UNIVERSITY OF SOUTHAMPTON
DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS

A HIGH PERFORMANCE PASSIVE TRACKING SYSTEM FOR WIND TUNNEL FREE-FLIGHT EXPERIMENTS

By

R.J.Halford

A Thesis

Submitted for the Degree of

Master of Philosophy

January 10, 1991

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND APPLIED SCIENCE
DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS
Master of Philosophy

A HIGH PERFORMANCE PASSIVE TRACKING SYSTEM
FOR WIND TUNNEL FREE-FLIGHT EXPERIMENTS
by
R.J.Halford

Free flight experimental techniques are used extensively in the determination of aerodynamic characteristics, particularly in studies of drag and stability. The acquisition of quantitative data on flight-history has been the subject of considerable effort and developments in technology adaptable to position sensing are continually tending toward higher performance. Experiments at high Mach numbers require resolution of small displacements while the susceptibility of the mathematical analysis to measurement noise necessitates high sampling rates.

This work investigates remote position sensing technology for an optimum solution to the problem of high speed tracking on a particular hypersonic gun tunnel with a flow Mach number of 8.4 for a 20 milli-second flight time. A practical system is demonstrated that exploits parallelism, high speed linear photo-sensor arrays and laser generated sheet beams in a unique geometry enabling economic flight-data acquisition, storage and processing on a microcomputer.

The author would like to thank the many people that helped to make the work reported here possible.
Sincere thanks to my academic supervisor Professor R. A. East, academic adviser Dr. M. J. Goodyer and to Dr. G. R. Hutt for his contribution to the original planning and supervision of the project. Special thanks to the technical services supervisor Mr. J. Taylor for his understanding and assistance during the various phases of this work.
Appreciation to Mr. C. W. Bielby and Mr. V. Judd of the mechanical workshops for their extraordinary efforts in understanding my constructional drawings.
Gratitude is felt to my friends and colleagues for their encouragement and helpful discussion, particularly Mr. D. S. Clayson.
Finally, I reserve my utmost gratitude for my wife, Maria, whose sustained support and tolerance has been essential over the last few years.
This work was funded by the Royal Aircraft Establishment, Procurement Executive, Ministry of Defence, Farnborough, Hants.

Contents

List of Figures

List of Symbols and Abbreviations

adc	analogue to digital converter
ccd	charge-coupled device
cid	charge-injection device
dac	digital to analogue converter
f_c	sensor pixel and acquisition sampling frequency
fov	field of view
fps	frames per second
k_1 to k_4	derived factors used in cone position recovery
led	light emitting diode
PAL	programmable array logic
pc	personal computer
pixel	picture element
psd	position sensitive detector
m	gradient of shadow edge
mos	metal-oxide semiconductor
mtf	modulation transfer function
TEM	transverse electric-magnetic mode
y_1 to y_4	shadow image/ sensor intercepts
y_{c1}, y_{c2}	shadow centroids
X,Y,Z	sensor axis
X_a, Y_a	cone apex co-ordinates
θ_c	cone half-angle
θ_p	cone axis pitch
θ_{pa}	cone apparent pitch
ψ_y	cone axis yaw

Chapter 1

Introduction

Wind tunnels have been used for studying aerodynamic performance since the late 19th century. Since that time, tunnels have been constructed in many shapes and sizes, using various flow media in an effort to simulate some of the conditions that exist in real flight. The demands on simulation and have increased dramatically as the need has grown for experimental work in support of hypersonic modelling. In sympathy with these developments has been the need for new instrumentation techniques to enable the collection of quantitative aerodynamic data. In this field, the convenience and economy of microcomputers, and related hardware (e.g. analogue interfaces) along with developments in semiconductor based sensing devices have had a considerable impact on the methodology of wind tunnel experimentation.

The traditional approach employed in wind tunnel testing to achieve both test model positioning and force measurements has been the use of mechanical supports in the form of either sting, struts or rods. The struts and rods connect the model directly to the tunnel wall with the advantage of allowing the use of large models, with an attendant improvement in simulation quality. However, the problems introduced by struts include flow interference and the influence of the wall boundary layer on the flow field which are both difficult to predict or control. Under these circumstances the sting may be preferred as aerodynamic interference can be minimised by positioning the sting in the model wake. The forces on the model are usually measured by strain gauges mounted on flexures within the sting or base mounting. The mechanical support approach has the advantage of being simple and reliable but for some applications the overwhelming disadvantage is the difficulty in prediction of support interference effects. Flow distortion, particularly for hypersonic flows can introduce significant variation in observations (ref [1]). In addition, the physical presence of a sting precludes studies of base shape effects. (A bibliography of papers on support interference may be found in ref [2]).

The problems of physical support interference can be avoided by the use of either suspension or free-flight techniques. Suspension is achieved using electro-magnetic solenoids

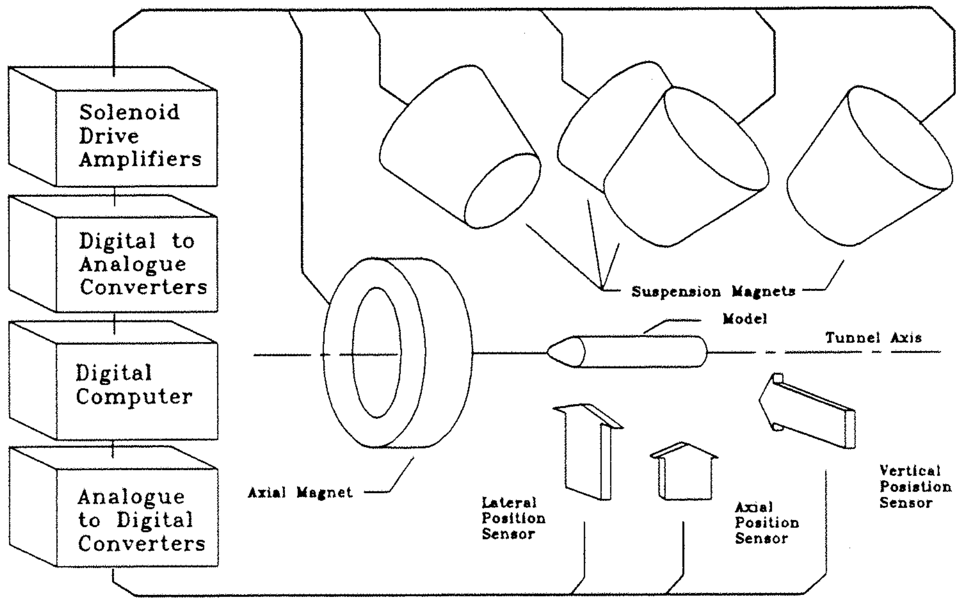

Solenoid Drive Amplifiers
Digital to Analogue Converters
Digital Computer
Analogue to Digital Converters
Suspension Magnets
Model
Tunnel Axis
Axial Magnet
Lateral Position Sensor
Axial Position Sensor
Vertical Posistion Sensor

Figure 1: An outline of a magnetic suspension system.

operating within a control loop and acting on magnets mounted in the test model[3]. There are several clear advantages to magnetic suspension, including total elimination of mechanical support interference, precise control of attitude and indirect sensing of aerodynamic forces, (via monitoring of station keeping drive to the electromagnets).

Despite the advantages of magnetic suspension, the expense of such an installation is generally less justified in application to an intermittent flow wind tunnel than for a continuous flow system. For the particular gun-tunnel providing the basis for this work the total running time is of the order of 20 milli-seconds and while precise model launching would be facilitated by suspension, free-flight tests provide a less complex solution and allow model motion to be determined purely by gravitational and aerodynamic forces.

With both suspension and free-flight techniques the model position detection system determines overall performance. Suspension requires real time absolute position response, usually at loop rates of a few hundred Hertz. Free-flight essentially requires a recording of the flight history, traditionally achieved by high speed cine-photography at rates of five thousand frames per second (fps) or more.

The techniques used for extracting aerodynamic data from the time history of the free-flight oscillatory motion of a model are well established[4]. The methods commonly used involve distance/ time differentiation and are, as a result, susceptible to measurement noise.

Typically film at 5000 fps would provide results with less than 140 data points per run with an accuracy on the film surface of about ± 0.15mm. The behaviour of vehicles at hypersonic speeds is of continuing relevance to military and scientific research. As performance expectations rise, there is demand for more precise behavioural modelling and higher quality of data. In the context of this report the implications are that improved spatial and temporal resolution must be achieved by development of a high resolution flight history recording or tracking system.

This work examines the requirements for improving data acquisition for a particular gun-tunnel/ model configuration and investigates sensor and microcomputer technology for an optimum methodology.

The techniques previously employed are discussed in this thesis and the logical progression from full, two dimensional image recording to high speed line scanning is examined. A practical imaging system is presented that demonstrates the use of vertical sheet beams of light and shadowgraphy to allow the deduction of flight history by simple geometric processing. Optical difficulties are addressed and the work progresses to the development of an operational system. While the objective of the practical work was to demonstrate a bench-top system that met the required specification, the design presented necessitates only minor mechanical modifications to allow integration with the hypersonic gun tunnel.

Chapter 2

Previous Work

One of the key advantages of film over other methods of recording flight history in free-flight experiments has been the abundance of information often contained in the two dimensional image. A great deal of useful work has been done with cine film on the study of flow and wake formation using schlieren lighting techniques to reveal variations in flow density gradients[4]. Recent experiments on the hypersonic gun tunnel

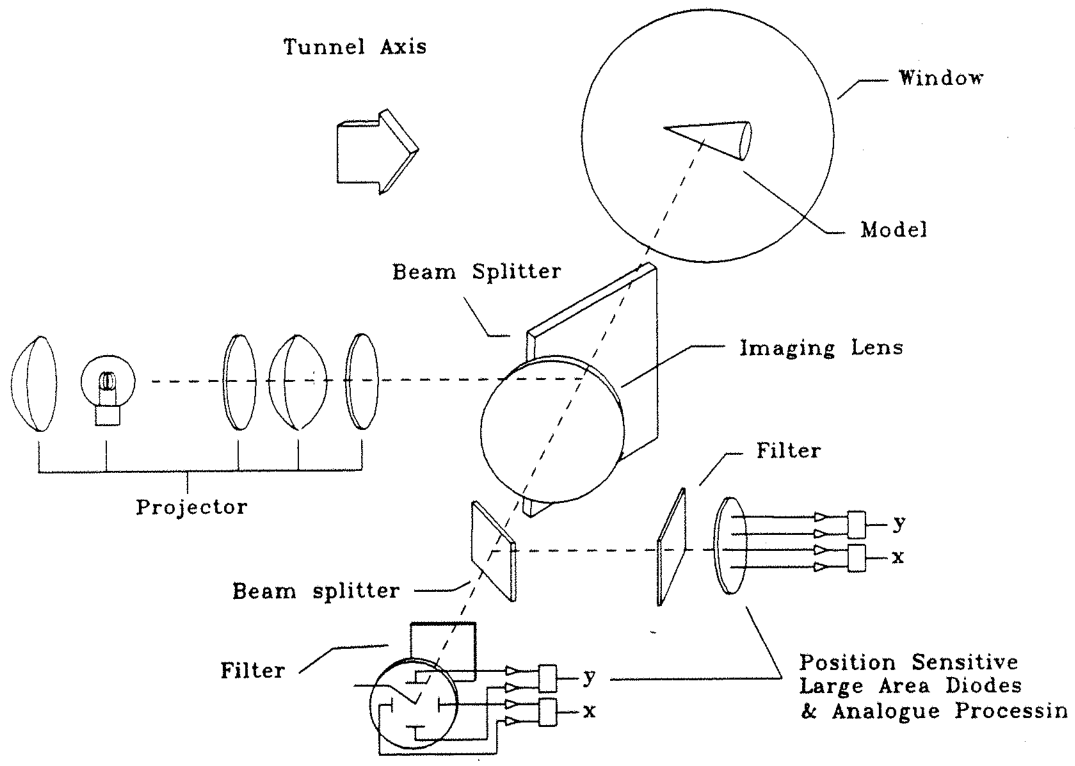


Figure 2: Tracking Experiment using position sensitive diodes.

at Southampton[5] were aimed at obtaining free-flight trajectory data over a wide field of view without the delays inherent in the use of film. An optical system using front illumination and large area position sensitive photo-diodes (psd's) was employed to produce an analogue output signal representing the position of the model in one plane. The psd is typically accurate to ±0.5 % over the field of view, at a resolution of 0.05 % and the experiments successfully demonstrated tracking for a single point on a conical model. The intention was to extend the principle to track multiple points using colour filtering. Difficulties experienced with this approach were related to uniformity and consistency of field illumination that resulted in problems with repeatability.

Chapter 3

Flight Tracking Requirements

Improvements on existing methods of determining test model flight history in terms of increased spatial and temporal resolution are necessary to enhance the quality of experimental results and accuracy of analysis. In addition, single shot wind tunnel firings can be fraught with timing problems. Instant computer access to acquired data would facilitate rapid validation of the run. The ideal system should achieve these ends economically.

Before proceeding to consideration of appropriate technology, the key characteristics of the gun tunnel will be outlined and a notional specification for flight tracking suggested. The tracking system is required to operate with free-flight experiments

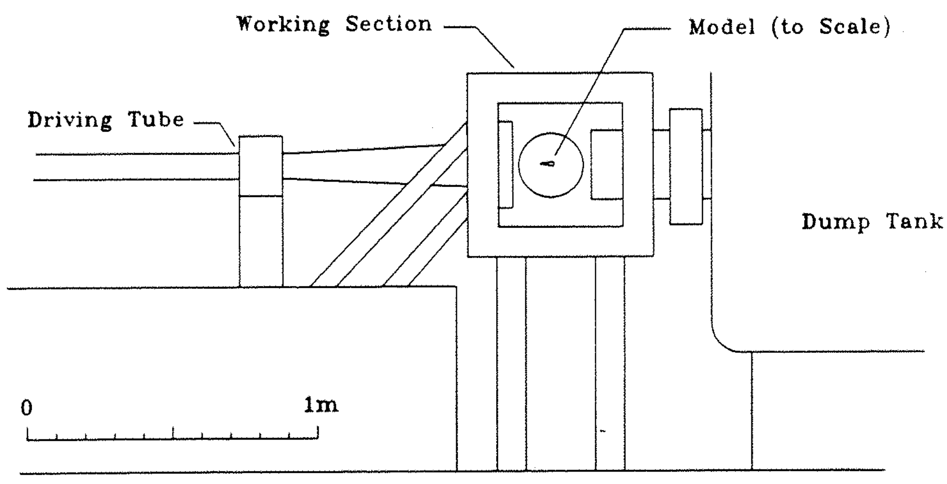


Figure 3: The working section of the Hypersonic Gun Tunnel - (Side Access Door Removed).

conducted in the open jet test section of a hypersonic gun tunnel with Mach 8.4 flow, a unit Reynolds number of 4.8 x $10^6 m^{-1}$ and 20 ms flow duration. The field of view is approximately 150mm diameter and accessible from both sides of the working section. A typical test model would be a 10 degree semi-angle cone with a 16mm diameter base. The test model would normally be held by electromagnet prior to the run and released in synchronism with flow commencement. Estimates taken from studies of film of typical model flights taken at 5000 fps indicate average horizontal speeds of approximately 2, 3.3, 4.6 and 5.6 m/s at downwind positions of 25%, 50%, 75% and 100% of the horizontal field of view.

The tracking system is required to follow planar motion vertically aligned with the wind tunnel. The principle of operation should be adaptable to tracking a model in the full volume of the test flow. The need for constant re-calibration should be avoided and data should be immediately available in digital form. Incidental objectives were to avoid use of esoteric or limited availability devices and emphasis was placed on minimising development time and costs.

A notional specification was proposed as follows:

- Field of view (fov): 100 by 100 mm nominal.
- Range: not critical.
- Target Dimensions: typically 10-50% fov.
- Resolution: 0.1mm.
- Number of samples: 1000.
- Flight Duration 10–20 ms.
- Maximum terminal velocity: 20 m s^{-1}.

A full image recording method meeting this specification would be required to record 1000 frames at 50,000 frames per second (fps).

Chapter 4

Full Visual Field Acquisition

The accumulation of working experience on the use of film in experimentation and the quantity of work that has been done on image analysis (see for example, target area tracking [6], manoeuvre estimation[7]) justify an investigation into the possibilities of achieving the required specification with a full image recording system. The investigation must determine the commercial availability of a suitable imaging system and the viability of the alternative, developing a custom design from available technology.

4.1 Cine Photography

One significant attraction of modern films is the high sensitivity to light, enabling exposure times down to nano-second durations to capture very high speed events on film. The techniques used in achieving high effective framing rates are not peculiar to hypersonic studies and have been the subject of intensive effort and much ingenuity[8].

Rotating prisms have been used in high speed cameras employing conventional film transport up to 40,000 fps. Above this speed the stress limits of film causes mechanical problems but by attaching the film to drums or discs, framing rates up to 100,000 fps are achieved. Turbine mounted mirrors and sophisticated lens geometry have taken the technique to over 8 million fps.

The photographic approach is currently widely used and provides high speed acquisition suited to many applications. The main limitations would appear to be the high initial equipment costs, running costs, the quantity of frames available, synchronisation to real-time events, film processing delays, geometric stability of the film, and the measuring precision on film. Timing precision can be improved by using tightly controlled timing of flash-lights. Digital image conversion of film images and subsequent image processing on a microcomputer work-station can improve film inspection throughput and facilitate a more quantitative analysis.

4.2 Electronic Imaging

One of the major disadvantages of film, the inherent delay from experiment to tangible results, can be avoided (with some limitations) by the use of video photography, which can be used to provide high resolution *or* moderate speed.

There are several options available for the recording medium but the most commonly used is multi-track magnetic tape. Helical scanning heads or rapid tape transport provide video bandwidth to cope with the high data rates. The performance of the video system is ultimately limited by the imaging device.

4.2.1 The Vidicon - High Resolution

The vidicon, a conventional electronic imaging device introduced by RCA in 1951, is capable of high resolution using a magnetically steered electron beam to read the charge pattern on a photo-conductive surface. The use of a Vidicon device with a resolution of

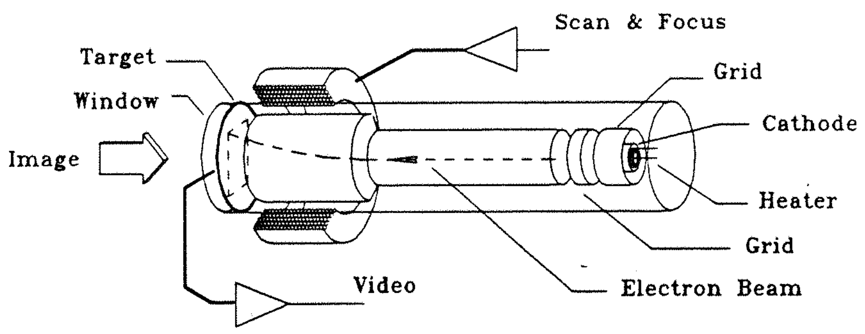


Figure 4: The Vidicon.

1600 by 3400 pixels was investigated in application to the Eglin Aeroballistics Range, USA, (refs [9, 10]). Image read-out time would seem to be excessive at 0.5 seconds and geometric distortion can be up to 3%. The cost of the high resolution vidicon is high and other attendant disadvantages, such as susceptibility to stray magnetic fields and mechanical fragility, have tended to suggest replacement with solid state devices.

4.2.2 The Solid State Camera - High Resolution

Semiconductor photo-sensor arrays using various readout strategies are now commercially available for use in video cameras with over 1000 by 1000 picture elements (pixels)[11]. Ford Aerospace, USA, will produce arrays of 4096 by 4096 pixels but at some considerable expense[12]. If pixels could be accessed at a clock rate of 10 MHz, the frame access time for this device would be 1.6 seconds. On the positive side, the

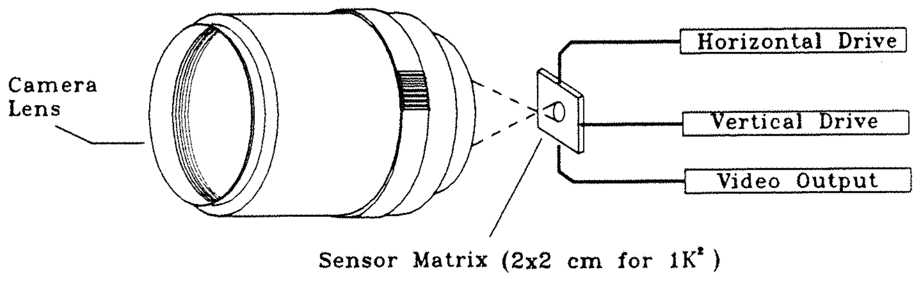
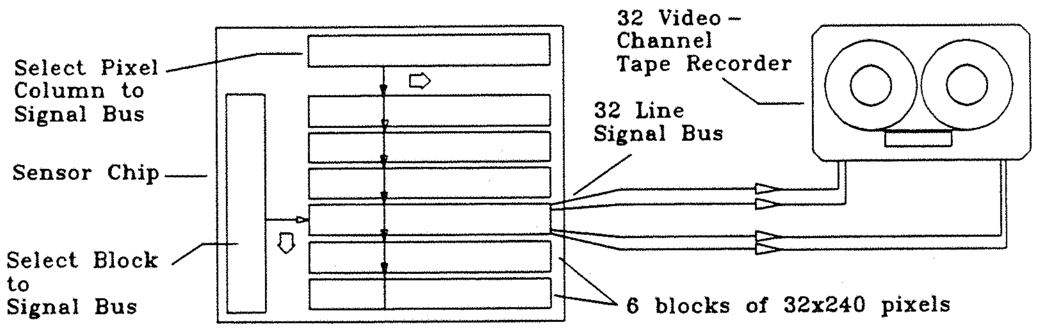


Figure 5: Basic elements of a CCD camera.

medium-to-high resolution devices are cheaper than conventional high resolution vidicon tubes, are more sensitive to light, geometrically precise and also stable against external fields. The negative aspect is that access is dominantly a serial process and the transport of 10^6 pixels from the 1000 by 1000 pixel array at a maximum clock frequency of 14 MHz produces framing rates inadequate for most free-flight recording applications.

4.2.3 The Solid State Camera - High Speed

Faster semiconductor cameras are available at the expense of sensor matrix density. A particular camera and magnetic tape system by Kodak[13, 14] uses a scheme of parallel access to achieve framing rates of 2000 fps with a sensor matrix of 192 by 240 pixels.



Select Pixel Column to Signal Bus
Sensor Chip
Select Block to Signal Bus
32 Video-Channel Tape Recorder
32 Line Signal Bus
6 blocks of 32x240 pixels

Figure 6: A high speed system using parallel access and storage.

In the Kodak sensor array, field effect semiconductor switches are used to first select one of six blocks of 32 lines in the array , and then sample each of the 240 pixels per line in turn onto a bus, each line being sampled in parallel onto 32 separate buses. The resulting 32 video signals are recorded onto a multi-track tape recorder using frequency modulation.

Permutations in pixel block access are possible and by reducing the matrix size to 32 by 240 pixels, the frame rate can be as high as 12000 fps. Spatial measurements within a frame can be made to an accuracy of about 0.5% of the frame width. Film run time is limited by the magnetic tape to about 45 seconds at 2000 fps.

Chapter 5

Two Dimensional Imaging Sensors

Electronic cameras can currently provide *either* high density imaging or high speed imaging. As electronic rather than film based high speed, general purpose recording systems do not seem to be available, the scope for developing a custom design matching the particular requirements for free flight studies with short flight durations needs to be examined. It is important to minimise technological risk and achieve useful results within a reasonable period of development.

The ideal acquisition medium would retain the information gathering potential of film and offer the geometric precision and immediate data access of two dimensional photo-sensor arrays. As fabrication of special sensor arrays was considered beyond the scope of this work in terms of cost, the development of a system with the required performance depended on the application of available device technology.

5.1 The Limitations

Interpretation of the specification suggested in chapter 3 as a full imaging system would involve a massive quantity of data. Over 1000 megabytes of data would need to be transferred at a rate of 50 GHz for a single run. Even if storing such massive amounts of data was practical, devices with the necessary serial pixel access rates are not available.

The most widely used semiconductor imaging device is based on charges produced by incident photons within a semiconductor. The mechanism for accessing these charges is a transport system that uses a pattern of electrodes on the semiconductor that set up electrostatic potentials that (with suitable phasing) cause the charges to move to to a collection region. The principle has many non-sensing applications in signal processing and devices using the transport mechanism are known generically as charge coupled devices, (ccd). Since the invention of the technique in 1969[15], much effort has gone into improving ccd performance, particularly in application to photo-sensors, where the emphasis has been on noise reduction, speed enhancement and increased matrix density. Work to increase ccd signal rates has concentrated on defining a region within the semiconductor that minimises transfer loss and electrode interaction. This approach has resulted in devices with a limiting transfer frequency of 180 MHz[16].

