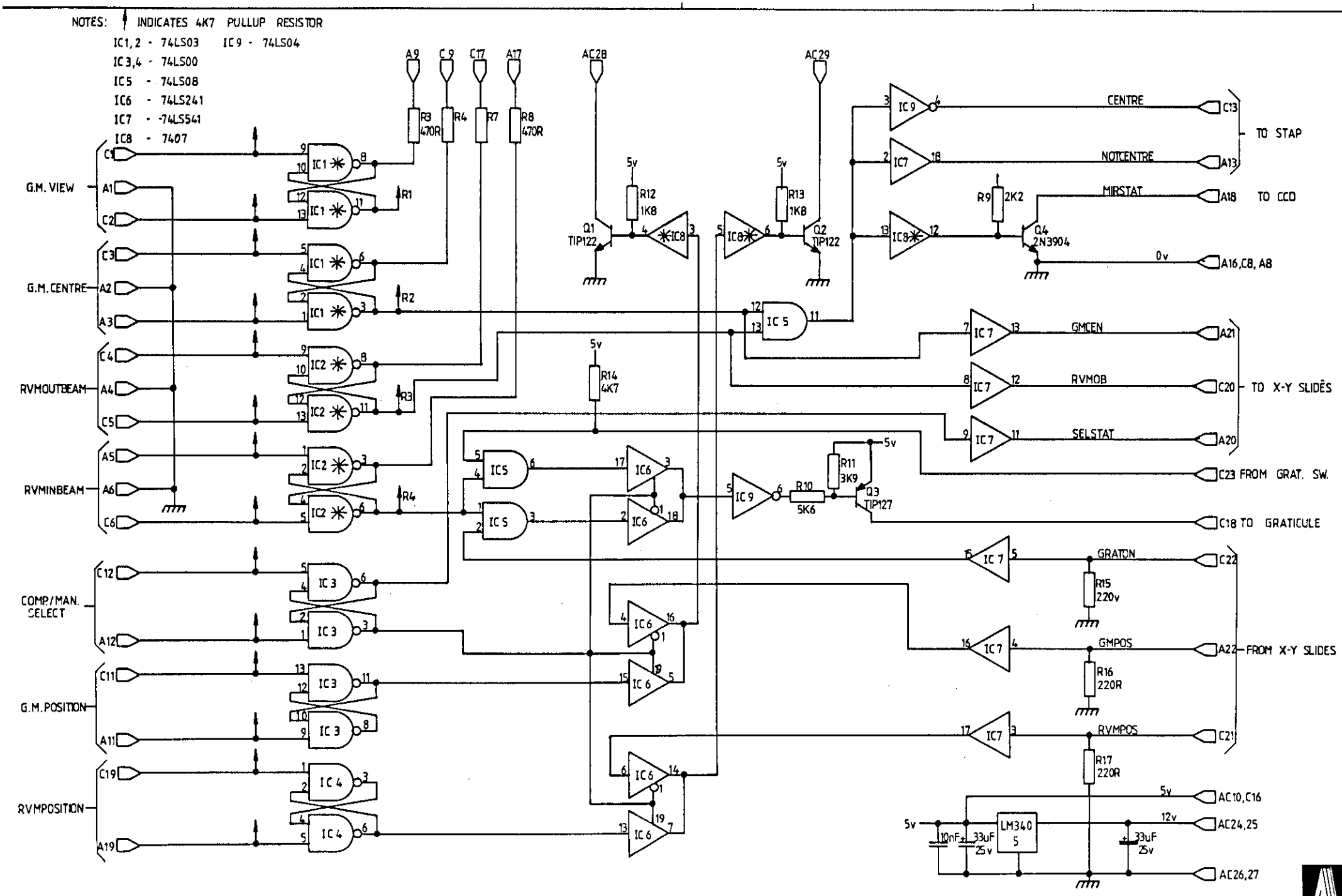
DATE 5-7-85	DRAWN D. C.	DESIGNED D. CARTER	TITLE SABUS OSCILLATOR MODULE WITH PRE-DIVIDE MOD. FOR XYSLIDES	CSIR-SAAO	E3-0125-01


SABUS
AB 7 C13
A13
C12
A12
C11
LS 86
IC 2
IC 5
6,11,12
IC 7
LS 30
8° CAROSET
IC 19
LS 390
IC 18
LS 390
SW4
SW3
SW2
SW1
IC 20
541
SYNTH. OUTPUT
5
IC 4
INPUT
220R
LEMO
0302
CONN.
IC13
LS 30
A11
C10
AB 0 A10
541
IC 6
LS04
IC 8
7425
SYNTHSEL
A C1
A C32
5V
33uF
16V
01uF
RESET A21
DATASEL
OW A20
IC 11
IC 10
IC 9
IC 14
IC 15
IC 16
74167
IC 12
8255
CS
A1
A0
RST
WR
DB 7
DB 0
C5
A5
C4
A4
C3
A3
IC 1
LS 541
IC 17
8212
DS1
DS2
9-S DELTA
2K2
NOTE: 1. DENOTES 2K2 PULLUP RESISTOR IN DIL PACKAGE 16-2-222, IC 3.
2. 4-WAY SW. TRUTH TABLE. X-NOT ALLOWED.
SW1 SW2 SW3 SW4
CL. X CL. X +200
CL. X X CL. +500
X CL. CL X +2000
X CL. X CL. +5000
DATE 21-9-82
DRAWN D.C.
TITLE SABUS FREQUENCY SYNTHESIZER
SAAO-CSIR
E3-0127



DATE 6-2-85	DRAWN D.C.	DESIGNED D. CARTER	TITLE 74" & 40" AQUISITION BOX WIRING DIAG.	CSIR-SAAO E3-0259

ozalid 84-325 02-84



DATE 7-2-85	DRAWN D.C.	DESIGNED D. CARTER	TITLE 74" & 40" AQUISITION BOX P.C.B CIRCUIT DIAGRAM	CSIR-SAAO E3-0260