Design concepts for sensors using surface acoustic waves travelling in a piezo-electric substrate to achieve a charge transport rate of several GHz have been presented[17] but practical devices are not currently available.

Standard devices can be modified to optimise characteristics for a particular application. One approach to improving performance of a two dimensional array is described in ref [18]. Part of the sensor was masked off to provide a reduced matrix exposed to the image. Successive line images were exposed and transferred in turn to the masked portion of the device. This could be accomplished more rapidly than transferring the signals off-chip and short exposures were made possible. A temporal resolution of 6 μS was achieved using 511 rows of pixels transferred at a rate of 160 kHz.

Some manufacturers have addressed the access problem for high speed arrays by dividing an array into multiple segments that can be accessed in parallel. This approach as been applied to two and one dimensional arrays.

Even with this approach to fast access, the two available dimensional devices lack the required spatial resolution and effective framing rates to acquire whole field images with a single sensor. The prospect of using two dimensional sensors in this application depends on a strategy to reduce the quantity of pixels involved.

5.2 Image Dissection

One approach to reducing pixel throughput is to reduce the image detail loading on a sensor by dissecting the image into sub-areas, each scanned by a separate sensor.

Optic fibre assemblies have been used to achieve image dissection and also to map two-dimensional images into one dimension[19]. The advantage of this mapping is to allow the use of fast access linear arrays. Application of tapered bundles of fibre can also be used to provide an increased light flux at the sensor without using additional lenses.

5.3 Passive feature-tracking

A large reduction in the required pixel transfer rate can be obtained by restricting the number of pixels transferred to a scanning pattern surrounding features or outline of the model. Feasibility of this approach relies on the availability of sensors with random access or addressable pixels.

With regard to random access, two dimensional sensor arrays use one of three distinct access methods to transfer signals to the chip output[20] as shown in figure 7. Interline-transfer and frame transfer methods employ a scheme of parallel clocked analogue shift-registers for signal read-out and cannot be used for random access.

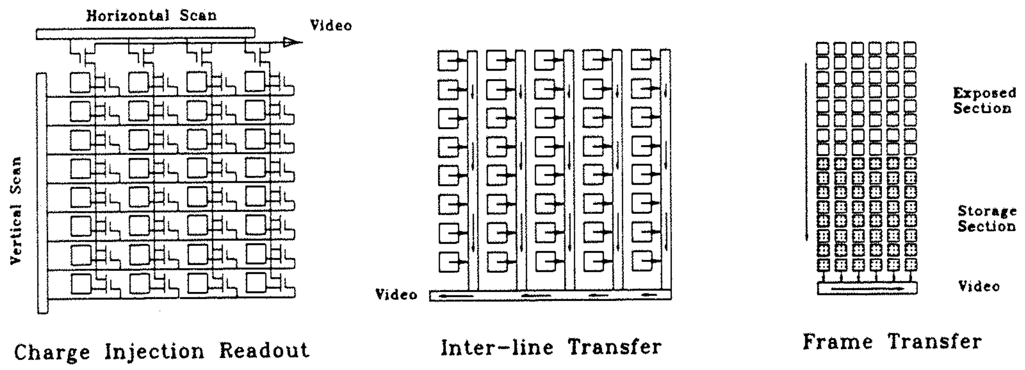


Figure 7: Accessing sensor arrays.

Rapid dumping of lines by synchronised increases in clock rates can be used to access a selected line within a frame but the only commercial photo-sensor array type with random access potential is based on metal-oxide semiconductor technology (mos), the MOS-XY or Charge Injection Device (CID). Unfortunately, CID's are not available in a suitable format. An example, produced by the General Electric Company, USA, could randomly access pixels in 2 μs, but was limited to a matrix of 128 by 128 pixels[21, 22]. CID arrays with a matrix of 512 by 512 pixels are available[23, 24] but random access would now seem to be inaccessible externally due to use of on-chip shift registers which ease connectivity from the manufacturers point of view. Schemes employing rapid clocking of access shift registers can be used to achieve pseudo-random access within the sensor area but the overall speed is poor.

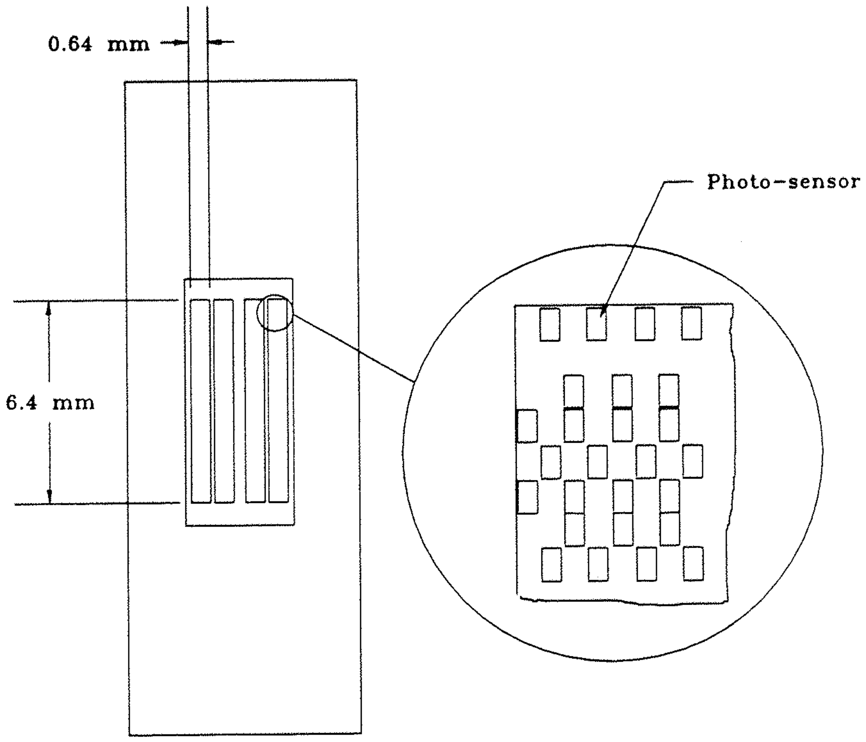


Figure 8: The optic ram device, (four 128 by 512 element arrays). Note the irregular photo-sensor distribution.

A device adapted to two dimensional image sensing and offering random access at very low cost is the optical ram (figure 8). This sensor merits consideration because of the low cost which would allow a scheme using large numbers of the devices to operate in parallel with the potential to achieve high speed and spatial resolution. The sensor consists of an adapted memory array fitted with a glass window to exploit the inherent light sensitivity of field effect devices used in dynamic memory arrays[25, 26, 27]. In one version of the device the array consists of a matrix of 512 by 512 pixels, with access time to any pixel of 200 ns.

Although the optic ram has a cost advantage over other types of sensor , several difficulties emerge from the fact that the device is adapted from a memory array and the circuit geometry is not optimised as an efficient optical sensor array:

- The layout provides a staggered but regular pixel arrangement of four rectangular groups, separated by non-sensitive strips. The aspect ratio of each group is 10:1 and an overall aspect ratio is approximately 2:1. The pixel placing is staggered so logical addressing would have to incorporate mapping to allow re-assembly of a representative image for subsequent processing.

- The polysilicon or aluminium connective circuitry reduces the photo-sensitive area to 50% of the total.

- The output state of each pixel is determined by the charge level remaining on the gate of a mos transistor, which is detected by an internal thresholding process. Operation is essentially binary, although grey scale in an image can be determined at low rates by making several interrogations of a pixel at varying time intervals[28].

- Intentional anti-blooming structures are not present in the fabrication and there could be problems with repeatability of the internal thresholding and linearity of response.

While these problems may be overcome with application of ingenuity, they do present technological risk and would probably require extensive development to achieve the aims of this work.

5.4 Active feature-tracking

Finally, image data could be reduced by steering a localised image of model features onto a fast sensor matrix of smaller dimensions than the overall image. Mirrors mounted on galvanometers or piezo-electric transducers could be used for image steering, with model position data being obtained from a combination of direct mirror attitude measurement and the image position on the sensor matrix. The mirror transducers could be controlled by a fast analogue servo loop, maintaining maximum signal output from a secondary photo-sensor. Tracking performance would be limited by the mirror steering elements (e.g. a typical galvanometer would have an unloaded settling time of 0.6 ms for a 20 degree step). Cascaded piezo-electric devices have a potentially faster response with good linearity (0.1 to 0.3%) but are limited to small motions (50 microns at 100 volts) and are subject to hysteresis and thermal effects.

5.5 Summary

The devices outlined in this chapter have in each case some associated disadvantage or difficulty that undermines application to the hypersonic tracking project. These factors include cost, performance and technological risk. The emphasis in this work was on minimising development time and avoidance of complicated calibration procedures. Attention was thus focused on position sensing using static sensors.

Chapter 6

Tracking Detail in the Visual Field

The conclusion drawn from consideration of available two dimensional imaging systems and sensor components was that a system producing a record of full or substantially full field images would not offer the performance required for this tracking application. The alternative to capturing the whole flight history as a series of image frames is to use a system that registers the position of some model image detail within the field of view. One way of achieving this aim is to filter or pre-process information from the full image and this can done either optically or electronically.

6.1 Use of Image Distortion

An approach that employs a form of optical pre-processing exploits an idea first described in a U.S. patent in 1979[29] in which a cylindrical lens is used to project line images from point sources of light attached to the test model onto a linear sensor array (figure 9). The sensor is insensitive to model motion lateral to the axis of the array and

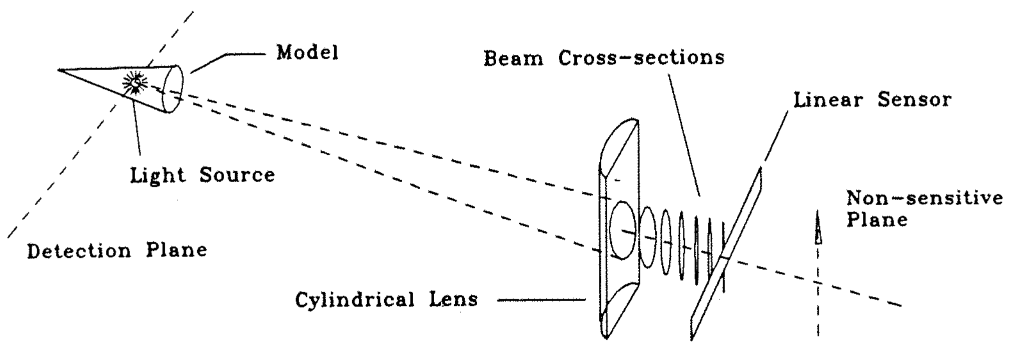


Figure 9: A cylindrical lens system.

detection of motion in several separate point sources is also possible with a single array.

The Eloptopus system[30] uses multiple sensors employing this idea to offer a field of view of 60 degrees horizontal and 12 degrees vertical with an angular resolution of 20 micro-radians, (0.04 mm at 2 m). The sampling frequency was restricted to 200 Hz, although the performance of currently available sensors should permit higher rates.

6.2 The Image Dissector Tube

A device that is capable of tracking image detail is the dissector tube [31,32]. This device is based on a photo-emissive surface combined with an electrostatic or electromagnetic field zone followed by an electron multiplier chain (figure 10). A focused image can

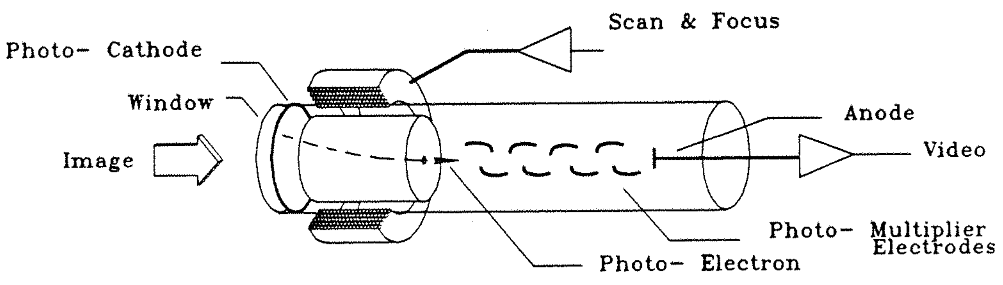


Figure 10: The image dissector tube.

be scanned selectively by steering photo-electrons from the required image region to the multiplier. The device is capable of high speed tracking, with effective pixel access to any location as fast as 15 μs. An ensemble of image dissectors has been used in triangulation sets for real time wind tunnel tracking over a volume of 0.8x 1.4x 0.5 metres. The resolution achieved was 0.02 mm at sampling rates of 1280 Hz[33].

6.3 Optical Computers

Optical computing methods offer extremely high equivalent image processing rates and are currently the subject of intense research[34]. High pass filtering of an image for edge detection, for example is easily accomplished optically but a higher degree of exploitation uses cross-correlation to detect target shift[35]. The target scene is illuminated with coherent light and presented at the focal plane of a convex lens, producing a pattern at the back focal plane representing the Fourier transform of the spatial frequency content of the image. A spatial filter matched to the object being tracked is placed at the

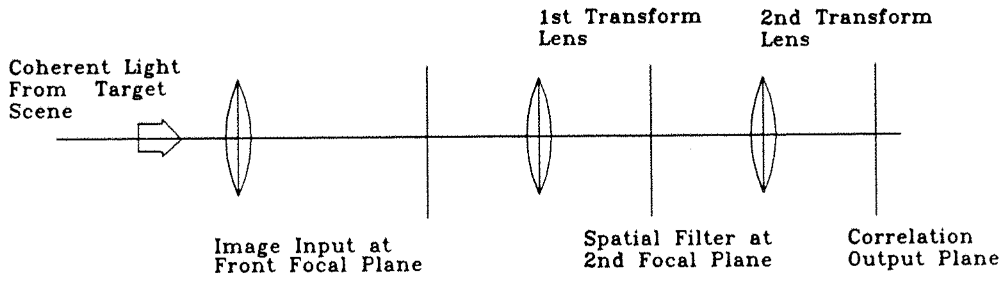

1st Transform Lens
2nd Transform Lens
Coherent Light From Target Scene
Image Input at Front Focal Plane
Spatial Filter at 2nd Focal Plane
Correlation Output Plane

Figure 11: Optical correlation.

back-focal plane and a final lens produces a correlation pattern peaking in intensity at a position corresponding to the location of the object. As the object moves the phase information changes but not the power spectrum, thus the correlation peak position tracks actual object position. A system demonstrating tracking of multiple objects has been developed at NASA's Jet Propulsion Laboratories, California[36]. The correlation provided by this technique is shift invariant but unfortunately sensitive to object rotation and scale.

In application to the free-flight tracking problem under discussion, model scale could be taken as constant, (on the assumption of negligible yaw motion). With regard to model rotation, various transducers have been developed for use as 'spatial light modulators' to enable the correlation filter to be dynamically varied. Addressable liquid crystal devices using electric fields to vary light transmission have proved an economic solution. A large area position sensitive diode could be used to provide rapid information on two-dimensional position and correlation peak amplitude. The correlation mask could then be rotated for maximum correlation to 'acquire' the target and tracking would be a matter of rotational dithering when the correlation peak drops below a threshold. Unfortunately the access times for the spatial light modulation device is restricted to normal television rates of a few tens of Hertz, (a device operating on the Faraday effect, a magneto-optic modulation of polarisation, claimed a 2 μs write time, but with a 70 μs erase time and a cost of \$23,000 for a 128 by 128 pixel array[37]).

The optical computer approach has great potential but the technical difficulties in meeting the required specification in terms of speed and economy would seem prohibitive at this stage of development.

6.4 Minimising Complexity and Data Redundancy

The techniques discussed in this chapter have similar problems to the devices outlined in the previous chapter in terms of high cost, development times and technological risk. Emerging from these considerations was the conclusion that tracking or acquisition of flight history must involve a reduction of imaging demands to achieve the required specification. The desirability of avoidance of constant re-calibration biased interest toward the use of fixed sensor arrays. The aim then becomes one of selecting a sensor geometry and distribution that minimises the pixel data acquired to a level consistent with flight history deduction.

Chapter 7

Tracking With Linear Arrays

The demands on a system for recording or tracking flight history can be reduced dramatically if the restricted range of model shapes used in hypersonic experiments is acknowledged. Advantage can be taken of characteristics of the particular environment, such as precise knowledge of model dimensions, access to the working section, the static field of view and the monotonicity of model flight. These factors allow the flight history of conic models to be deduced from a record of a few points on the edge of the model image or shadow.

7.1 The Basic System

A basic linear sensor system would consist of an ensemble of arrays that scan vertically with respect to the tunnel axis (figure12). Basic trigonometry is used on the intersection

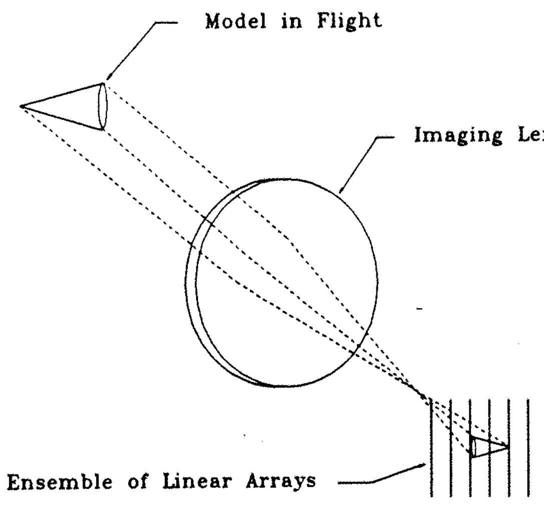


Figure 12: Position scanning using one-dimensional arrays.

position data in conjunction with known model dimensions to recover model position and attitude. The quantity and spacing of sensors along the tunnel axis are dictated by consideration of the model length and the need for a continuous record of model motion.

As the quantity of image data being collected has been minimised by the use of linear arrays, emphasis on the quality of the data increases. Less opportunity exists to employ statistical smoothing to improve signal to noise performance so this needs to be optimised by arranging suitable operating conditions. The short exposure interval implied by the high scanning rates required must be offset by high sensor sensitivity.

7.2 Sensor Response

The exposure interval required to provide 1000 positional samples during the model flight time of 20 ms imposes restrictions on the type of sensor device that can be used to provide a useful signal response.

The response speed in simple photo-diode structures is determined to some degree by the charge collection rate but more dominantly by the time constant formed by diode self-capacity and the value of the external current sense resistor. Small resistor values are required for rapid response to changes in light flux and consequently the output signal produced by realistic light levels is small.

In an effort to improve sensitivity, optical gain may be employed in which a multiplicative phenomenon is used to produce an intensified response to incidence of photons. Included in this category are avalanche diodes and photo-multiplier devices. The image intensifier (used in low light level applications) employs micro-channel plates to form an array of minute photo-multipliers with a phosphor screen as an intermediate output device. Phosphor glow persistence in these devices can be useful in extending response from brief events but generally the long decay of emissions prevents rapid repetition rates[38, 39]. Array devices using the optical gain of micro-channel plates with direct electron sensing to avoiding the rate problem associated with phosphors have been reported[40] but the current status is experimental.

Sensor sensitivity can be increased by integration of the photo-electric charge in a region physically close to the point of production and this approach is used to produce high signal to noise ratios in ccd photo-sensor arrays. In this technique, the charge carriers produced by incident photons are swept from the discrete photo-sensitive regions by an applied electrostatic field during the integration period to collect at adjacent potential wells produced by an electrode structure. The accumulation of charge at these sites is precisely proportional to exposure. Tests using pulse light emitting diodes (leds) have shown the time/ illumination level reciprocity relationship to be linear over a 10^6:1 range[41].

7.3 Sensor Access

Various schemes have been used to transport the charges collected at each site in the photo-sensor array to the device output. In one widely used technique, each photo-site is sampled in turn onto a bus structure, allowing remaining sites to continue integrating during site readout.

The impact of sampling noise can be reduced and a true image snapshot provided by replacing sequential sampling by a process that involves simultaneous transfer of the line of charges from a sensor array into an analogue shift register. In this approach the photo-sites start a fresh integration cycle while the analogue transport process is in progress. Signal to noise ratios of over 7500:1 are typical for single pixels, though pixel to pixel variations produce an overall ratio of typically 600:1. The energy requirements for saturated output depends on spectral content of the light, the bulk semiconductor used for sensor fabrication and the photo-site geometry.

A particular device well suited to this application is a Reticon linear array, type RL1288D[42] which consists of 1024 photo-sites with output taps at 128 pixel intervals (figure 13).

A feature that makes this device uniquely suitable for the high scan rates required is pixel access via 16 video outputs in parallel. At each 128 pixel interval there are two video outputs operating on alternate clock phases to provide 64 pixels each (figure 13). These output signals allow parallel signal access rates equivalent to an overall pixel sampling frequency of up to 240 MHz with a resultant line scan time of 4.2 μs.

7.4 Sensor Illumination

The RL1288D has a typical saturation energy requirement of 0.7 μJoules cm^{-2}, (2870 deg.K source with Fish-Shurman HA-11 1mm filter). For a sampling interval of 20 μs this would necessitate an irradience of approximately 31.3 mW cm^{-2} for array saturation. The decision to operate the sensors under saturation conditions is dependent on the characteristics of the illumination needed to ensure the required precision of measurement. Operation of the sensors in saturation was preferred because this offers a reduction in susceptibility to field intensity variations and reduces the impact of inherent non-uniformity in pixel response. Ideally, a high contrast image was required with rapid edge transitions which would simplify edge detection.

To assess the light levels that could be expected using reflected light, direct measurements were made on the gun tunnel using a 750 Watt quartz lamp mounted inside the working section. The lamp was mounted out of view of the sensor and away from the side window and directed towards the tunnel axis. The optics consisted of a spot frequency calibrated large area diode and a single lens, f-number 1.2, (focal length 12 cm), providing an image covering the sensor for an object distance of 155 cm. Various target coatings were tried and the irradience at the sensor surface was interpreted from the diode responsivity curve. Tests with fluorescent coating, matte white and no model present produced estimated irradiences of 51, 48 and 43 μWatt cm^{-2} respectively at the sensor. These approximations are nearly three orders of magnitude below the saturation requirements for the tapped array.

Higher intensity light sources could be applied, especially taking advantage of the short flight duration to exploit a well developed flash-tube technology but the high light levels needed for saturation are most effectively produced using a system of back-lighting. In this case, image surface details are of course lost but model edge detection can be reduced to a process of binary thresholding.

High intensity point sources have been used for back-lighting in tracking applications[9] but there are several distinct advantages to using lasers. The key benefits are high beam intensity and a predictable, stable beam geometry. The light beam can be expanded, collimated and shaped into multiple sheets of light that exactly match the dimensions of the linear sensor arrays. Visible light facilitates beam setup and a 16 mW Helium-Neon multi-mode gas laser was selected for the light source.

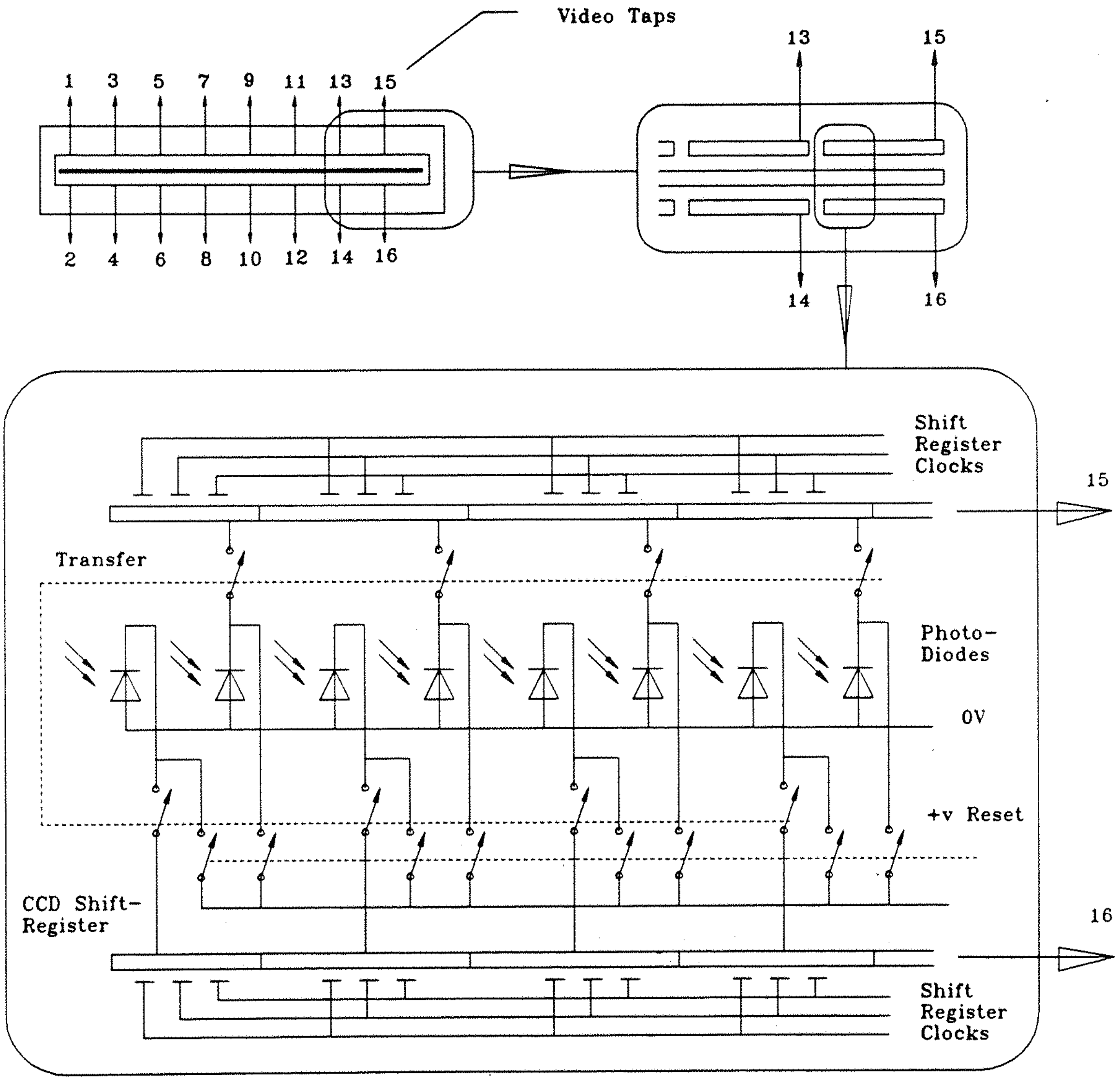


Figure 13: Parallel access with the tapped array, (representing the sensor mechanism).

Chapter 8

A Practical System

As the model in free-flight is constrained by the forces acting on it to vertical and horizontal, downwind motion, any particular vertical scan region (figure12) is redundant once the model has moved downwind. Advantage can be taken of this fact by arranging each sensor array to scan a two vertical zones separated by a distance greater than the model length (figure 14). While this scheme does introduce optical complexity, it

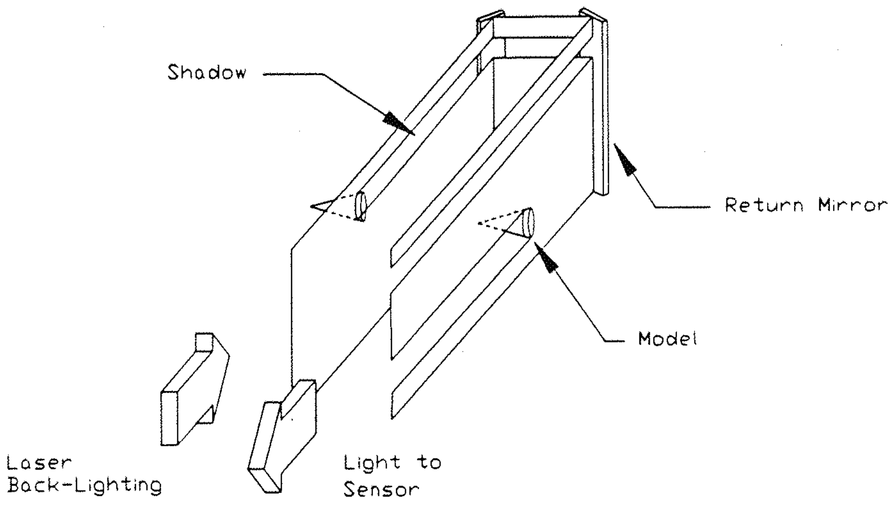


Figure 14: Laser beam folding to extend tracking.

also significantly reduces the expenditure on sensor arrays while maintaining a wide field of view, (at the time of this report the unit cost of array and circuitry was £2000). The modular optical and electronic design of the tracking system would allow additional sensors and support electronics to be added for a different performance/ budget emphasis.

8.1 The Impact of Diffraction

Layout and practical access requirements for the wind tunnel impose design conditions that result in significant optical path lengths. The obstruction of the beam by the model results in non-planar wave fronts caused by diffraction. The average intensity where these fronts meet at a distant surface are characterised in the near field as Fresnel diffraction and in the far field as Fraunhofer diffraction. Without additional lenses the shadows produced at the sensor plane for the system described so far approaches the Fraunhofer situation. This is more easily analysed as the phase components of the different rays arriving at any given point vary in a linear manner with lateral position. The resultant diffraction pattern has the form of the two dimensional Fourier transform of the complex amplitude distribution across the obstruction. For thin obstructions, such as a small conical model, the image produced is a symmetrical, low amplitude diffraction pattern from which the true edge positions cannot be recovered (see figure 15).

Fresnel diffraction occurs for receiving surface positions close to the obstruction and is comparatively complicated to analyse as the phase components no longer vary linearly with lateral displacement on the surface but must involve quadratic terms. For short distances from the obstruction, the diffraction effects at a single edge can be taken in isolation and a vector approach employing Fresnel's integrals can be used to find the resultant intensity with displacement. A plot of intensity with lateral displacement produces a curve that indicates some light diffracting into the shadow, and a damped ringing in intensity at the shadow edge, indicating interference of the unobstructed light with the diffracted light produced at the edge of the obstruction. A useful property of this curve is that the true edge of the geometric shadow corresponds to a point on the diffraction pattern where the intensity is 0.25 times the unobstructed beam intensity.

8.1.1 Shadow Imaging

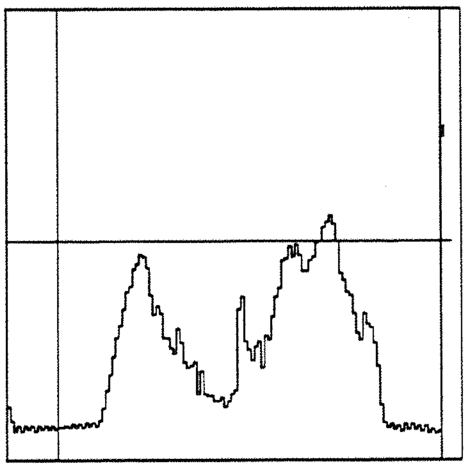
From consideration of diffraction effects it can be seen that some optical remedy is required if the model shadow edge positions are to be recovered. To overcome this problem an imaging lens is added to each of the sensors. These lenses are arranged to project a focused image onto the sensor when the test model is coincident with the plane in line with the tunnel axis, which is located at the principal object plane of the lenses as predicted by the Gaussian lens formula.

When the model deviates by a short distance from the object plane, the lens will produce a diffraction pattern of the Fresnel form, similar to that produced by direct observation of the shadow at a distance equivalent to the short distance separating the model from the object plane. It is possible to recover the geometric edge positions from the intensity profile produced.

The use of an imaging lens improves model edge detection but also introduces another problem related to the use of beam folding. A conventional lens will produce a single conjugate object and image. The use of folded beams as in figure 14 results in two separate regions where model intersection can occur for each beam. The image on a sensor will thus be either focused or unfocussed, dependent on the interception region.

To maintain a reliable track of model motion despite periodic loss of a focused image, the sheet beams have been interleaved as indicated in figure 16. Tracking with this modification is based on the principle that at least one array will provide a focused image, from which the precise position of the model shadow edge can be determined. Given this information, the centroid of the shadow obtained from the adjacent unfocussed image is sufficient to recover the model position and attitude. The operation of a sequence is indicated in figure 17 which represents four successive positions of a model in flight. The horizontal traces labelled 'a,b,c,d' indicate the output of the three sensors as the model progresses from positions 'a' to 'd'.

(a)



(b)

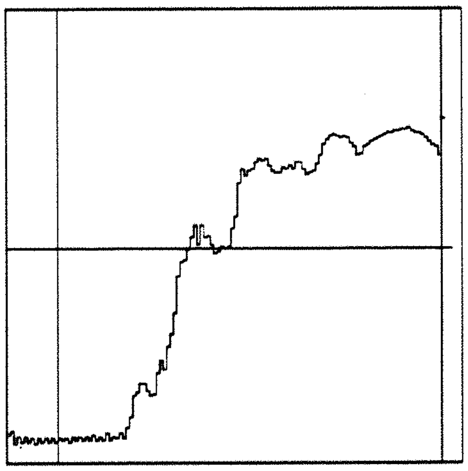


Figure 15: Distant model shadow image for (a) a thin wire (0.76 mm), (b) edge of a thick wire (5 mm). The vertical axis corresponds to sensor response (saturation near the base-line) and the horizontal axis corresponds to displacement along the sensor. The illustrations were captured using the system software in DEBUG display mode.

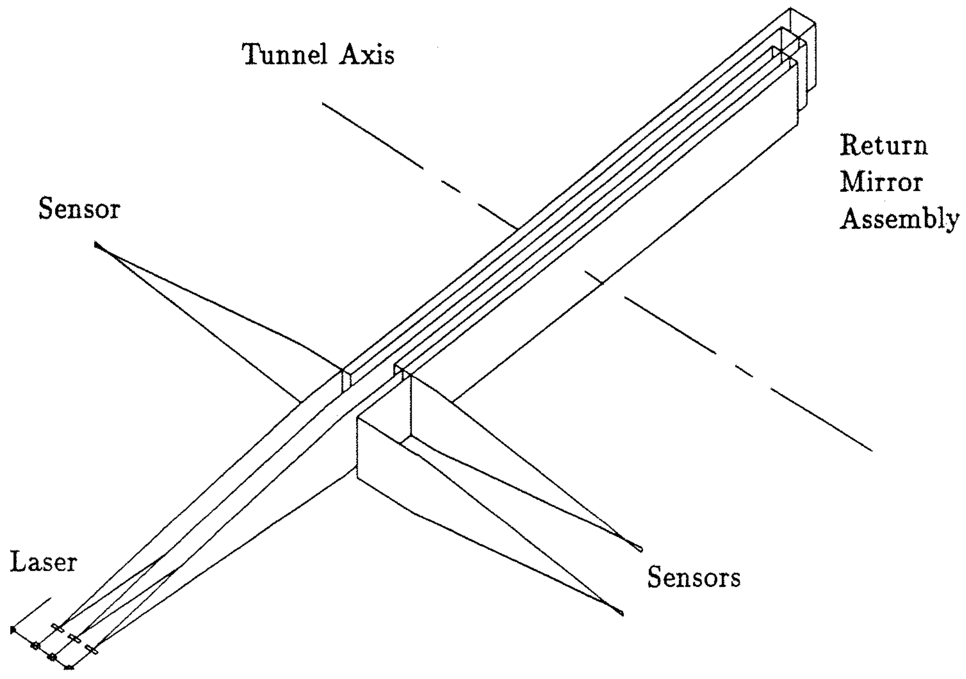


Figure 16: A simplified view of the ray path.

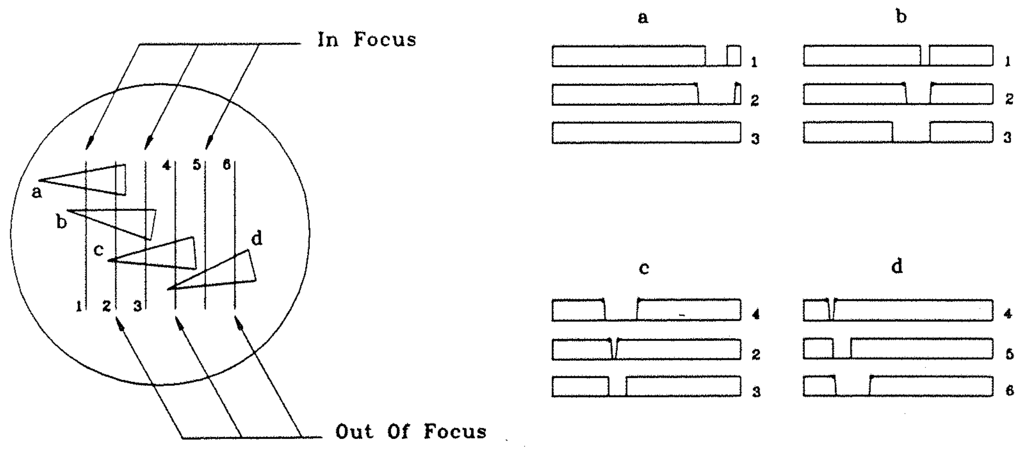


Figure 17: A representation of sensor views resulting from the use of interleaved beams.

8.2 Position Recovery

For the purpose of this discussion, the optical system axis is considered orthogonal to the tunnel axis, with the usual wind-tunnel axis convention replaced by the assignation of the y-axis for vertical displacement, and the x-axis for horizontal displacement confined to the tunnel axis. Positive x- axis displacement corresponds to downwind motion.

For periods during the model flight, the shadow will intersect more than two adjacent scan lines and signals will be available from all three sensors. The combined signals can be used to reduce the effects of noise and reduce the fluctuations in x-axis resolution. The minimum information available for a worst case of high pitch incidence consists of two precise intersection points on one sensor (focused image) and one value representing the centroid of the diffracted image (unfocussed image). The shadow transitions in the focused zone are abrupt and may be used with pre-processing to enhance signal to noise performance. Spatial and temporal filtering may be useful in smoothing the signal from the unfocussed image before the centroid is determined.

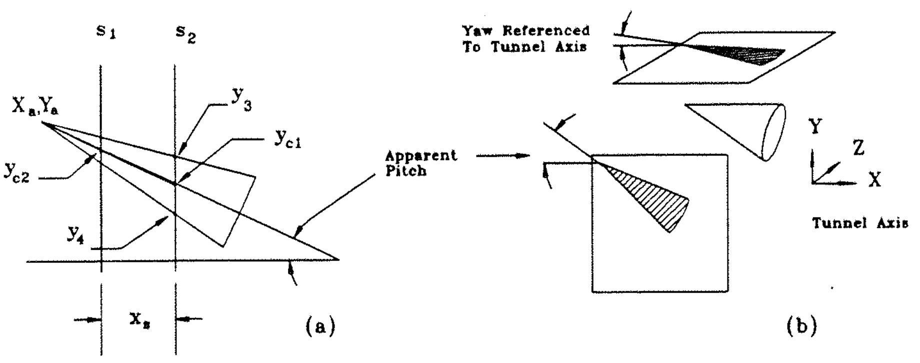


Figure 18: (a) The conic model/ sheet beam intersection, (b) Projection of the conic model shadow onto the sensing plane.

A typical conic model is represented in figure 18(a). S_1 and S_2 indicate the sensor scanning region, with S_1 currently providing an unfocussed image.

Given a precise cone geometry and the expected conditions of negligible yaw, it can be shown that the gradient of the shadow of one edge of the cone projected onto the sense plane can be given by :

$$m_1 = \frac{-k_3 + (k_3{}^2 - 4 \times k_2 \times k_4)^{\frac{1}{2}}}{2 \times k_2} \tag{1}$$

where

$$k_1 = \frac{(y_{c2} - y_{c1})}{x_s}$$

$$k_2 = tan(2 \times \theta_c)$$

$$k_3 = 2 \times (1 + k_1 \times k_2)$$

$$k_4 = k_2 - 2 \times k_1$$

y_{c1} = centroid of diffracted image.

y_{c2} = centroid calculated from points given for intersection of focused image.

θ_c = cone half-angle.

x_s = sensor x-axis separation.

from this gradient, the axis pitch can be calculated from :

$$\theta_p = arctan(m) + \theta_c \tag{2}$$

and the position of the apex from :

$$X_a = \frac{y_{c2} - y_4}{m - k_1} \tag{3}$$

$$Y_a = k_1 \times X_a + y_{c2} \tag{4}$$

The following programme extract shows how this is executed using Pascal (shadow intersections y_3 & y_4 and centroid y_{c1} are known):

```
procedure get_apex(yc1,y4,y3);

begin
  yc2:=y4+(y3-y4)/2;       {get other centroid}
  k1:=-(yc1-yc2)/x_sep;    {k2 is model constant}
  k3:=2*(1+k1*k2);
  k4:=k2-2*k1;
  m:=(-k3+sqrt(sqr(k3)+4*k2*k4))/(-2*k2); {find valid root}
  est_pitch:=arctan(m2)+cone_half_angle;
  xa:=(yc1-y2)/(m2-k1);
  ya:=m*xa+y2
end;
```

The subroutine executes in approximately 2 ms with an 8MHz clock using real-type numbers without a maths co-processor.

The relationship between yaw (relative to the wind tunnel axis) and the pitch angle of the cone axis is dependent on pitch (figure 18) and is given by:

$$\theta_p = \arctan(\tan\theta_{pa} \times \cos\psi_y)$$

where

θ_{pa} = the apparent pitch of the model shadow projected onto the sensor plane,

ψ_y = the yaw angle formed by the cone and the tunnel axis.

The motion analysis needed to give the aerodynamic coefficients of the test model requires a history of model flight in terms of cone axis and co-ordinates of the centre of gravity of the model. This data can be derived by use of the equations 2 to 4.

Chapter 9

Optical Considerations

The practical objective of this work was to demonstrate the operational principle of the major tracking system components. Modifications from bench top rig to the tunnel application were to be restricted to minor changes. The disposition of optical components was consequently influenced by tunnel dimensions and access conditions. To avoid conflict with positioning of underground ducting and access covers fitted close to the tunnel working section, a three point optical bench support was necessary. A cruciform bench shape was chosen to satisfy these conditions and allow close proximity mounting of the imaging lens/sensor assemblies (figure 19).

The single laser beam is split into three beams, which are spread and collimated in the vertical plane to form three parallel sided sheets of light (figure 16). The three beams cross the working section where they are displaced and reflected by a mirror assembly so as to return across the working section. Three imaging lenses then intercept the beams and project images of the model shadow onto the three sensors. For a given beam, the imaging lens produces a focused image for model positions in the plane of the tunnel axis, where the beam crosses the tunnel for the second time.

The propagation properties of the laser beam and the diffraction effects on edge recovery must be considered in a refinement of the practical design.

9.1 Laser Beam Propagation

The laser beam passes through beam-splitters to produce three beams which are each expanded by a separate cylindrical plano-convex lens and then collimated by a second lens. A spherical plano-convex lens is used for the collimation lens to avoid the substantial cost of fabrication otherwise required for a cylindrical lens of this size. There

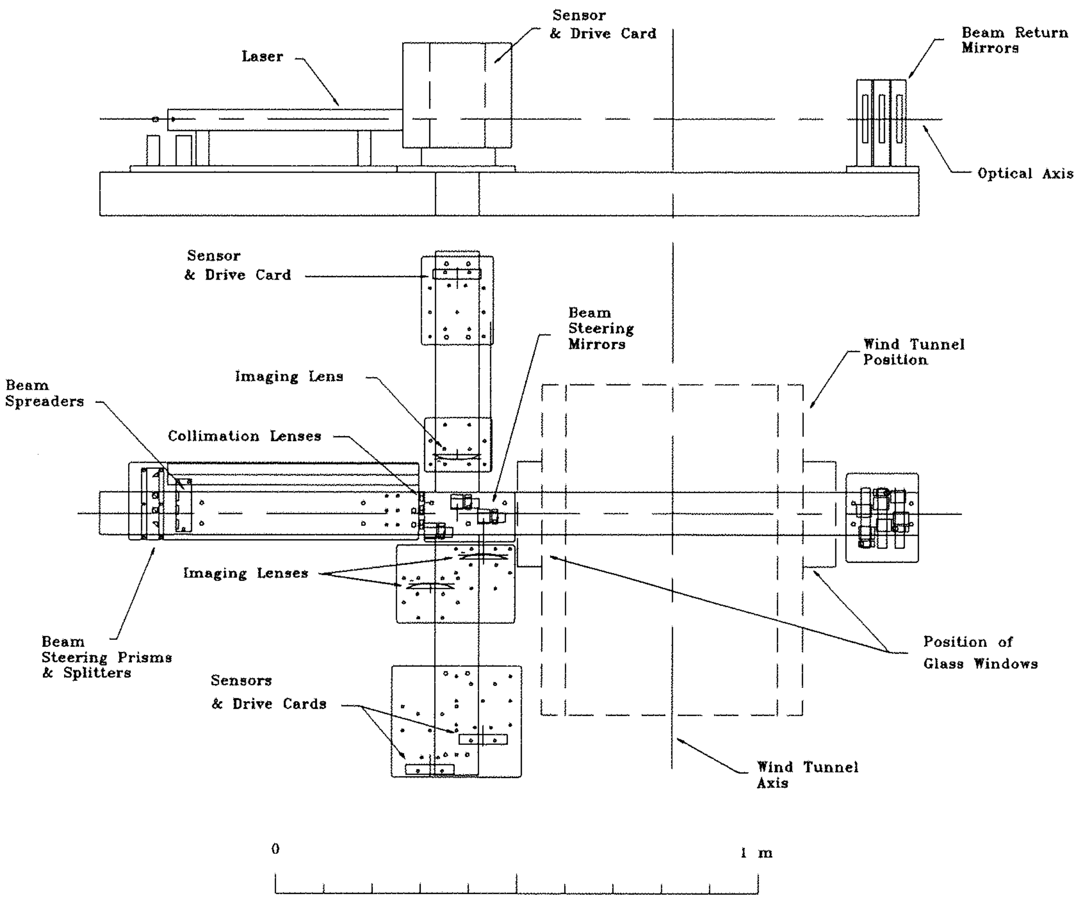


Figure 19: The optical bench assembly.

is also a beneficial side-effect to using a spherical lens.

Laser sources are distinguished not only by the coherence achievable in light output but (more significantly in this application) by the low divergence possible. Laser sources can operate in several transverse modes that produce distinct amplitude and phase profiles across the beam. The lowest order transverse mode (TEM 00) provides the minimum divergence, typically 1.2 milli-radians or approximately 1.2 mm increase in width per metre beam length. Unfortunately this mode produces a Gaussian intensity profile across the beam which is not the optimum form for the generation of sheet beams.

Various methods have been suggested to re-shape the intensity profile[43] but attenuation and interference effects introduce complications.

A laser operating in higher order transverse modes produces a more complex intensity and phase profile that tends to broaden the useful beam width and for a given laser size extract more energy from the laser cavity. The beam intensity profile is closer to the flat topped form required for back-lighting the model. On the negative side, the beam is wider and divergence greater than the equivalent low order device. For comparison to single-mode lasers, the divergence for a particular multi-mode laser of beam diameter of 1.32 mm was about 2.5 milli-radians.

An additional problem with multi-mode lasers is that diffraction effects are more difficult to predict because of the complex phase profile and an empirical approach is needed in optimisation for most applications.

The impact of vertical divergence on the sheet beam (increasing beam height) is reduced in the process of expansion and collimation by a factor equal to the beam expansion ratio. The lenses used in this design produce an expansion ratio of 50:1, so for a beam path length 2 metres and a laser source divergence of 2.5 milli-radians, given optimum collimation the final divergence will be of the order of 0.1 mm. This divergence represents one component of the overall static geometric distortion of the system for which it should be possible to apply compensation.

The effect of divergence in the horizontal plane is to broaden the sheet width with the possible effect of reducing the beam intensity at the sensor. This broadening is partially offset by the beneficial side effect of using a spherical collimation lens discussed at the beginning of this section. The cylindrical spreading lens has little effect on the ray geometry in a beam-width sense but the collimation lens will focus the beam in this plane to a minimum width at the lens front focal plane. The focal length of this lens is chosen to match the distance of the lens from the wind tunnel axis, so that the minimum beam width, and thus the maximum intensity occurs in a plane close to the tunnel axis. This results in a maximum beam intensity at the prime model position in the tunnel.

9.2 Depth of Field

The introduction of the imaging lens into the optical system necessitates consideration of the depth of field available. The use of folded beams results in two possible regions of beam contact for the model separated by approximately 1 metre. The imaging lens produces a focused image from an obstruction in only one of these regions. When the model is obstructing the beam in the other region, the shadow image produced is an interference/ diffraction pattern. The true model shadow edge positions cannot be recovered from this complex image but the pattern is symmetrical, (neglecting a small deviation resulting from asymmetry of model/ beam contact for the model with pitch incidence), and the sensor response is sufficient for recovery of the position of the image centroid.

The concept of depth of field is not so relevant for the unfocussed image: the use of collimated back-light ensures that the centroid of the diffraction pattern will track (within paraxial limitations) the model position independently of model position along the optical axis.

The recovery of model position (section 8.2) requires location of the centroid of the shadow of one model/ beam intersection and exact location of the shadow edges for a second intersection. The shadow edge positions are determined from the focused shadow image and depth of field is in this case important. Depth of field in imaging applications is usually concerned with the acceptable resolution of spatial detail. In this application it is resolution of edge position changes that is of interest and the apparent change in width of the shadow produced by variations in model distance from the imaging lens.

The image produced by the model when situated on the object plane conjugate with the image plane replicates the intensity profile at the point of beam obstruction. The sharpness of the shadow transition is limited by the quality or modulation transfer function (mtf) of the imaging lens. When the model is displaced from the conjugate object plane, the effect is equivalent to sampling the image of the scene at some distance away from the beam obstruction caused by the model.

The intensity profile of the shadow produced by the model is then of the Fresnel form as discussed in section 8.1. The true position of the edge of the shadow could be evaluated by extrapolation of the curve form to allow estimation of the 25% intensity point. The possibility of a more precise method is suggested by Seitz[44] for edge position extraction to sub-pixel resolution with independence from mtf. In this method a continuous function is reconstructed from the sampled data using a Gaussian reconstruction filter and the geometric edge position is determined as corresponding to the position of the first maximum in the first derivative.

The applicability of the sub-pixelation approach to the shadow imaging system in this project is not clear. The existence of intensity variations across the beam and unpredictable interference patterns from the source become more significant as the importance of the exact pixel value increases. Increased demands are placed on the performance of the optical system in terms of lens defects and alignment precision.

Basic video level thresholding has the advantage of simplicity and can provide the exact model position when the image shadow transitions occur in less than one pixel spacing. Under these conditions sub-pixel resolution is not possible and precision is limited by the pixel spacing. If the intensity of the back-lighting is assumed uniform across the field and the shadow transition is spread over several pixels, a fixed level can be chosen for the video threshold that approximates the 25% intensity level which, in the Fresnel diffraction pattern, corresponds to the shadow edge. However, the positional accuracy available does depend on the uniformity of illumination. The laser beam intensity profile is not by any means flat and the edge position recovered by fixed level thresholding must be subject to modulation by the transverse characteristics of the beam. The solution in this case is to use individual threshold levels that are preset according to the unobscured illumination levels for each pixel in the array.

9.3 Resolution of the Lens and Sensor

Beam profile, beam geometry, lens design and image blur due to model motion and vibration all have the effect of limiting the image resolution. Resolution of spatial detail (the visible separation of two points) is ultimately limited in a perfect lens by the effects of diffraction. This produces an intensity profile for each object point (called the point-spread function) similar to a $\frac{sin x}{x}$ function.

In any high resolution application account must be taken of the spatial frequency response of the lens indicated by the modulation transfer function. With a lens/ sensor combination, use of a good quality lenses will produce a spatial resolution limited by the discrete sampling matrix of the sensor. In this case the Nyquist criteria must be satisfied (from sampling theory, this criteria requires that the maximum spatial frequency detectable without aliasing is equal to half the pixel spacing).

A good quality lens with the image and object at conjugate positions can produce a near binary image. In this case the image of the edge will project onto some part of a single pixel element and the sensor output will depend on the percentage of the pixel illuminated. Elements either side of this particular element will be either in the light or in the shadow. The resolution of the lens-sensor system used for edge position detection in this work is primarily dependent on sensor element spacing, sensor separation, model profile, performance of the acquisition system and the form of image processing used. For a basic system using fixed video signal thresholding and image magnification of 4, the limiting sensor resolution is determined by pixel spacing to be 0.07 mm.

Chapter 10

The Design Outline

10.1 The Mechanics

The working height provided by the standard component fixings used with conventional triangular optic rail was not considered ideal for the basis of this work and the requirement for small separation on multiple sheet beams (15.3 mm) led to the decision to use a square section, extruded aluminium bench system with custom-made optical mounts. The various beam steering devices required sufficient adjustment resolution and enough degrees of freedom to allow beam alignment. Part of the design philosophy was to minimise costs and as repetitive alignment was not expected, screw adjusters were used extensively with simple slide-stages and alignment jigs. The optical sub- assemblies are mounted on base plates which can be relocated along the support bench to facilitate adjustments in design, (see figures 20 & 21).

10.2 The Optics

The useful beam width available at laser aperture must be estimated before determining the required beam expansion ratio. The useful beam width depends on the laser intensity profile. For a single mode laser (TEM 00) an estimate can be made as the profile is known to be Gaussian but the multi-mode device is more complicated as the exact transverse mode is often not specified by the supplier. The most effective approach to establishing the beam expansion ratio required is to measure the laser beam width directly, using neutral density filters and one of the linear sensor arrays. Once the expansion ratio has been estimated, the collimation lens can be chosen on the basis of available standard lenses and lens characteristics. A long focal length, spherical plano-convex lens was chosen for collimation for several reasons: cylindrical lenses of the required height are not generally stock items, requiring expensive fabrication; the long focal length lens has less surface curvature and is less susceptible to aberrations; by focusing the beam in the plane perpendicular to the collimation plane the impact of diffractive beam divergence over the long path lengths is substantially reduced. The focal length of collimation lens is chosen to give a waist in beam width coincident with the tunnel axis, providing optimum intensity in the imaging lens object plane. The focal length of the beam spreading lens is then set by the required expansion ratio, which corresponds to the ratio of the focal lengths of the two lenses.

For the imaging part of the design, the required magnification is set by the ratio of required scan beam height to sensor height. Image and object distances are restricted by practicalities of tunnel access, availability of lenses, the need to maximise depth of field and minimise geometric distortion in the image. These restrictions are met by using a long focal length lens, producing a focused image from an object intersecting the beam on its most distant pass through the working section.

Cube beam splitters and prisms are used to generate the three separate beams. Plane mirrors are used to return the beams across the working section and steer the beams into the imaging lenses. (See Appendix B on surface losses).

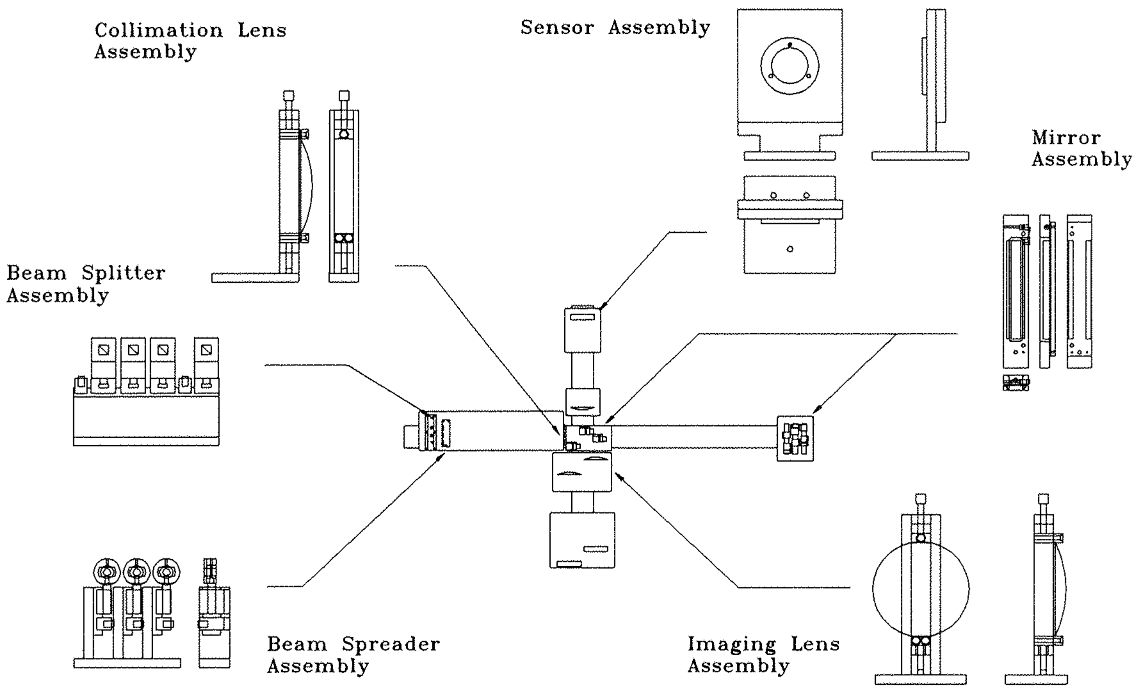


Figure 20: The major opto-mechanical components.

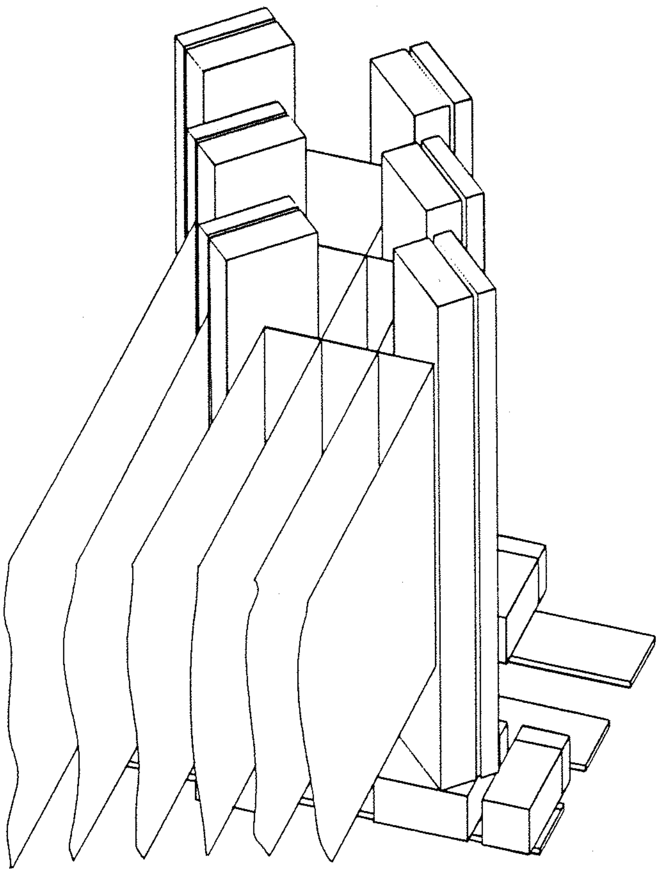


Figure 21: The beam return mirror assembly.

10.3 The System Electronics

The electronic components of the tracking system consist of a computer, (with 8086 microprocessor), the interface & control card and three rack mounted sub-assemblies (figure 22).

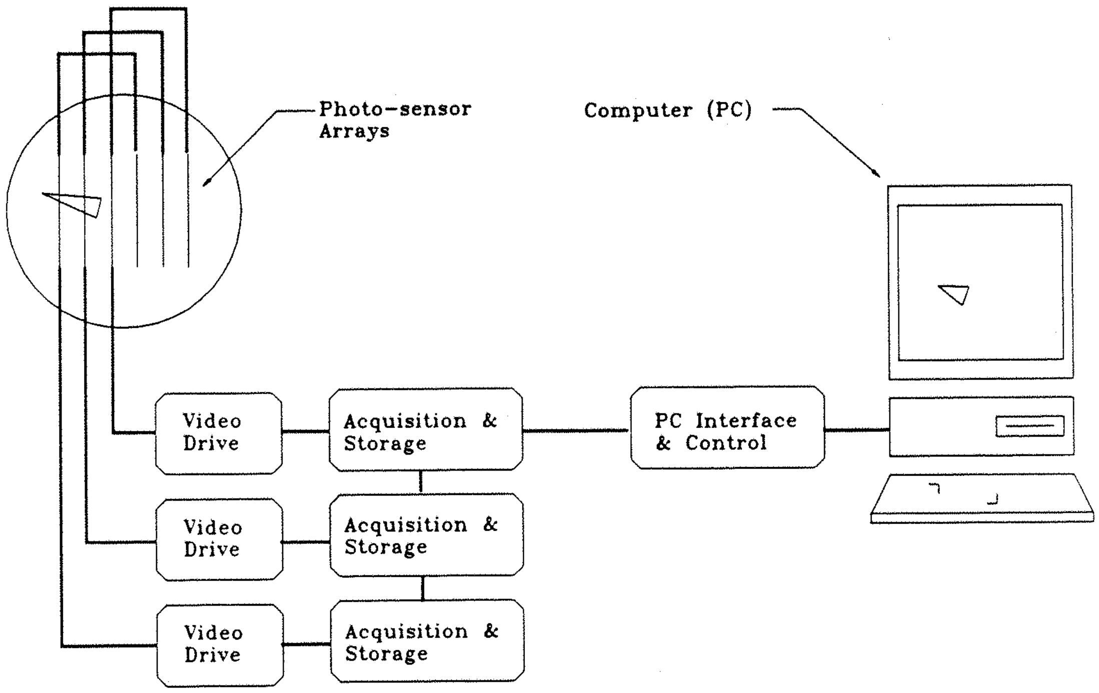


Figure 22: The principal electronic components.

The rack addressing method and modular design allows for up to 15 sets of racks and sensors (with appropriate upgrading of power supplies). Balanced line driver logic could be added to the logic lines communicating with the interface card to allow remote positioning of the computer. The physical layout of the three rack system is represented in figure 23.

10.3.1 Sensor Drive

The sensor arrays operate with photo-induced charges of a few pico-Coulombs and must be mounted as close to the drive circuitry as possible. The sensor drive cards have

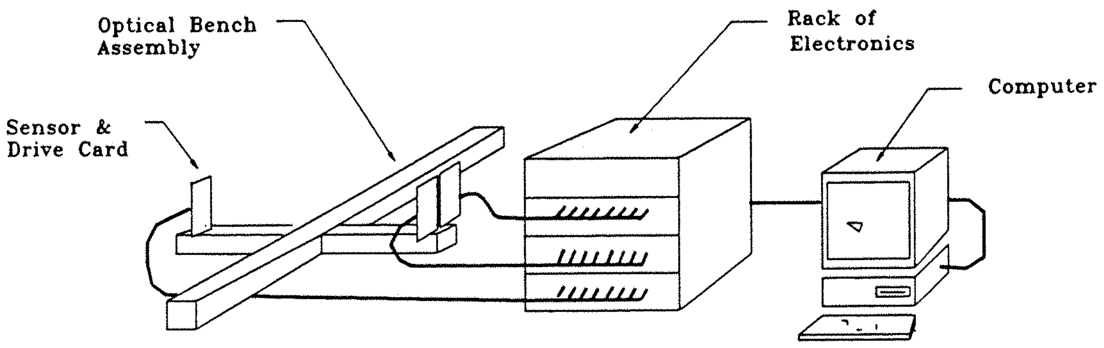


Figure 23: Layout of the electronic system.

consequently been mounted directly to the optical rig. Each drive card provides 16 video output signals which are connected by co-axial cable to the adjacent instrumentation racks. Each rack has 8 separate video cards for analogue-to-digital conversion, and each card holds 128 kilo-bytes of storage (figure 24).

The three sensors are driven by a common clock (f_c) and the line scanning cycles are synchronised to one of the drive cards to acquire the image data from all three cards simultaneously. The sensors integrate charge produced by incident light during the period that the previous scan is being shifted out of the device. The output shift register count was preset to 134 (128 pixels and 6 reference pulses) to make available reference video levels for use in drift and channel difference compensation. This results in an exposure interval of 22.3 μs for an f_c of 6 MHz. The photo-site to shift-register transfer period is preset for a duration of 4 clock periods (0.67 μs at 6 MHz), resulting in a maximum total line scan rate of 43 kHz. The saturation irradience requirements are consequently reduced to 31 mW. Although 1000 samples per segment are stored during the acquire cycle, the number of image samples acquired during the specified run of 20 ms is reduced by these settings to 860. This may be increased to 908 samples if required by removing the reference pulses by adjustment of switches mounted on the sensor drive cards (the corresponding global constant for reference elements should be changed in the software).

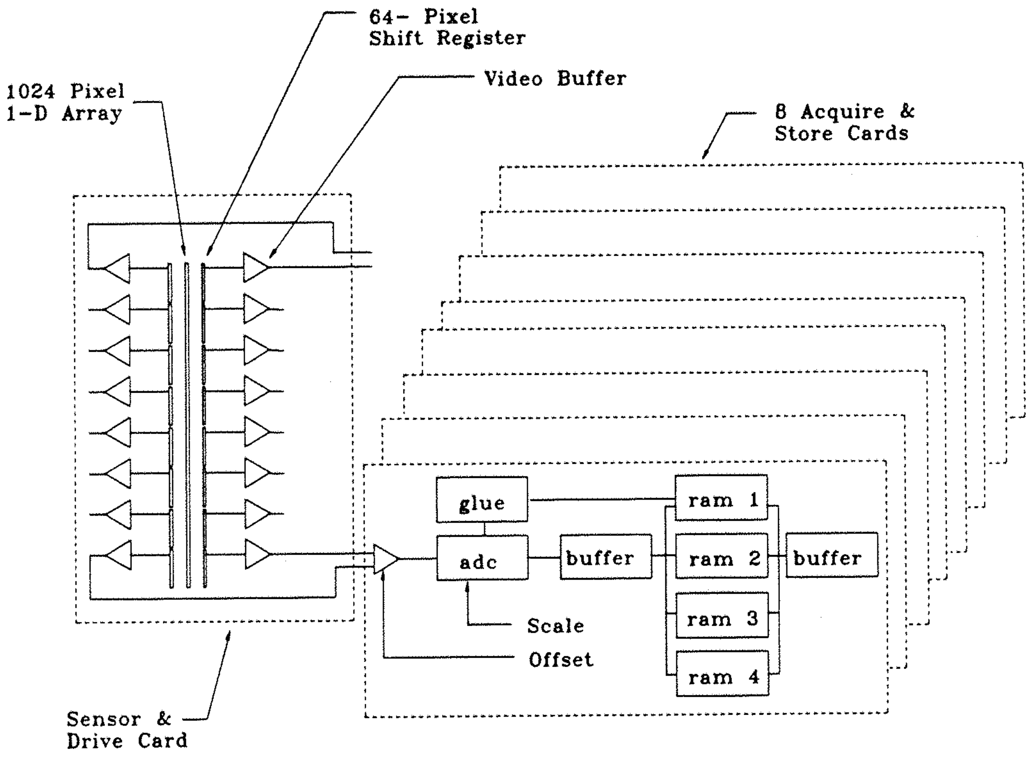


Figure 24: A block diagram of the sensor and acquisition cards.

10.3.2 Analogue Conversion

The choice of full analogue conversion results in a large amount of data that has to be stored during a run and later down-loaded to more permanent storage. Fixed video voltage thresholds have been used in similar shadowing systems to good effect[45, 46]. Fast real time pre-processing (filtering) and hardware based thresholding results in a massive reduction of data as the information stored can be reduced to just offset and shadow span pixel counts.

Basic thresholding works well for low noise signals but if noise is present in the image or there is ambiguity in edge shapes, multi-level information on the edge profile is essential.

Analogue to digital conversion allows the use of signal processing, making possible the recovery of centroid positions in poorly defined images. Fast memory and analogue conversion chips are now incredibly cheap and it is economically viable to use multiple channels to achieve high overall data throughput.

10.3.3 Outline of Circuit Operation

The basic electronic functions are illustrated in figure 25. (Abbreviated schematics are given in Appendix C and a more detailed description of the circuit operation can be found in reference [47]).

Programmable array logic devices (pals) have been used extensively to reduce component count throughout the system. The Boolean source listings for the pals can be found in Appendix D.

The three sensor drive cards produce three sets of video signals with 16 signals from each card. The 16 signals from each card are derived from shift register outputs which are arranged with a pair of registers for each of the 8 segments of a single array. The register clock signals for a pair of registers are phased so that the outputs are available at alternate half cycles of the system clock. The signals from a register pair can thus be mixed to double the data rate and halve the number of video channels required. The

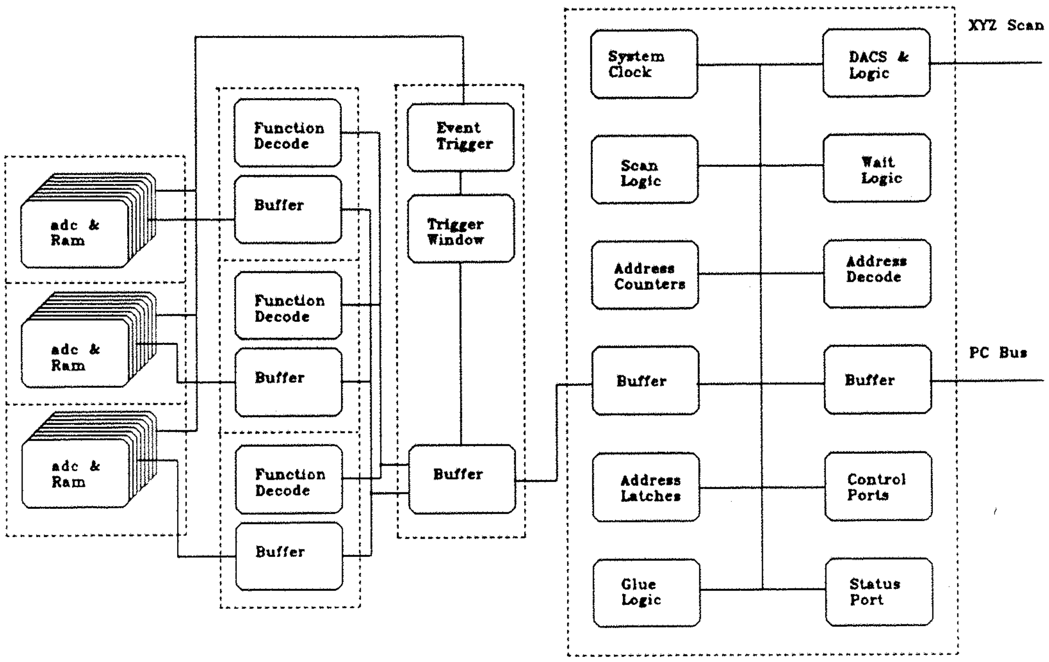


Figure 25: An overall block diagram of the electronics system.

mixing and translation of signal levels to suit the adc is accomplished by an amplifier on each acquisition card.

The combined video signals are converted to digital form by an 8 bit flash converter (adc) which operates in synchronism with the sensor array pixel clock. The data from the adc is stored in sequential addresses in the 128 kilo-bytes of static ram on each acquisition card.

The ram addressing, buffer direction control and general system house-keeping is achieved through the interface and control card, mounted within the computer. The system can be triggered to start the acquire and store cycle either by an external signal (acoustic or pressure indication of flow commencement) or by event triggered logic that can be armed to wait for a trackable image before starting the store cycle. The trackable image event is defined as occurring when the first segment of at least two sensors is illuminated while the second segment is under shadow. This would be the case for a model shadow that is moving downwards and is likely to produce the two shadow edges per sensor required for tracking flight history.

An additional function incorporated in the electronic design was direct video output of selected rack and acquisition card ram contents. While in the idle state (not sampling or under system interrogation), the system sequences through selected ram contents and feeds the results to a digital to analogue converter (dac) on the interface card. Two other dac's are synchronised to the line scan timing and provide ramps that can be used to produce line and frame drive signals on an oscilloscope. In principle, if the ram output dac is used to drive the brightness modulation on an oscilloscope (z-mod), and the two ramp signal used to drive the x and y deflection plates, a two-dimensional map is provided on the oscilloscope display that gives an instant impression of the validity of the data in the ram.

The operational statistics of the electronic system may be summarised as providing 8-bit quantisation on 48 signal channels, maximum input 2 volt peak-to-peak, synchronised at 6 MHz and producing a 23 ms duration of data sampling, totalling over 3 megabytes.

10.4 The Software

The system software has been written in Borland Turbo Pascal to take advantage of the excellent programme development environment, versatile graphics interface and source code intelligibility. The initial needs of the user are to acquire and store flight history data. Additional functions of value are data display, parameter adjustment and checks for system integrity. The task list grows considerably when functions to aid maintenance and debugging are added to the list (fig 26).

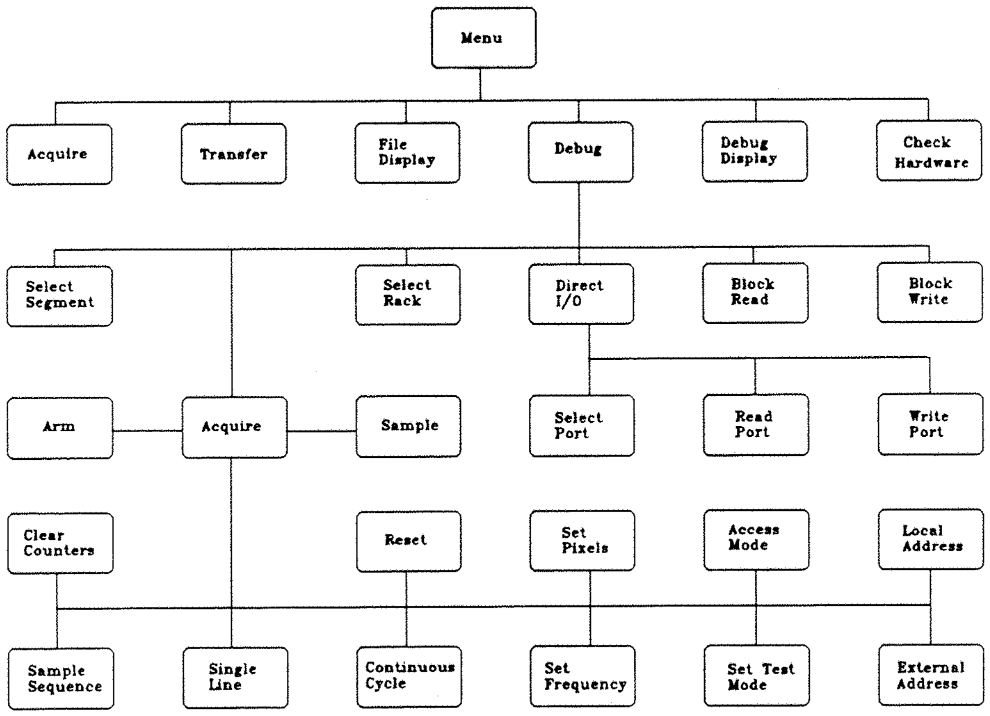


Figure 26: The software menu.

In the basic operating mode, the system is armed from the keyboard and acquisition arranged to start either automatically or on an external trigger level, in each case synchronised by the control logic to the start of a fresh line scan at the sensors. The acquire cycle, (lasting 23 ms at an f_c of 6 MHz), would normally be followed by an instruction to transfer and store the data from each rack in sets of reasonably sized files.

A generic name for the complete set of files is provided by the user. The programme automatically indexes the file extension to identify rack and file number. Four files of 250 lines are generated for each rack, taking approximately 3 minutes to transfer each file for a 8 MHz pc.

The data acquired from a run remains in the memory arrays until either the system is powered down or a fresh acquire cycle is made. The transfer process can be interrupted after storage of a suitable number of lines to a file to enable a rapid visual evaluation of the value of the acquired data. If the data looks useful the transfer can be re-initialised and completed. Two graphical display modes are provided (figures 27,28).

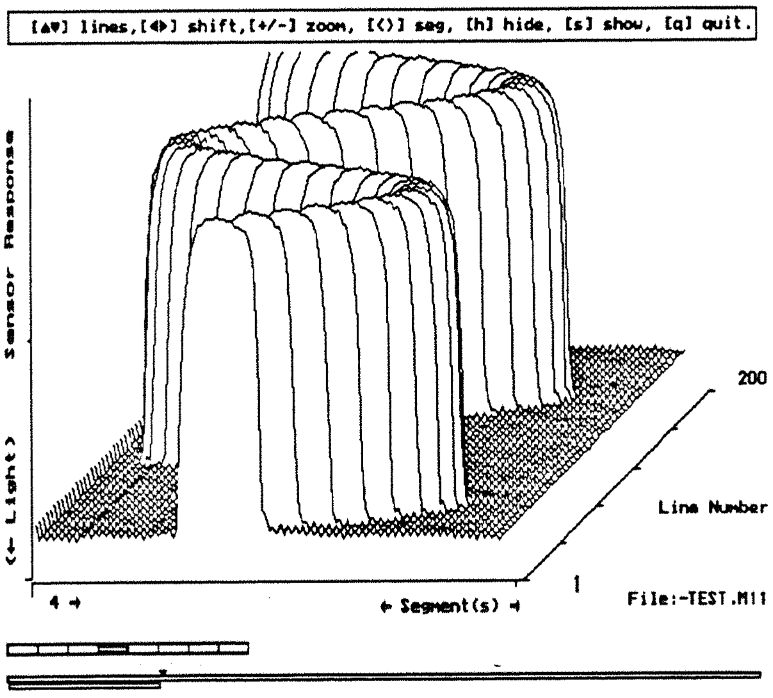
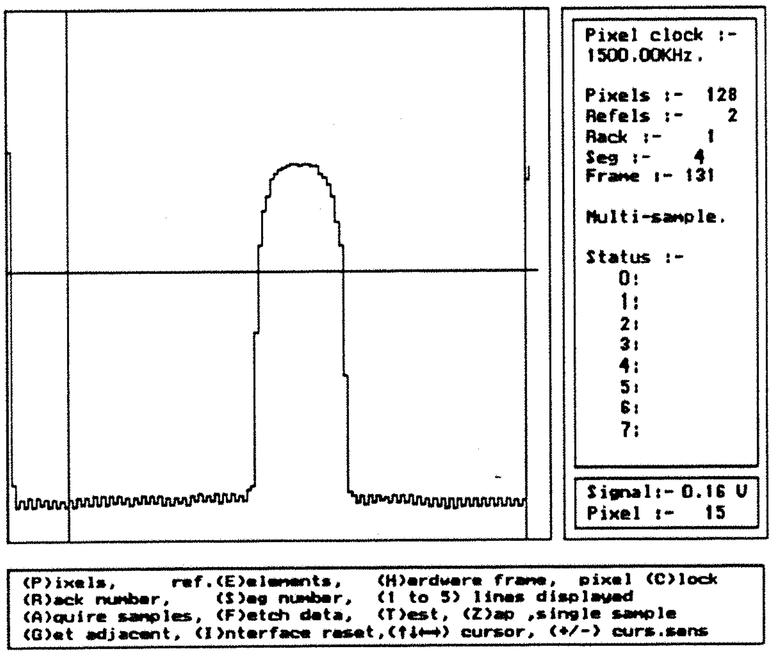

[▲▼] lines,[◄►] shift,[+/-] zoom, [<>] seg, [h] hide, [s] show, [q] quit.
<-- Light -->   Sensor Response
200
Line Number
1
4 →
← Segment(s) →
File:-TEST.M11

Figure 27: Typical FILE display.

Figure 27 represents the FILE screen display obtained from data in a file. The data is shown as a set of 50 line-scan traces. Successive scans represent a time series of sensor exposures, the interval between each scan being determined by the pixel clock, f_c. The vertical axis on the display represents sensor response with saturation level being near the base-line. The horizontal axis represents vertical displacement in the sheet beam. The software can display from 1 to all 8 segments in the array, starting from a selected segment, (indicated by the small partitioned segment *status* bar). The long bar represents the complete *data set* for a particular run (over 1 megabytes), the small pointer indicates the start position for the next display. The pointer may be moved by the user to display data from any point in the buffer. The bar below this indicates, by size relative to the *data set* bar, the percentage or *span* of the buffer being shown on the screen. The span can be increased progressively from 50 lines (each line of data displayed) to display samples of the complete run by skipping lines. The illustration in fig 27 was obtained at an f_c of 1.5 MHZ from a wire vibrating in the vertical plane at 125 Hz and shows segment 4 with the span increased to produce a displayed line interval of 372 μs.



```
Pixel clock :-
1500.00KHz.

Pixels :-  128
Refels :-    2
Rack :-      1
Seg :-     4
Frame :- 131

Multi-sample.

Status :-
    0:
    1:
    2:
    3:
    4:
    5:
    6:
    7:

Signal:- 0.16 V
Pixel :-   15
```

```
(P)ixels,     ref.(E)elements,   (H)ardware frame,  pixel (C)lock
(R)ack number,     (S)eg number,  (1 to 5) lines displayed
(A)quire samples, (F)etch data,  (T)est, (Z)ap ,single sample
(G)et adjacent, (I)nterface reset,(↑↓←→) cursor, (+/-) curs.sens
```

Figure 28: Typical DEBUG display.

The DEBUG display mode, shown in figure 28, allows immediate access to data and control of parameters such as pixel per segment count, reference pixels storage and pixel clock rates (adjustable from 6 MHz to 45 kHz). The trace represents a single segment in one of the three sensor arrays and data is accessed directly from the video memory as opposed to using file data as in the FILE display mode. The horizontal cursor can be adjusted to provide a voltage reference datum and the vertical cursor can be moved to measure individual pixel signal levels. The current pixel number and voltage level are displayed below the parameter status box. The display in figure 28 shows the shadow profile of a thin wire (1.6 mm diameter).

An annotated listing for the Pascal source is given in Appendix E. A modular approach has been used consistently to allow for modification and recombination of software modules to suit requirements. File storage is arranged with data in indexed records to allow a particular pixel in a particular line to be accessed unambiguously: $data = line[n_1].segment[n_2].pixel[n_3]$

where the index limits are $0 < n_1 < 1001$, $0 < n_2 < 9$, $0 < n_3 < 129$.

A file header is stored with the data to preserve information about operational variations. This header includes a storage mode byte to allow for accommodation for subsequent software developments. For example, the data could be smoothed or run-time encoded to reduce storage requirements: the mode byte would be set to indicate this and the display software would execute an appropriate method of display.

Chapter 11

System Characteristics

The characteristics of the flight history record obtained using linear arrays can be expected to have complex characteristics. Discrete sampling of the image occurs not only in a time sense, but also in separate 1-dimensional regions within the field of view. The scale of deviation from a true flight history is best observed in a practical demonstration but some of the influencing factors are listed as follows:

- Uniformity in laser beam intensity profile.
- Uniformity in sensor pixel response.
- Interference patterns due to interactions with optical surfaces.
- Variations in the shadow span of the unfocussed image: accuracy of centroid prediction varying with statistical sample size.
- Electronic noise levels.
- Model position and attitude.
- Model velocity.
- Presence of yaw motion.
- Mechanical vibration.
- Variable refraction in the tunnel due to gas density variations.

Other predominantly predictable influences on the flight record characteristics are:

- Variation in channel gain/ offsets, differences in analogue to digital converter performance and component matching.

- Variation in exact instant of flash conversion due to different propagation path lengths along the rack bus. Although this time skew is only a few nanoseconds, the resultant effect on the sampled signal can be significant in that system correlated noise present in the signal consists mostly of very short pulses or spikes, which can easily be missed or sampled, dependent on a small change in timing.
- Geometric Distortion in the lenses.
- Lens/ sensor resolution.
- Beam alignment.
- Model geometry.

The impact of the predictable factors listed can be minimised by careful design and by optimising beam alignment. Residual effects, such as minor geometric distortion can be compensated for by calibration. The variable influences are much harder to quantify and the function of the bench top rig to was signify some of these factors.

Chapter 12

Bench-top Tests on the Prototype Rig

The observations made on the prototype bench-top rig were intended to verify the operational principle and reveal potential problems. One complete rack of analogue conversion cards was available, enabling tests to be made on one of the three linear array-video systems. The electronics for the remaining two racks would be identical and similar results would be expected, (the racks are individually buffered to prevent any anomalous interaction). The weakest of the three sheet beams was used for the sensor tests. The distribution of the laser source output for the three beams was estimated at 13.6% for the test beam and 14.7% and 30.1% respectively for the other two beams (Appendix B). A digital storage oscilloscope was used to record and plot some of the analogue waveforms but most of the graphic results were obtained via the tracking system using the software developed to support the rig. A screen-saving programme was used to capture and preserve the scale of images produced by the FILE and DEBUG display functions.

12.1 Sensor and Electronic System Behaviour

A warm up period of at least 15 minutes was required for stable results. The laser output power would normally take this time to settle. The sensor arrays stabilise at a moderately high temperature but no significant drift was encountered after allowing a brief warmup delay.

12.1.1 Noise

The video channel functions were checked by applying various test signals to each input. Preset steady voltage levels were applied while observing basic system noise and checking channel gains and offsets. Noise was examined at f_c of 6 MHz and 3 MHZ using the DEBUG display mode to find the maximum and minimum pixel levels recorded by the system for the steady state input. Maximum noise levels were apparent at $f_c = 6$ MHz, and at input levels where maximum bit changes were occurring at the adc output. The worst of these events occurs when the input level causes a change in binary output from 01111111 to 1000000, which produces glitches resulting from the change in logic current demands. The resolution of the analogue converters was 4 mV and measured noise levels were from 10 mV to 40 mV. For typical video signals this represents a signal to noise ratio of 32 db. The use of saturation exposure and the rapid shadow edge transitions resulting from the use of imaging reduces the significance of noise for the focused model shadow position. The unfocussed shadow has a more gradual transition but the centroiding process used on this data provides some statistical reduction in the impact of noise.

12.1.2 Channel Gain and Offsets

The channel gains and offsets were observed by noting the voltage levels required to produce saturation and cut-off levels on the DEBUG display for each channel. The variation in gain between channels was less than 2.5 % with the exception of channel 6 which deviated by 8.5 %. The maximum difference in offsets was measured at 200 mV. These variations can be seen clearly in the figures discussed in section 12.1.5.

12.1.3 Test Signals

Triangular and square wave signals (1.7v peak-peak at 100kHz) were applied to the inputs. An example of the results is shown in figures 29 & 30.

12.1.4 Video Signals

Examples of the video signals at various stages in the acquisition process are shown in figures 31 & 32.

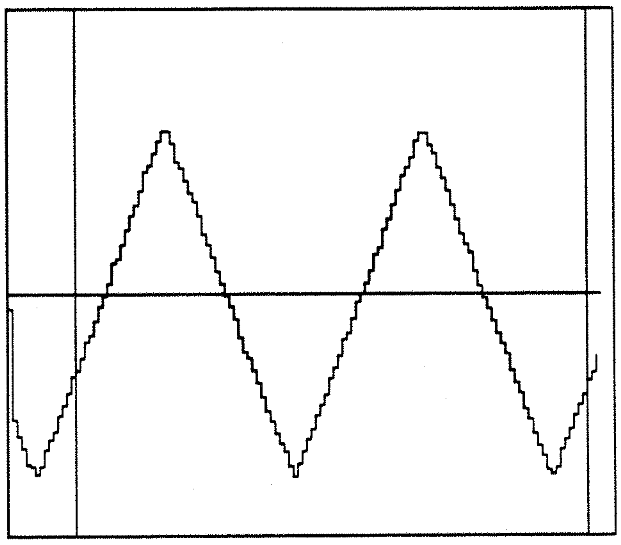


Figure 29: DEBUG display for triangular wave input.

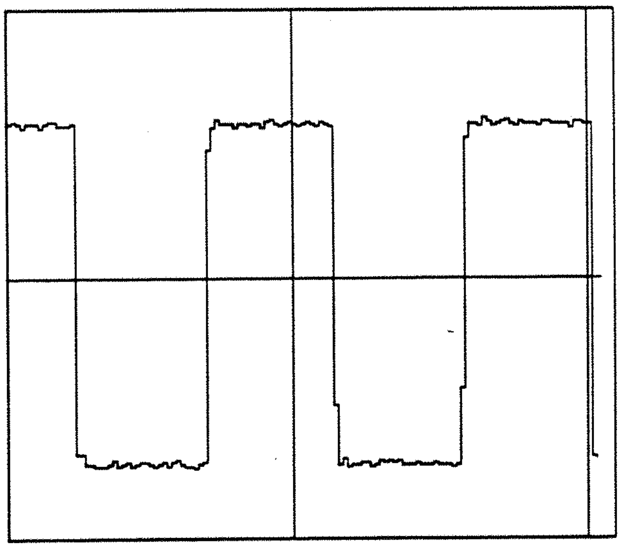


Figure 30: DEBUG display for rectangular wave input.

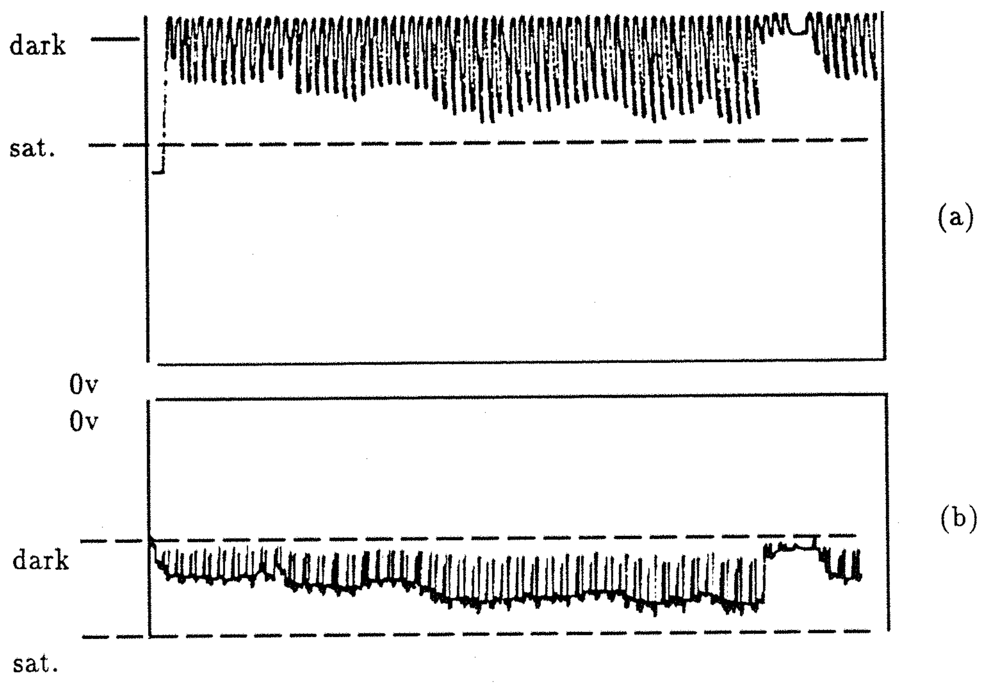


Figure 31: Typical video signals from non-saturated sensor at f_c = 6MHz. Oscilloscope signal from (a) a video shift-register output and (b) from the video mixer at the ADC input.

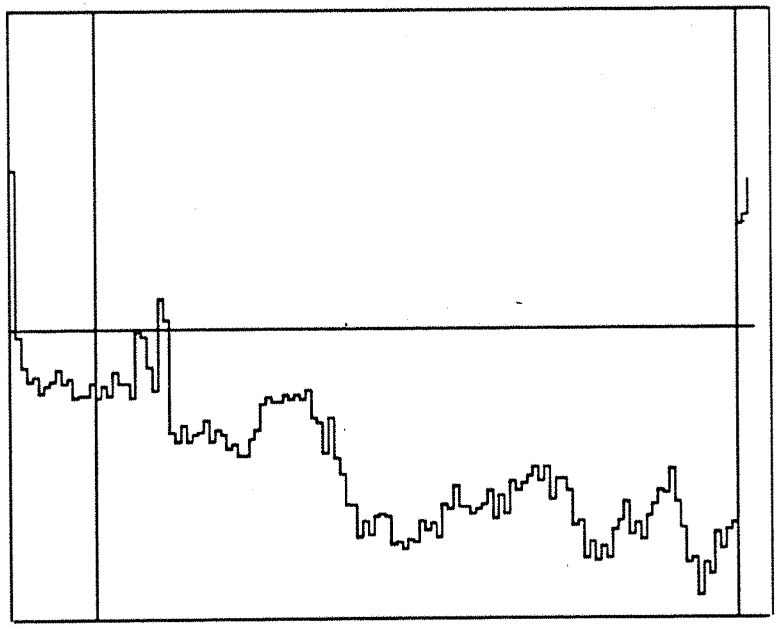


Figure 32: Debug display for non-saturated sensor

Figure 31(a) shows a digital oscilloscope trace of a video output signal from one of the 16 on-chip shift registers in the sensor. The pulse height as measured from the wave-form base line (top of the trace) is proportional to the incident illumination. There are sixty four valid signal pulses and three extra pulses that can be used to give a dark level reference (see section 10.3 for a description of the sensor mechanism). The brief steady level following the reference pulses represents the period in which the charges accumulated at the array of photo-sites during the serial transfer are loaded into the analogue shift-register, ready for the next serial output cycle. A ragged beam intensity profile can be observed in the envelope of the 64 pulses. The level that would be produced by saturation exposure is shown as a dotted line. Shift register outputs are combined in interleaved pairs (128 pixels) to result in a wave form that follows the pulse height as shown in Figure 31(b). This trace represents the signal presented to the analogue converter (adc) by the input mixer amplifier. The signal trace has in fact been inverted to clarify the correlation of the wave-shape to the envelope in figure 31(a). Glitches can be seen where pulses from one shift register interleave imperfectly with the pulses from the other shift register. The adc is clocked to sample the waveform at the centre of each pulse position, away from the region of the glitch. The digitised and reconstructed waveform is shown in figure 32. The dynamic range of the adc is optimised to accommodate an input signal range that fits within the dark and saturation levels shown in figure 31(b) so the intensity profile variations seem enhanced in figure 32.

The noise on the video signal is produced predominantly by ambient electronic logic activity and is thus synchronised to the system clock. This may be seen from figure 33 which shows an expansion of the waveform during the shift-register loading period. The upper traces (a) & (c) are from a pair of shift-register outputs which were mixed to produce the envelope waveform (c). Because of system noise, slight changes in the precise moment of sampling the analogue signal can have a significant effect on the noise in the sampled signal. This may be an explanation for the varying noise levels observed for the various channels. Also visible from these traces is the delay in the stabilisation of the reference pixels and the first two pixels in the next cycle. These

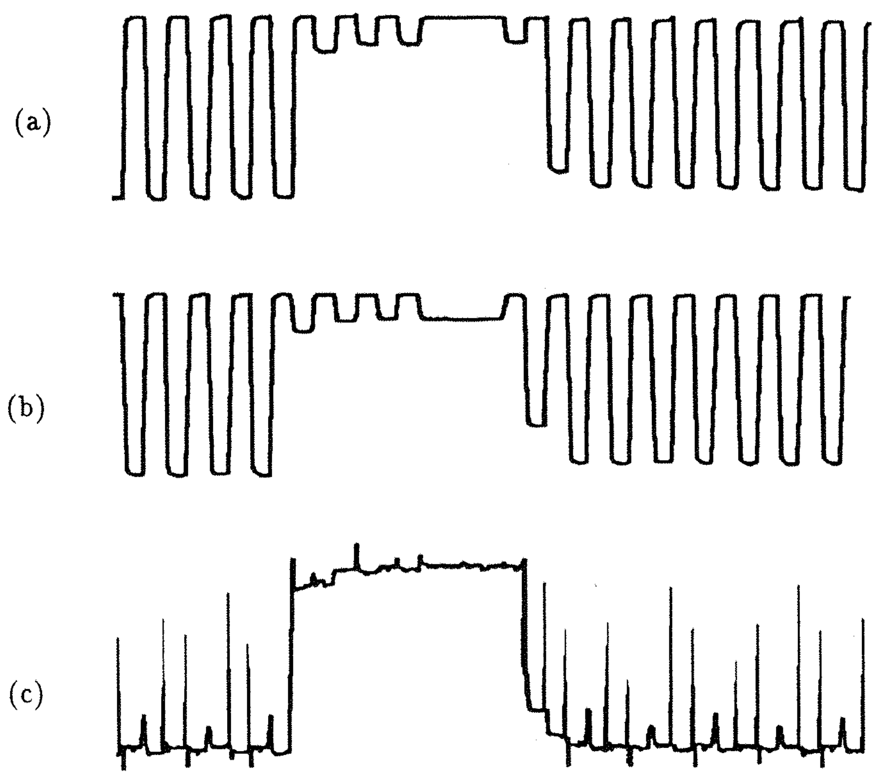


Figure 33: Detail of video outputs and ADC input. Oscilloscope traces of sensor output for segment 3, (a) even numbered shift-register output, (b) odd shift-register output, and (c) mixed video presented to the ADC for a f_c of 3 MHz.

effects persist at all illumination levels and thus are not thought to be linked to the saturation behaviour problems discussed in the next section. At this stage in the tests a subtle timing anomaly was noticed that results in the first pixel in each line scan to be stored with a low value (e.g. the first pixel in figure 28). The impact of this problem is not too severe and is discussed further in the concluding remarks in section 12.7. (The software used for the FILE mode display has subsequently been modified to ignore the first pixel in each segment to avoid obscuring the illustrations).

12.1.5 Response Uniformity

The sensor response to uniform illumination from a miniature fluorescent lamp is shown at various f_c's in figures 34 & 35.

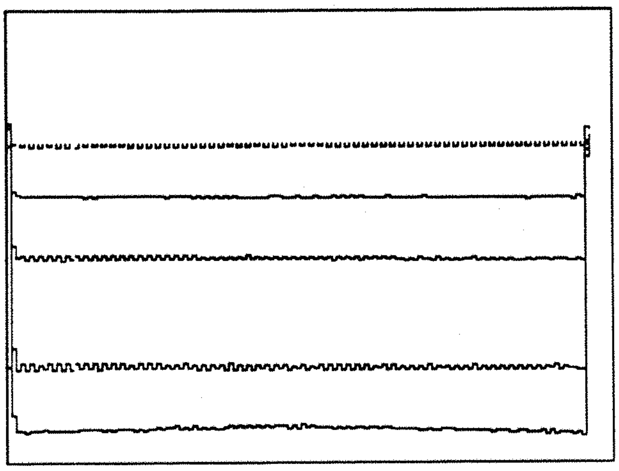


Figure 34: DEBUG display of sensor segment response to uniform illumination with f_c varying from 1.5 MHz to 46.87 kHz. The traces progress down towards the saturation level with decreasing f_c.

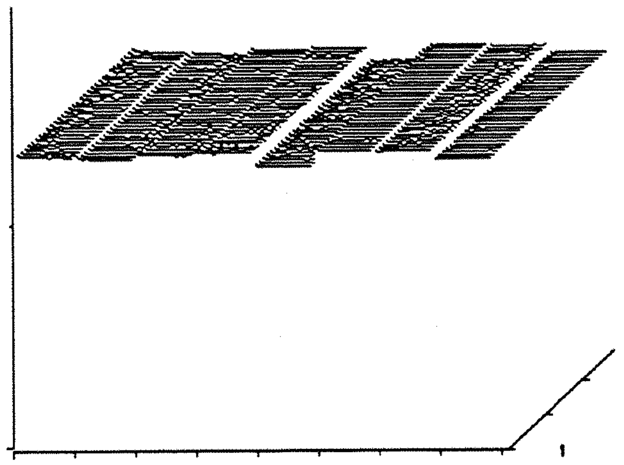
The response can be seen to be flat for any given segment until saturation is approached, when a distinct curve in the response profile is visible. The exact mechanism for this effect is not clear. The sensor is more sensitive to ambient light at the low sampling frequency but the problem is thought to relate to the charge handling characteristics of the device when operated near saturation. Significantly, figures for pixel uniformity provided by the sensor manufacturer are specified at 50% saturation[42]. Detail of the shift-register loading period (figure 36) indicates another characteristic occurring under saturation conditions. Trace (a) shows the sensor behaving normally under moderate saturation, while trace (b) shows that the reference pulses, (usually in a charge state representing the dark level), now appear saturated. The saturation level takes on a distinct distortion when the exposure is increased further as in figure 36(c). The effect on the perceived model shadow is indicated in figures 37,38 which represent the focused shadow cast by a cylindrical model for various values of f_c. Each successive frame represents a doubling in exposure interval, so taking figure 37(a) to represent the sensor just on the verge of saturation, the exposure doubles for each successive illustration, through to figure 38(c) which represents over-saturation by a factor of 32. In each frame, the horizontal cursor has been located at the point where the width of magnified image corresponds to the precise width of the model. The anomalous behaviour of the sensor with over-saturation can be seen clearly in figure 38(c) where the shadow is no longer visible in the sensor output. The width of the shadow determines the levels of local charge excess within the sensor so thin shadows, allowing incidence of more light, are more easily obscured than broad shadows under these conditions, (operating conditions are discussed further in section 12.7).

Some time was spent investigating these anomalies and attempting to optimise operation of the sensor with saturation exposure. The three areas on-chip that could be causing problems in handling the excess charges produced under these conditions were:

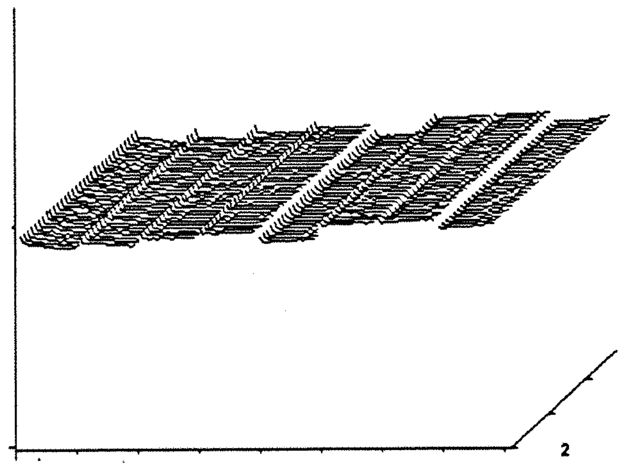
- the charge collection sites (the photo-sensitive regions)
- the charge transport registers
- the output charge/detector buffer stage

Anti-blooming gates are provided within the sensor[42] and these should dump excess charge building up at the photo-sites. Indirect charge injection into the analogue shift-registers was thought a possibility but experiments using an extended photo-site reset period to dump some of the charge reaching the output sense amplifiers suggested that the output stages were being overloaded. Optimum performance under saturation exposure was obtained by adjusting the output gate bias from $5.0V$ to $2.5V$. Further comments on significance of these findings is made in the discussion in section 12.7.

(a)



(b)



(c)

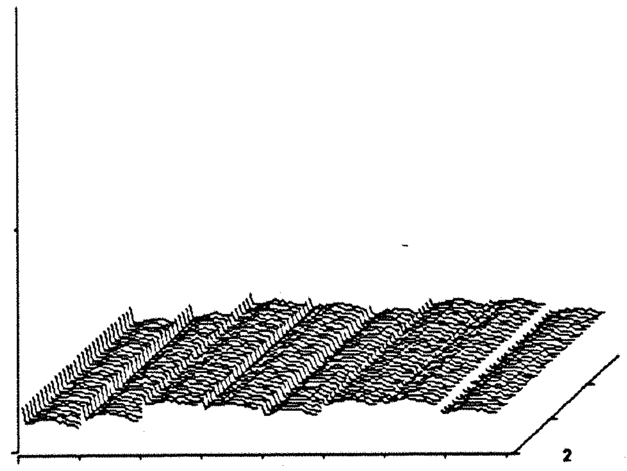
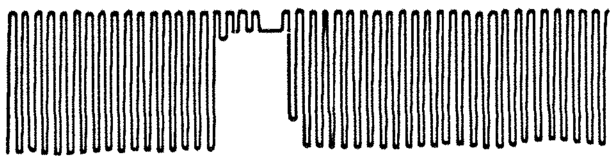
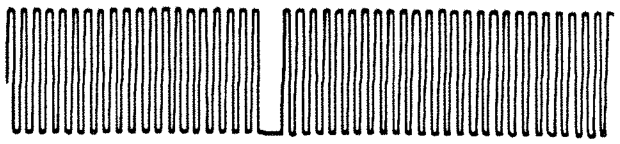


Figure 35: FILE display of sensor array response to uniform illumination at (a) $f_c = 750$ kHz,(b) $f_c = 187.5$ kHz and (c) $F_c = 46.87$ kHz.

(a)



(b)



(c)

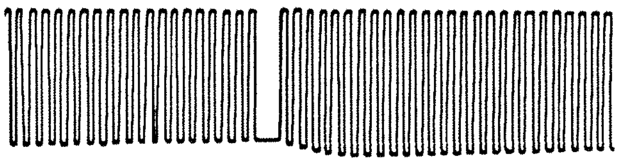


Figure 36: Sensor behaviour under saturation conditions. Oscilloscope traces of the sensor output, video tap 3, segment 2, at (a) f_c = 375 kHz, (b) f_c = 187.5 kHz and (c) f_c = 93.75 kHz.

(a)

(b)

(c)

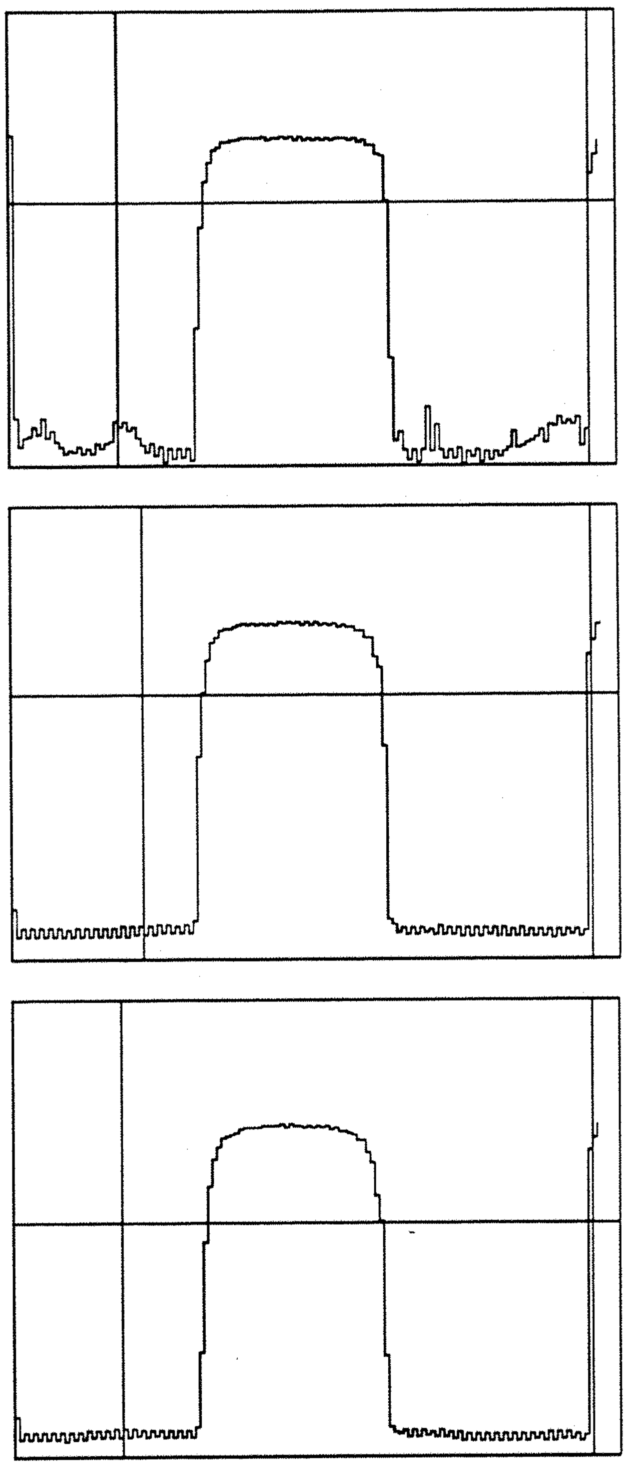


Figure 37: DEBUG display of shadow width for various f_c values: (a) $f_c = 6$ MHz, (b) $f_c = 3$ MHz and (c) $f_c = 1.5$ MHz.

(a)

(b)

(c)

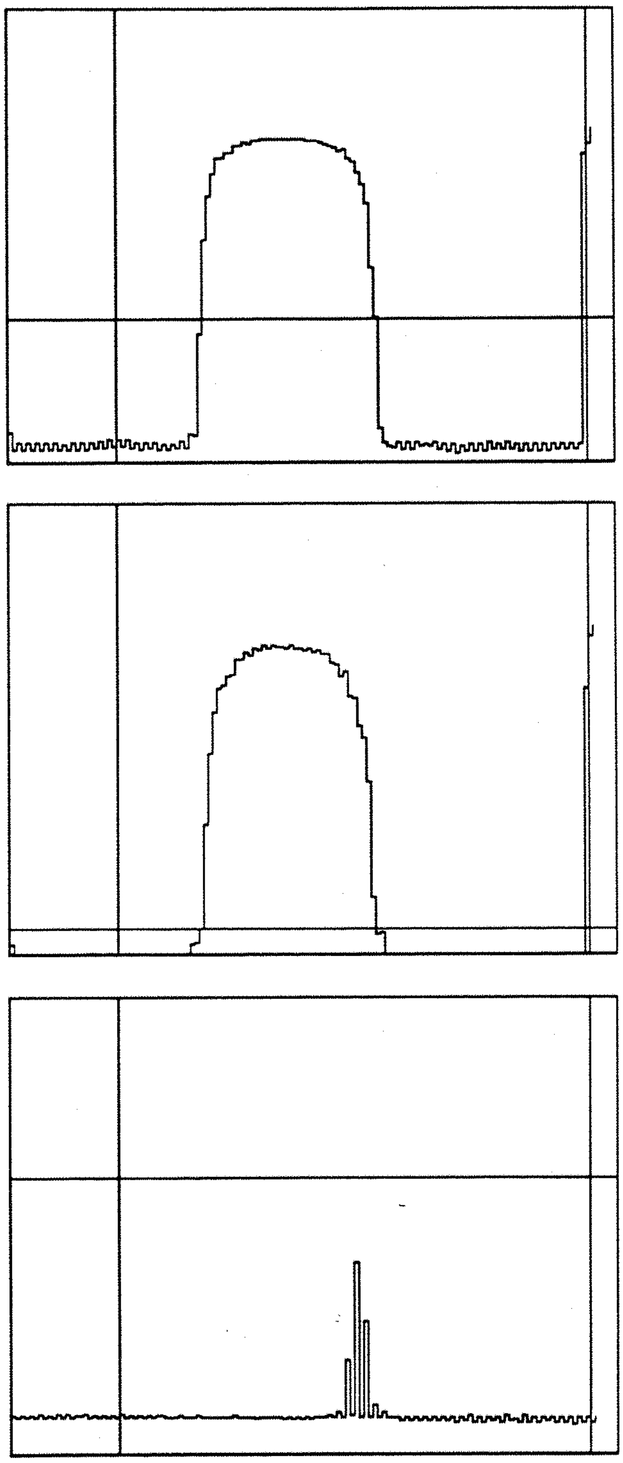


Figure 38: DEBUG display of shadow width for various f_c values: (a) $f_c = 750$ kHz, (b) $f_c = 375$ kHz and (c) $f_c = 187.5$ kHz.

12.2 Lateral Displacement

The change in apparent shadow width of a 2.0 mm model was observed for two positions 62 mm apart with the model in the focused object plane (figure 39). The difference in width was observed to be 1 pixel (0.07 mm).

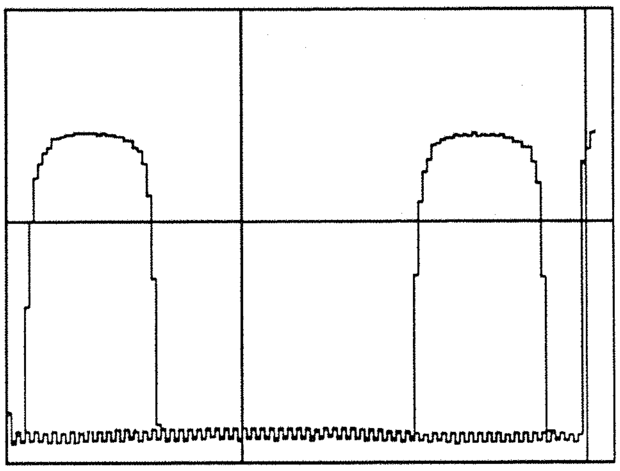


Figure 39: DEBUG display of shadow width for two model positions perpendicular to the optical axis at the focused object plane.

12.3 Depth of Field

The change in apparent shadow width of a 2.9 mm model was observed for three positions along the optical axis. Figure 40 shows the shadow for the model displaced from the focused object plane by (a) +30 mm, (b) 0 mm and (c) −30 mm. The horizontal cursor is positioned at the point in the image corresponding to the magnified object width. Over the 60 mm range of movement, the width was observed to change by 1 pixel, equivalent to 0.07 mm in the object plane.

(a)

(b)

(c)

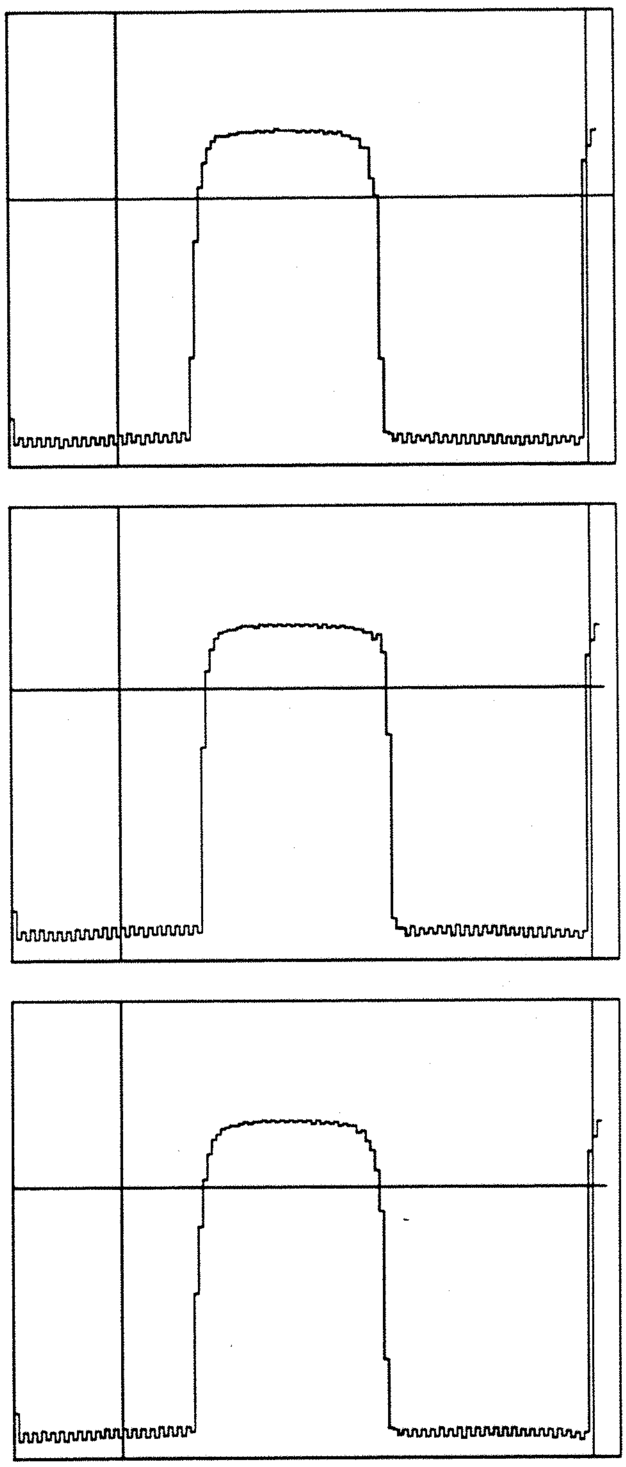


Figure 40: DEBUG display of shadow width for various model positions along the optical axis from focused object plane: (a) −30 mm, (b) 0 mm and (c) +30 mm.

12.4 Models in Motion

The displays in figures 41 to 46 demonstrate graphically the quality of images observed for moving models. Figure 41 was produced by progressively moving a conical model (10 deg 1/2 angle) into the sheet beam. The model moves forward 0.5 mm during each line scan for 30 or so lines and then remains static.

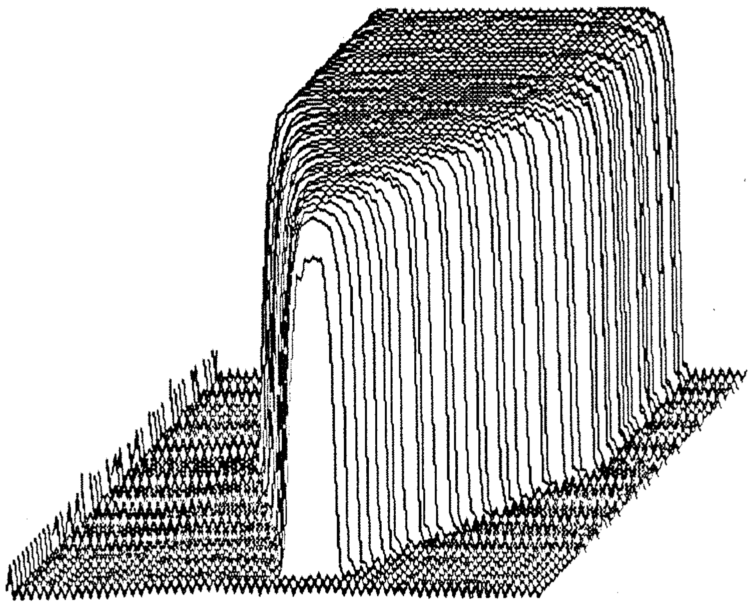


Figure 41: FILE display of shadow displacement history for a conic model moving into the beam at a constant rate and then remaining static. The data is from one segment in the array.

Figure 42 shows part of the motion history of a 1.6 mm diameter wire vibrating at 100 Hz in the focused object plane. The adc sampling frequency was 3 MHz, resulting in a line scan rate of 21.58 kHz. The display was generated by skipping every other line so the interval between lines was 92.7μs.

Figure 43 shows the corresponding motion history for the same wire positioned at the unfocussed object plane, where the beam would re-cross the wind tunnel working section.

The illustrations in figures 44, 45 were made by a wire rotor attached to an electric motor mounted adjacent to the sheet beam at the focused object plane. The motor axis was parallel to the optical axis and separated by 22 mm, the rotor being 80 mm in span and made from 1.6 mm wire. The view shows the continuity of indicated wire positions across the separate segments, with little impact from the apparent differences in levels corresponding to saturation. The shadow image can be seen to get progressively wider as the rotor/ sheet beam interception angle becomes increasingly oblique.

The final illustration of model motion is that of a pointed wire attached to a transducer (figure 46). The wire was being projected rapidly into the sheet beam at approximately 100 Hz.

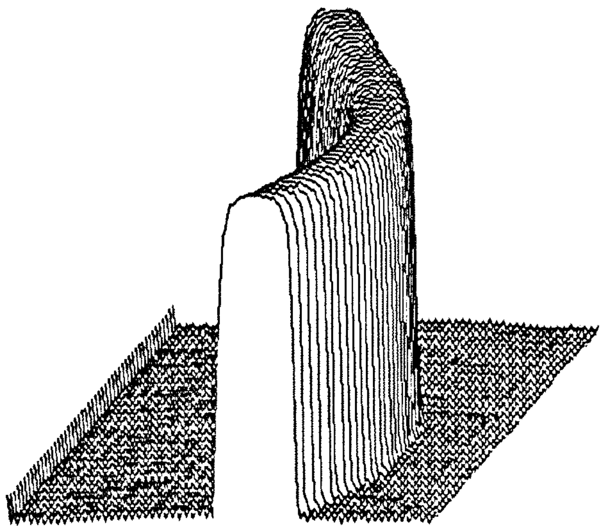


Figure 42: FILE display of shadow displacement history for 1.6 mm dia. wire vibrating at 100 Hz in the *focused object plane.*

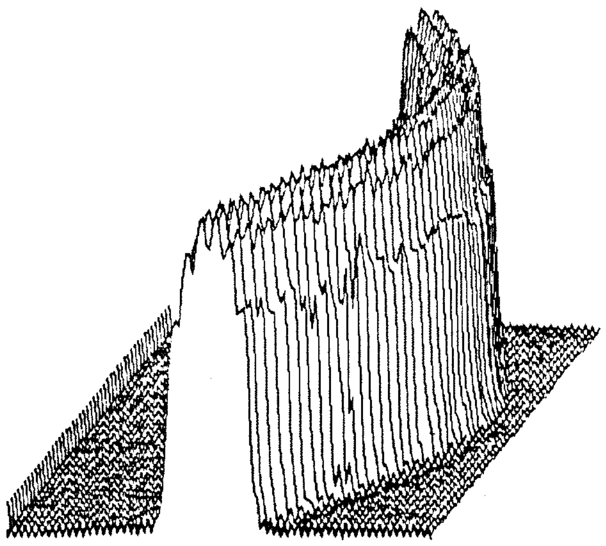


Figure 43: FILE display of shadow displacement history for 1.6 mm dia. wire vibrating at 100 Hz in the *unfocussed object plane.*

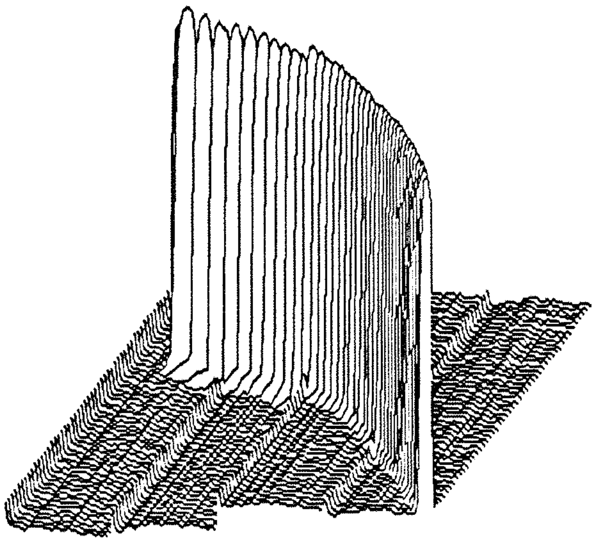


Figure 44: FILE display of shadow displacement history for a 1.6 mm dia. wire rotor moving in the focused object plane. The view spans 4 segments and the interval between line scans is approximately 93 μs.

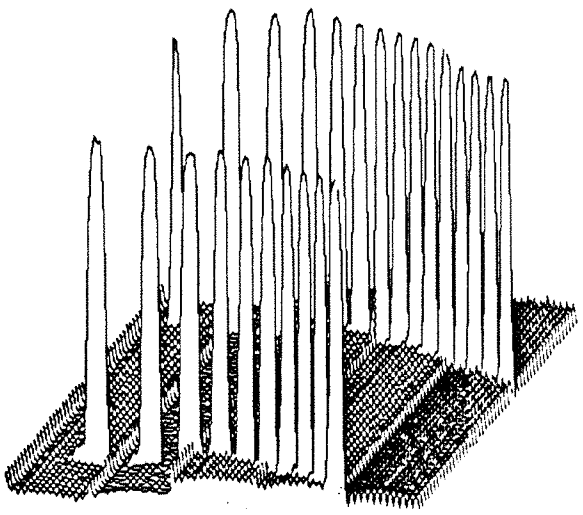


Figure 45: FILE display of shadow displacement history, detail for a 1.6 mm dia. wire rotor moving in the focused object plane. One in every 4 lines stored in the data file is being displayed, resulting in a line interval of 370 μs.

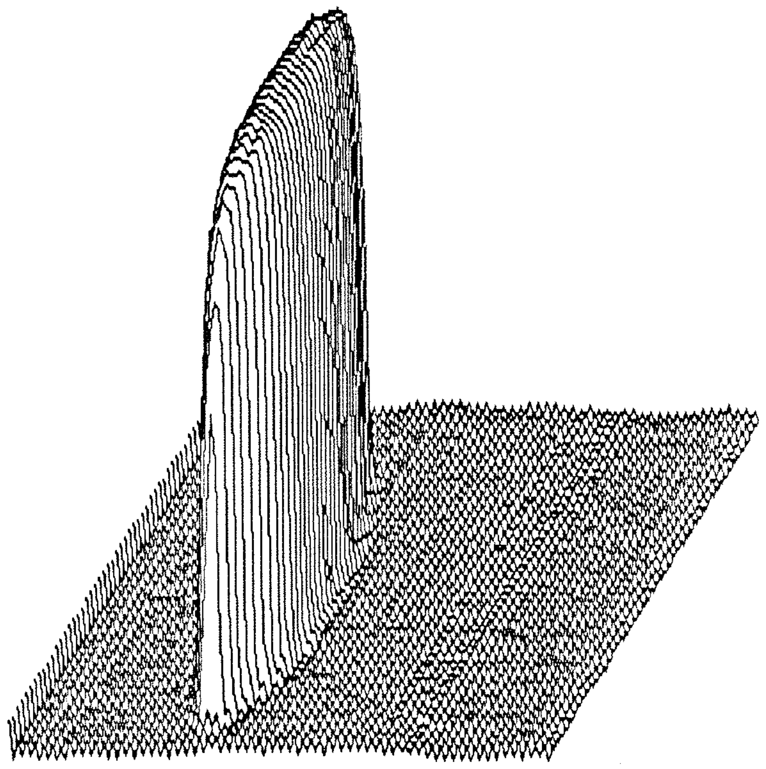


Figure 46: FILE display of shadow image history for an intruding wire model (1.6 mm dia).

12.5 Beam Profile

The problems produced by over-saturation observed in section 12.1.5 underlined the importance of the laser beam profile. The ideal beam shape would be flat-topped and the irradience level set to be just sufficient to saturate the sensor. The following beam profile illustrations were modified for clarity to show the dark level as a baseline (normally towards the top of the FILE mode display). Figures 47 & 48 were made by directing the beam (without intervening lenses) onto the sensor at a range of 1 meter. The sensor was moved across the beam to sample the intensity profile at discrete positions with a 0.1 mm intervals. The displays span three sensor segments along the baseline. Figure 48 represents a slice through figure 47. The useful width of the beam (taken as the level just below the deep trough in the intensity shown in figure 48) was estimated at 4.2 mm.

The same method of lateral sensor displacement was used to observe the *sheet beam* profile at different pixel clock frequencies, (figures 51 to 56). The displacement interval in this case was 0.02 mm, giving a 'depth' of 1 mm to the display. Figures 49 & 50 compare the sensor just out of saturation for an f_c of 6 MHz with the sensor for the most part in saturation at an f_c of 3 MHz. The ragged slopes of the profiles highlight the obscure structure present in the beam profile and the need to operate the sensor with saturation exposure. The oval shape of the sheet beam can be clearly seen at the top of the profile in figure 50. Figures 51 to 54 show an expansion of the intensity profile at the end segments. Figures 55 & 56 show the situation in a segment at the beam centre. The profile in figure 56 represents the ideal state of a uniform, flat topped beam in this region. The beam width has been reduced by the convex collimation lens and imaging lens to approximately 0.25 mm. The unmodified beam divergence at this distance from the source would have been approximately 8 mm.

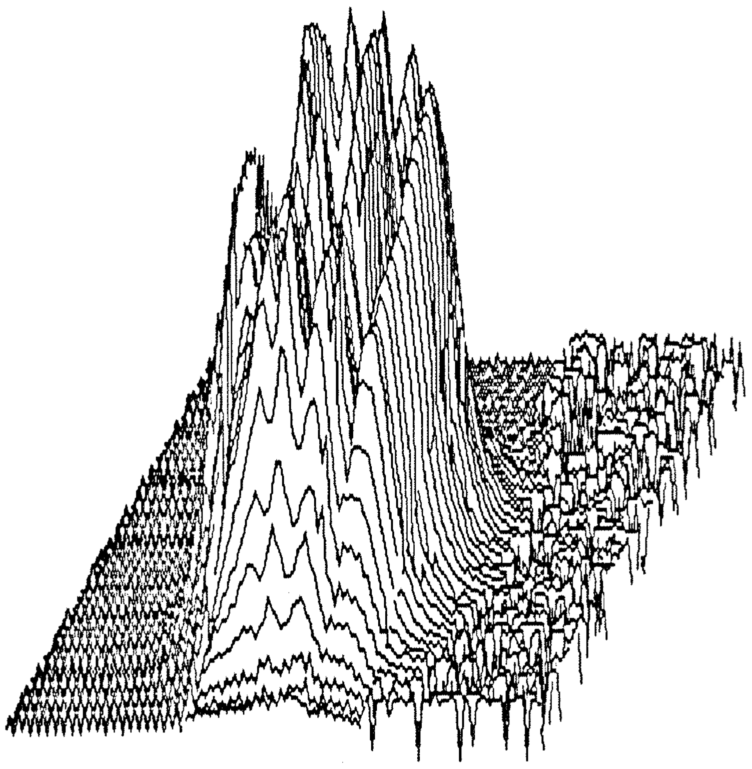


Figure 47: Laser beam intensity profile sampled at 0.1 mm intervals at a distance of 1 meter from the laser source. The view spans 3 segments.

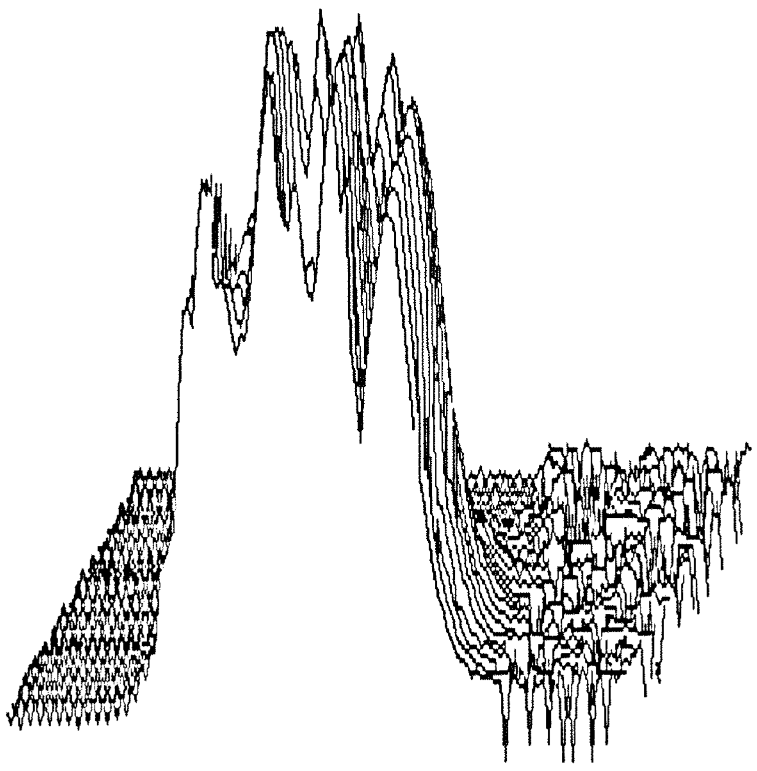


Figure 48: Central section of laser beam intensity profile.

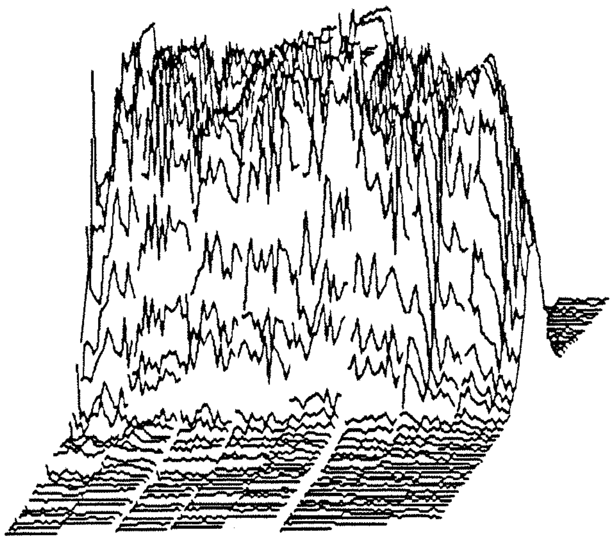


Figure 49: Sheet beam profile across a single array at f_c = 6 MHz, (showing 50 line scans for 8 segments over a displacement of 1 mm).

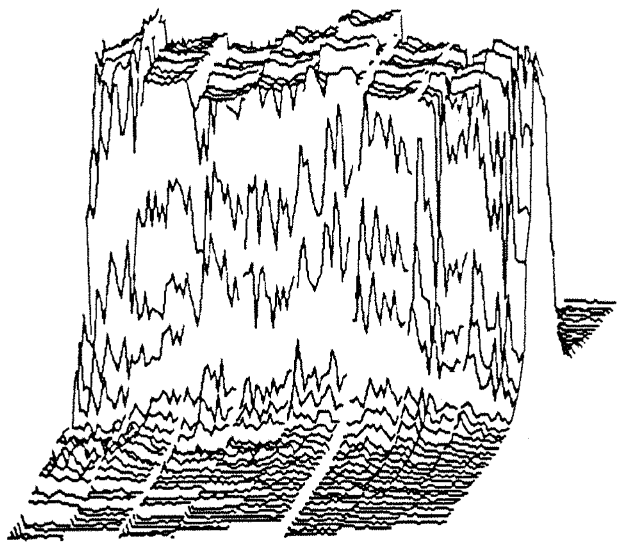


Figure 50: Sheet beam profile across a single array at f_c = 3 MHz, (showing 50 line scans for 8 segments over a displacement of 1 mm).

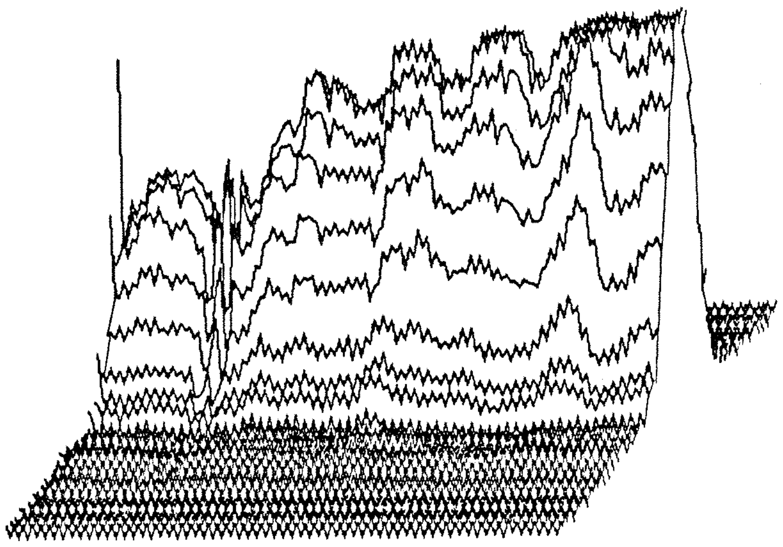


Figure 51: Sheet beam profile across the first segment (# 1) at f_c = 6 MHz.

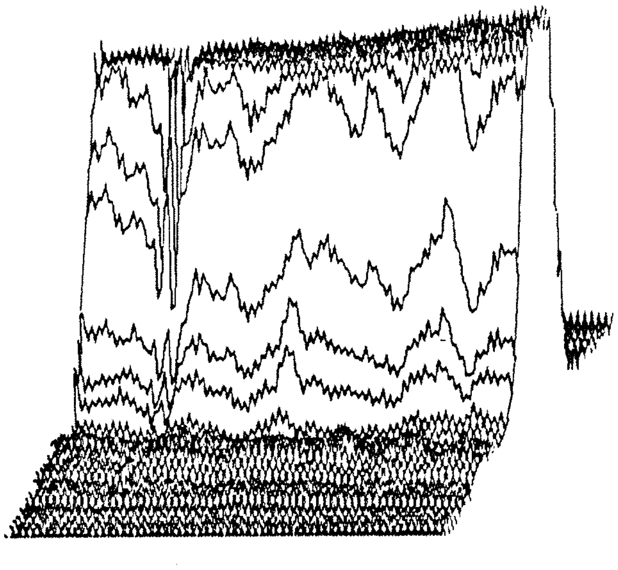


Figure 52: Sheet beam profile across the first segment (# 1) at f_c = 3 MHz.

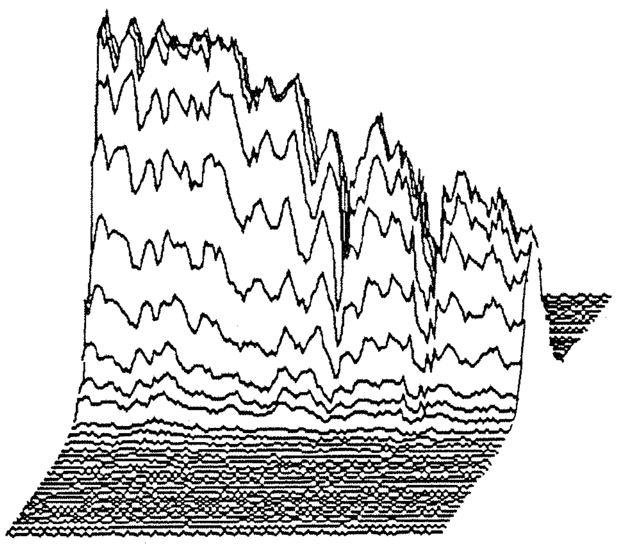


Figure 53: Sheet beam profile across the last segment (# 8) at $f_c = 6$ MHz.

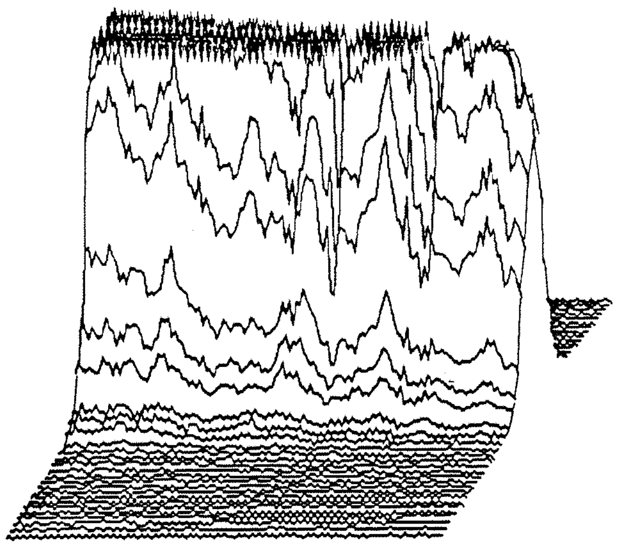


Figure 54: Sheet beam profile across the last segment (# 8) at $f_c = 3$ MHz.

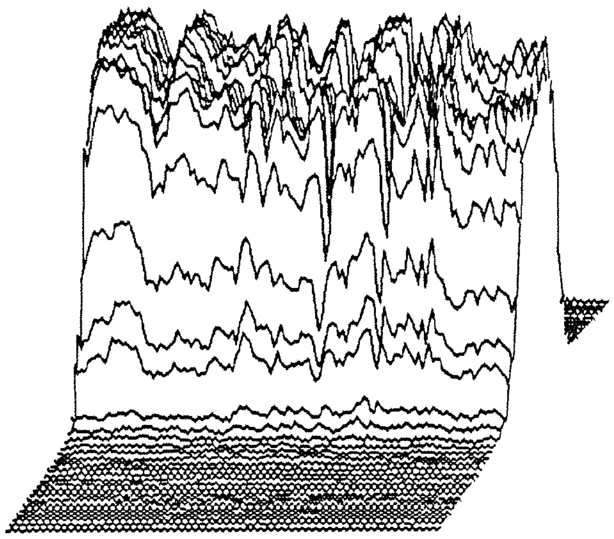


Figure 55: Sheet beam profile across a central segment (# 5) at f_c = 6 MHz.

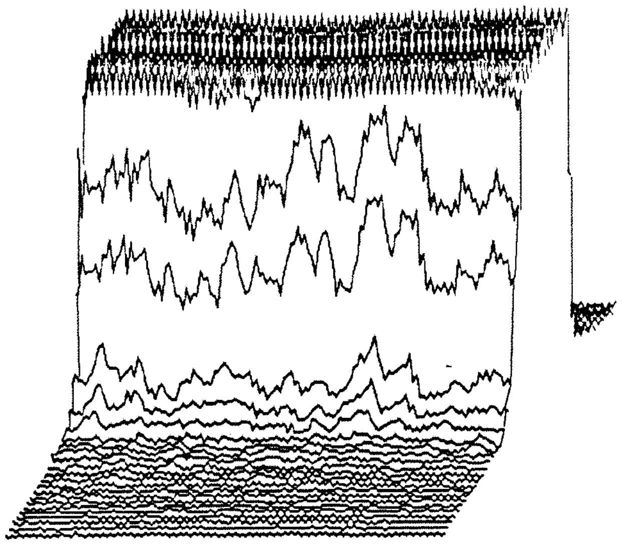


Figure 56: Sheet beam profile across a central segment (# 5) at f_c = 3 MHz.

12.6 Beam Alignment

An important aspect of the bench tests was to prove the practicality of the optical design. The beam adjustments required are shown in figure 57. To assist in calibration,

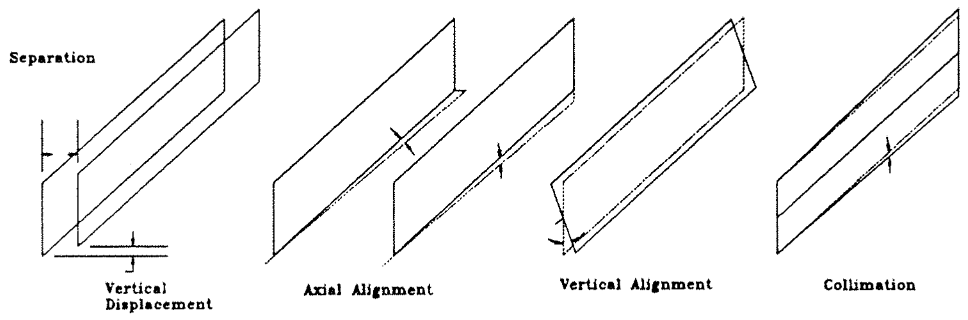


Figure 57: The beam adjustments required.

two plates were placed either side of the active scanning region (figure 58). Eighteen holes were drilled in the plates to coincide with the precise locations required for the six sheet beams (three holes per beam).

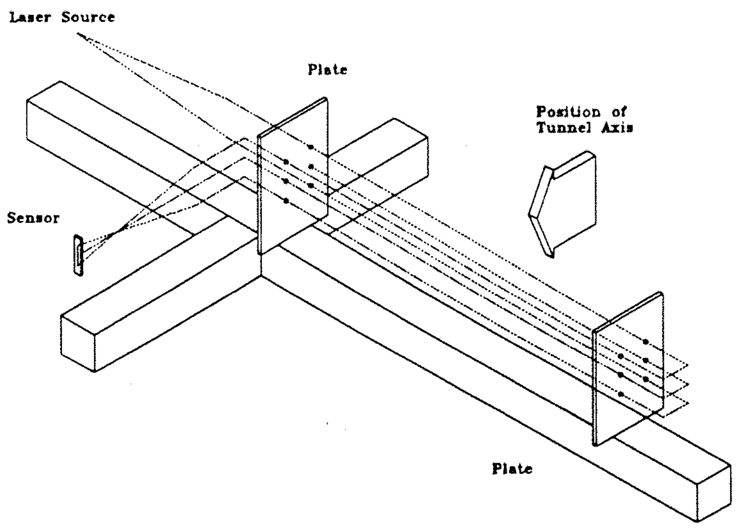


Figure 58: The alignment plates in position, showing hole locations and ray traces for one beam.

The unexpanded beams were used without lenses to allow alignment of the optical axis with the aluminium support structure. The beams were adjusted to pass through the centre hole of each set, producing, (when parallel and at the required separation and axial/ vertical alignment), a spot of light on the central segments of the sensors. The collimation and imaging lenses were then positioned to maintain the location of the central spot on the sensors. The expansion lenses were added and adjusted in conjunction with the collimation lenses to provide the required vertical skew and collimation.

12.7 Discussion

The practical observations in sections 12 and 12.6 were intended to establish the functionality of the linear array system in application to tracking flight history of a free-flight model. This entailed characterisation of the electronic system, sensors and optical system and demonstration of the acquisition of model motion data. The initial concern was to demonstrate the generation of the required beam structure and the practicality of beam alignment.

12.7.1 The Optical Design - Alignment

The purpose of beam alignment was to establish an orthogonal frame of sheet beams, geometrically precise within the region being used to track model motion. The separation between the calibration plates was much greater than the required depth of field so, even allowing for the limited precision in visual alignment in arranging laser spots to overlap holes in plates, the errors in beam placement inside the tracking region were less than the pixel-limited resolution of the sensors. For example, the calibration plate separation was 740 mm. It was estimated that the laser spots produced from the first plate could be aligned with the hole in the second plate to within 0.2 mm. The angular error produced results in a maximum displacement across the 60 mm deep tracking region of 0.016 mm, less than the 0.07 mm resolution set by pixel spacing.

The beam adjustment provided in the rig design was concluded to be satisfactory for the level of accuracy set by pixel resolution. For use with higher, sub-pixel resolution, a more precise spot alignment scheme could be devised using, for example, quadrant detectors. These devices consist of four photo-diodes, each forming a quadrant in a circular array. An incident light beam can be accurately centralised by observing the balance of the outputs of the four diodes.

12.7.2 The Electronic System

The electronics operated in general as expected, with acceptable dc drift and noise levels. Improvements on the noise levels on the adc cards may be possible by further experimentation with decoupling and filtering.

Channel gain and offset variations were not negligible but these could be reduced by careful component matching. An approach to reducing the problems of channel differences that would also accommodate laser aging, beam intensity profile changes and unpredictable channel-differential effects would entail the use of a monitoring scheme for dark and saturation levels before and after the run. The signal levels obtained from each segment would be used to set up channel scaling and offset factors which would then be applied to normalise the data. This idea could be extended to individual pixel levels to compensate for non-uniformity in illumination and response.

12.7.3 Sensor and Electronic System Deficiencies

Inspection of the direct video output from the sensor array revealed a settling period for the first two pixels following the shift-register loading period. These pixels would be interpreted as showing a reduced response to illumination, an effect which could be offset by the scaling factor scheme just mentioned.

A more significant defect was observed in the acquisition system in the form of a false signal level recorded for the first pixel in every segment. This problem is thought to be due to logical timing on the control interface card, and it is considered likely that the reprogramming of one of the PAL's would correct the problem. To establish the need for modification, the impact of a missing pixel needs to be considered.

The pixels in question can be easily identified within the stored data because of the use of indexed access. Considering the situation with edge data representing the unfocussed shadow edge, the gradual, multi-pixel transition allows interpolation to provide a good estimate for the missing pixel. The main impact occurs in the case of data for the focused shadow image, which can provide a light/ dark transition in a single pixel. Under these circumstances the loss of a pixel means that changes in shadow edge location in this region are not detected. The impact on the record of model flight history depends on the variations in coincidence of the four shadow edges available with the first pixels in the array segments. For normal flight trajectories, motion is progressive and edge transition through any of the possible blind spots would be brief and asynchronous. The missing pixels would in effect contribute to the overall noise in the perceived model position data.

12.7.4 Saturation Effects

One of the key decisions in this work was to operate the sensors under saturation exposure conditions to reduce problems with uniformity of sensor response and beam intensity. Beam intensity variations are a particular consequence of the use of laser illumination, selected to satisfy requirements in intensity and precision in beam geometry.

The sensors are protected to some degree against the effects of excessive light levels by an anti-blooming gate structure which should dispose of excess charge. However, the sensors were observed to have undesirable characteristics when used with excessive exposure, (see section 12.1.5). Under heavy saturation the pixel response to thin shadow images can be completely lost.

Following adjustments to the biasing of the sensor output gates, operation at a maximum of four times saturation exposure produced acceptable results. It was concluded that operation at about twice saturation exposure would allow a reasonable operational margin.

12.7.5 Beam Profile

The flatness of the beam intensity profile is a factor in determining the degree of over-saturation occurring across the sensor. The observations made on the beam profile, (figure 48) show deviations in intensity across the beam of approximately $\pm 55\%$. If the minimum observed in figure 48 was taken as the saturation exposure operating point, (by adjusting the exposure interval), the maximum intensity peaks can be expected to over-saturate the sensor by approximately 160 %. The effect on the sensor of intensity variations of this order can be seen by comparing figures 37(a) & 37(b) which show changes in perceived shadow width of about 1 pixel.

The weakest of the three sheet beams was used for sensor response observations and figures 52 & 54 indicate that the end segments of the sensor array are not completely in saturation at 3 MHz. The beam shape and dimensions would seem to be optimum (figure 50) but either more light, lower losses or a lower f_c are needed to achieve complete saturation across the array. Reducing the f_c to 1.5 MHz would result in 215 model position points for a 20 ms duration run. Improved surface coatings could produce a relative increase $_\wedge$ sensor irradience of approximately 30%. Higher illumination levels would increase the design margin and could be achieved economically by the addition of a second 8 mW laser (see appendix B). One further possibility would be the use of a short focal length cylindrical lens placed near the sensor to increase the beam intensity by beam width reduction. The beam width must exceed the sensor array width (18.4 μm) to allow for practical alignment and the implications of the effects of the cylindrical lens on image focusing would need to be considered.

12.7.6 Thresholding

The geometric position of the model in the shadow image for model locations away from the exact focused object plane is predicted by diffraction theory: the geometric edge position corresponds to a level representing 25% of the unobstructed beam intensity, I_0. However, I_0 at any given point cannot be accurately predicted because of the complex beam intensity profile and the use of saturation exposure.

When thresholding was used for edge estimation, it was found by experimentation that the optimum level for the shadow edge threshold (in lieu of knowledge of I_0) was close to the dark level, just clear of the ambient noise level . This position produced the least variation in perceived model width with moderate light level variations or changes of position within the field of view.

12.7.7 Summary

The hardware has been shown to function substantially as expected. The sensor, signal acquisition and data transfer system operate correctly at the full 6 MHz clock rate. The optical rig layout has been demonstrated as functional and a practical calibration scheme has been developed. Previous free-flight work has indicated typical model oscillatory motion with a period of 16 ms [5]. Operation of the tracking system with models in motion at twice this frequency has been demonstrated. Operating restrictions and limitations for the sensors and acquisition system have been identified and some solutions or suggestions for improvements have been made.

Chapter 13

Further Work

The key components of the optical tracking system described in this work have been successfully demonstrated.

The next stages in the practical application of the system to the gun tunnel at Southampton University are :

- Installation of the two final acquisition racks and power supplies.
- Development of data processing software to automate edge detection and data reduction.
- Modification of the bench top rig for application to the tunnel.

13.1 Mechanical Modifications

The mechanical modifications required to convert the bench top rig for use on the tunnel are restricted to changes to the support members (figure 20). The main cantilever needs to pass under the tunnel working section and so must be offset from the optical working surfaces. Three anti-vibration mounts isolate the assembly from the impulsive vibrations occurring during tunnel firing. Ballast weights fitted to the assembly assist in isolation and counter the weight of the support beam passing under the working section.

13.2 Optical Modifications

The observations made in chapter 12 indicate that the system would require enhancement to achieve operation at an f_c of 6 MHz. Estimates on optical surface losses (Appendix B) indicate that the most effective way to increase the beam intensity to the level required for sensor saturation at the higher clock rates is to use the existing laser

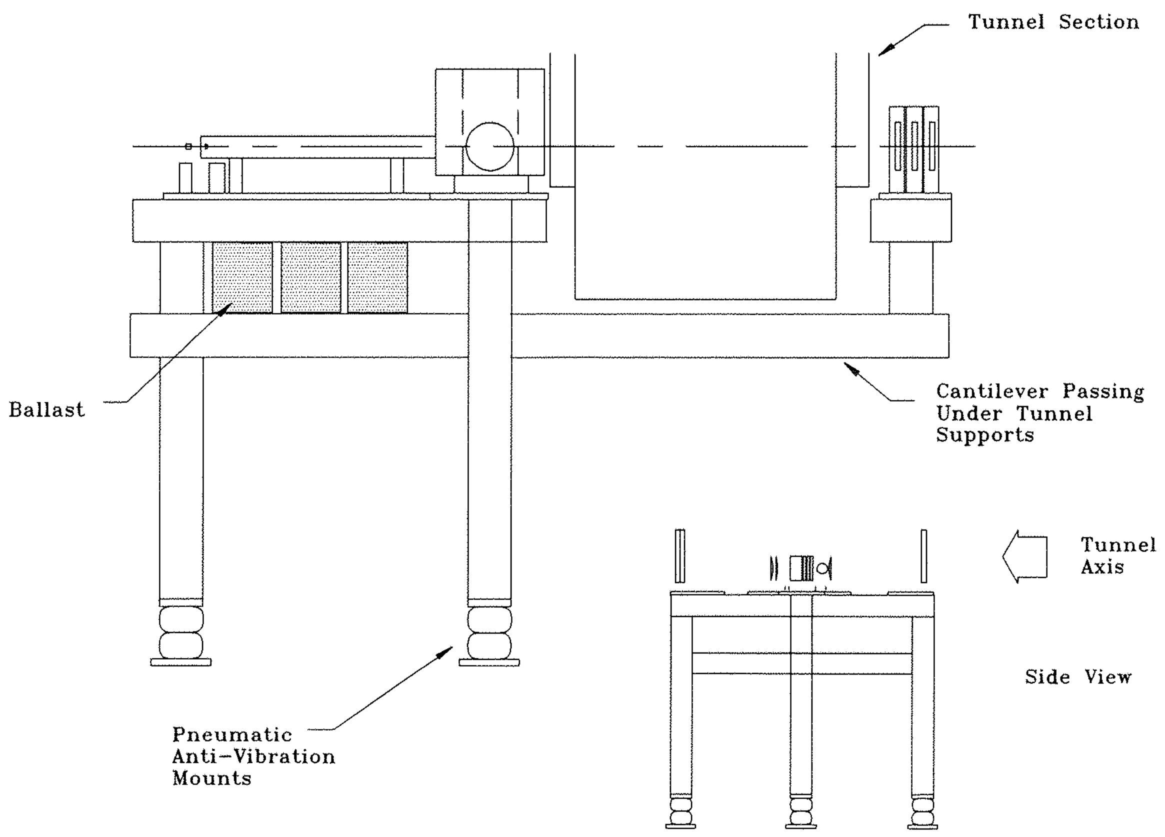

Tunnel Section
Ballast
Cantilever Passing Under Tunnel Supports
Tunnel Axis
Side View
Pneumatic Anti-Vibration Mounts

Figure 59: Adaption of the bench-top rig to the wind tunnel.

to provide two sheet beams and derive the third beam from an additional low power laser. A 8 mW Helium-Neon laser applied to the third beam would ensure similar illumination levels for the three sheet beams. Incorporation of anti-reflection coatings to all transmission surfaces in the beam path would increase operational margins and offset the gun tunnel working section window losses.

13.3 Semiconductor Lasers

Semiconductor laser sources may provide a compact alternative to gas lasers. High power lasers of several Watts are available for the long wavelengths, visible light devices being currently limited to a few mW. The present cost of these devices, taken with the cost of additional optics for correction of astigmatism and beam collimation tend to make the final price approach that of similarly sized gas lasers. Operation at infra-red wavelengths is possible but not recommended because of alignment difficulties and significant degradation of sensor performance due to increased photon penetration. This latter phenomenon causes degraded spatial frequency response and increased fixed pattern noise.

13.4 Data Storage

Each data set for a run consists of 3 megabytes. To avoid rapidly filling the hard disk drive, either some form of data-reduction or alternative bulk storage must be used. A tape streamer could be one solution. A short term alternative is to use a data archiving programme of the type freely available in the public domain. These programmes can achieve 90% data compression on typical run-time data.

Chapter 14

Conclusion

The aim of this work has been to establish the requirements of a particularly demanding tracking application, investigate the availability of systems, components and techniques, and to derive a design for a practical system that could economically achieve the high performance required.

An opto-electronic tracking system has been designed and built that will enable the free-flight history of a model to be recorded in digital form with high spatial and temporal resolution (sensor resolution of 0.07 mm, at an image sampling rate of 43 kHz for an f_c of 6 MHz). The tracking system has been optimised to match the characteristics of the tunnel, model and available sensor technology to provide a solution that satisfies the required specification.

A software operating system has been developed to facilitate data acquisition and qualitative analysis. Observations made on a functional bench top rig have established the operational principle and indicated the improvements needed to achieve full performance in application to the hypersonic gun tunnel working section. With minor modification, the system could be applied to considerably enhance the accuracy and scope of short duration, high Mach number aerodynamic investigation using free-flight techniques.

Bibliography

[1] Schueler C.J.,Ward L.K.,Hodapp Jr. A.E. Techniques for measurement of dynamic stability derivatives in ground test facilities. Agardograph 121, 1967.

[2] Tuttle M.H.,Gloss B. Support interference of wind tunnel models - a selective annotated bibliography. NASA TM-81909, 1981.

[3] Tuttle M.H., Kilgore R.A., Boyden R.P. Magnetic suspension systems - a selected, annotated bibliography. NASA TM-84661, 1983.

[4] Dayman B. Free-flight testing in high speed wind tunnels. Agardograph 113, 1966.

[5] East R.A., Hillier T.J. Free-flight oscillatory experiments in a hypersonic gun tunnel. In *The Proceedings of The Shock Tube and Waves Symposium*, 1985.

[6] Jackson R.M.B. Target tracking using area correlation. In *Int.Conf on Advanced Infra-red Detectors and Systems.* IEE Oct 1981.

[7] Woolfson M.S. An evaluation of manoeuvre detection algorithms. *GEC Journal of Research*, Vol 3:181–190, 1985.

[8] Dubovik A. *The Photographic Recording of High Speed Processes.* John Wiley & Sons, 1981. Edited by Lunn G.H.

[9] Brown R.R., Parker J.R. Conceptual design and analysis of high-speed electronic imaging. Technical Report LA-10072-MS, Los Alamos National Lab, November 1984.

[10] Holt D., Parker J., Morel R., Winchenbach J., Kelley R.J. A prototype high speed electronic imaging system for aeroballistics research and analysis. In *Conf Proc.ICIASF*, 1987.

[11] Videk megaplus camera data sheet. Videk, Kodak, 1987.

[12] Schofield J.A. Mosley J.D. CCD Imaging Arrays. *EDN Magazine*, Vol 35:116–124, 1990.

[13] Teh-Huang Lee, Tredwell T., Burkley B.C., Anagnostopoulos C., Hayward J.S., Kelly T.M., Khosla R.P., Losee D.L., Lavine J.P. A solid state image transfer sensor for image recording at 2000 frames per second. *IEEE Trans*, 1982.

[14] Bixby J.A. High speed television camera and video tape recording, system for motion analysis. *SPIE Vol 301 Design of Digital Image Processing System*, 1981.

[15] Sequin C.H, Tompsett M.F. *Charge Transfer Devices*, volume Supl. No 8, Advances in Electronics and Electron Physics. Academic Press, 1975.

[16] Herman M., Wölber J. The p^2ccd 500b linear imager. *Philips Electronic Components and Applications*, Vol 1(No 3):183, 1979.

[17] Papanicolaou N.A. Whitlock R.R. Streak and framing camera designs using surface acoustic wave charge-transfer devices. *Optical Engineering*, Vol 25(No 12):1267–1277, December 1986.

[18] Aikens R.S., Epperson P.M., Bonner Denton M. Techniques for operating charge coupled devices in very high speed modes. Technical Report AD-A152 878, Arizona University, Tuscon, Department of Chemistry, 1985.

[19] Klocek P., Roth M., Rock R.D. Chalcogenide glass optical fibres and image bundles: Properties and applications. *Opt.Eng.*, Vol 26(No 2), February 1987.

[20] Collet M. Mos-xy, interline and frame transfer sensors compared. *Electronic Components and Applications*, Vol 7(No 2), 1985.

[21] Random acces charge injection device (racid). Design Aid Module,General Electric Company, Syracruse, USA, NY 13221, 1982.

[22] Burke H.K, Michron G.J., Tomlinson H.W., Vogelsong T.L., Grafinger A., Wilson R. Design, fabrication and delivery of charge injection device for stellar tracking.

Technical report, NASA NASB-32801, General Electric Company, 1979. Contract Report.

[23] Grafinger A.B. Review of charge injection device (cid) technology. Operation Notes from General Electric Opto-electronic Systems, 1983.

[24] White C.M., Green W.T. Recent innovations in cid cameras. *State of the Art Imaging Arrays and Their Applications*, SPIE Vol 501, 1984.

[25] Is32 optic ram spectral data applications note. Micron Technology Inc, 2005 East Columbia Road, Biose, Idaho 83706, 1985.

[26] Is256 optic ram data sheet. Micron Technology Inc.,USA, 1986.

[27] Sewell F.A. The light sensitive mnos memory transistor. *IEEE Trans. on Electron Devices*, Vol 20(NO 6):pp563–572, 1973.

[28] Ciarcia S. Micro d-cam solid state camera. Byte, Sept-Oct, 1983.

[29] Seymour H.R. Electro-optic space positioner. U.S. Patent No. 4,136,568.

[30] Eloptopos information bulletin. Eloptricon, Billingstadletta 99, N-1360 Nesbru, Norway, 1986.

[31] Data sheet on multipoint x-y tracker htv-c799. Hakuto Ltd, 1979.

[32] C1181 random access camera product bulletin no 8001. Hamamatsu, 1979.

[33] Koenig E.W. Stereo electro-optical tracking system (SETS). NASA CR-172471, September 1984.

[34] Feitelson D.G. *Optical Computing : A Survey for Computer Scientists.* MIT Press, 1988.

[35] Gara A.D. Real time tracking of moving objects by optical correlation. *Applied Optics*, Vol 18(No 2), 1979.

[36] Tien-Hsin Chao, Hua-Kuang Lui. Real time optical holographic tracking of multiple objects. *Applied Optics*, Vol 28(No 2), 1989.

[37] Data sheet for smd128. Semetex Corporation, 3450 Fujita Street, Torrance, California,USA, 1986.

[38] Sandel B.R., Collins D.F., Lyle Breadfoot A. The effect of phosphor persistence on photometry with image intensifiers an integrating readout devices. *Applied Optics*, Vol 58(No 20):3697–3704, 1986.

[39] Horn J.E., Cutche M. Decay time of some image tube phosphors as a function of excitation time. *Proc. IEEE*, Vol 58:592, 1970.

[40] Timothy J.B. Multi-mode Microchannel Array Detector Systems - Performance Characteristics. *Optical Engineering*, Vol 24(No 6):1066–1071, 1985.

[41] Yir Talmi, Simpson R.W. Self-scanned Photodiode Array: A Multichannel Spectrometric Detector. *Applied Optics*, Vol 19, May 1980.

[42] D-series tapped array, rl1288d data sheet. EG & G Reticon, 34/35 Market Place, Wokingham, Berkshire, RG11 2PP, England, 1988.

[43] Kreuzer J. Laser Light Redistribution in Illuminating Optical Signal Processing Systems. *Optical & Electro-optical Information Processing*, 1965. MIT Press.

[44] Seitz P. Optical superresolution using solid-state cameras and digital signal processing. *Optical Engineering*, Vol 27(No 7), 1988.

[45] Parker D.H. *Techniques for Extreme Attitude Suspension of a Wind Tunnel Model in a Magnetic Suspension and Balance System.* PhD thesis, University of Southampton, Department of Aeronautics & Astronautics, 1989.

[46] Ping Tcheng, Schott T.D. A five component electro-optical positioning system. Proc. ICASF, 1987.

[47] R.J.Halford. A prototype of a high performance passive tracking system for wind tunnel free-flight experiments. Technical Report, Dept. of Aeronautics and Astronautics, University of Southampton, 1991.

[48] Programmable logic handbook, 4th edition. Monolithic Memories Ltd, Farnborough, Hants, England, 1985.

Appendix A

Dimensions of the Optical System

The optical system dimensions are indicated in figure 60. The basic dimensions common to the three beams are:

(A) the inter-beam separation (30.6 mm),

(C) the imaging lens to sensor spacing (375 mm)

(F) the return beam separation (45.9 mm), and

(G) the beam expander lens pair separation (510.4 mm).

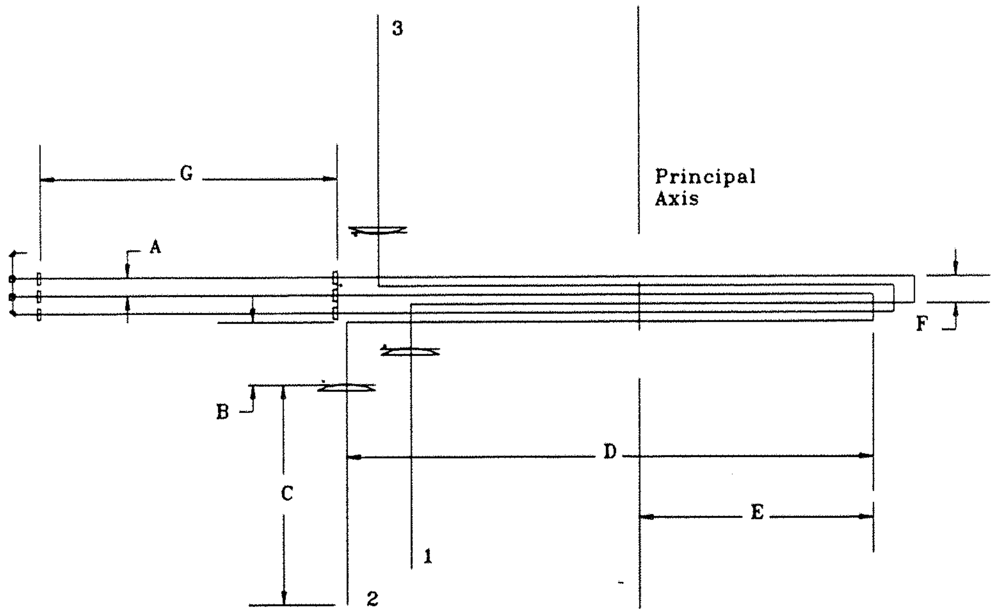


Figure 60: The dimensions for key optical components.

The expansion lens is plano-cylindrical, of focal length 10 mm, the collimation is plano-convex, of focal length 500 mm, and the imaging lens is plano-convex with a focal length of 300 mm. Object magnification is 4.

Referring to figure 60, the respective distances (in mm) for the three beams for the bench top rig were as follows:

	Beam 1	Beam 2	Beam 3
B	77.37	108.72	92.97
D	884.18	923.43	903.88
E	492.55	421.95	457.25

In application to the wind tunnel, the dimensional adjustments required to offset the effects of the thick glass observation windows need to be considered. The refractive effects of the windows produces an apparent change in object distance position given by:

$u_r = t(1 - 1/\mu)$ where

t = the glass path length (4 x 25.4 mm)

μ= is the air/ glass refractive index (1.5)

These values result in a shift in apparent object position along the optical axis of 33.86 mm. The easiest adjustment to the system to maintain focus is to increase the return mirror-to-collimation lens separation by 16.9 mm.

Appendix B

Rig Transmission Losses

The various optical surfaces present in the bench top system are indicated in figure 61. The letter 'P' indicates a prism, 'Sp' a beam splitter, 'L' an lens, 'W' for wind tunnel

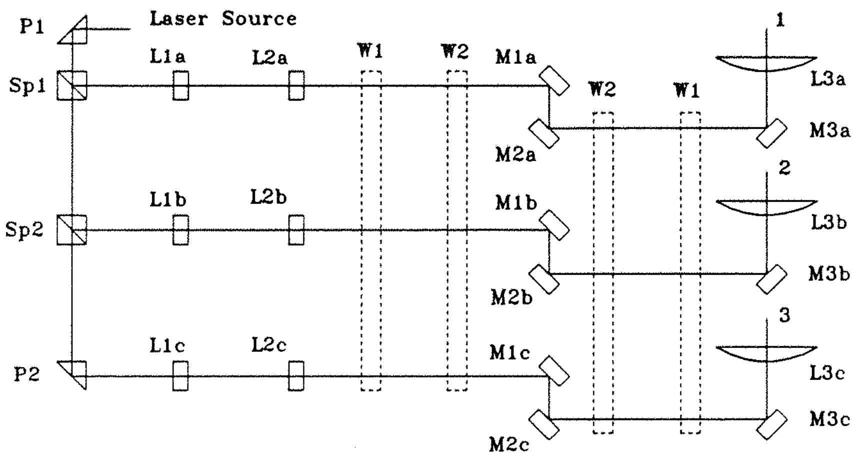


Figure 61: The optical surface losses.

window and 'M' for mirror. The estimated percentage transmission (or reflection) for each device used for the bench top tests was as follows:

Device		Accumulative Transmission			
		Beam 1	Beam 2	Beam 3	Comments
P1	92%				uncoated
Sp1	49/49%	45.1%			coated
Sp2	49/49%		22.1%		coated
P2	92%			20.3%	uncoated
L1	92%	41.5%	20.3%	18.7%	uncoated
L2	92%	38.2%	18.7%	17.2%	uncoated

M1	95%	36.3%	17.8%	16.3%	Aluminium surface
M2	95%	34.6%	16.9%	15.5%	Aluminium surface
M3	95%	32.7%	16%	14.7%	Aluminium surface
L3	92%	30.1%	14.7%	13.6%	uncoated
Beam distribution:					
	30.1%	14.7%	13.6%		

The shape of the beam arriving at the sensors was approximately elliptical with minor and major diameters estimated from observations (see figure 49) at 0.25 mm and 19 mm, resulting in an area of 0.037 cm^{-2}. If the available laser beam power is taken as 80% of the specified power, the irradience at the each sensor can be estimated at 104, 50 and 47 mW cm^{-2}. The saturation exposure has been estimated at 31 mW cm^{-2} (22.3 μs exposure interval at $f_c = 6$ MHz; see section 10.3.1) for a broad band light source. An improved response of some 12% can be expected from the near monochromatic laser light but the observations made in section 12.5 indicate that the array is close to saturation only for the central segments. This discrepancy is probably not surprising on consideration of the approximations necessary in deriving estimates.

The observations indicate that the power available from the laser is inadequate for system operation at the full f_c of 6 MHz. The addition of glass windows to the light path will present further transmission losses when the system is installed on the wind tunnel (estimated at 50% for a λ of 632.8 nm).

The methods available to ensure saturation of the sensors at the full sampling rate of 6 MHZ are discussed in section 12.7.5. Revised estimates for the cases of improved optical coatings (magnesium fluoride) indicate an increase in percentage of source light incident on the sensor for beams 1,2 and 3 to 37.2, 18.2 and 17.6 % respectively.

The replacement of one of the prisms with an additional laser would result in the following improvements in transmission:

Device		Accumulative Transmission			
		Beam 1	Beam 2	Beam 3	Comments
P1	92%				uncoated
Sp1	49/49%	45.1%	45.1%		coated
P2	92%		41.5%		uncoated
P3	92%			92%	uncoated
P4	92%			84.6%	uncoated
L1	92%	41.5%	38.2%	77.8%	uncoated
L2	92%	38.2%	35.1%	71.6%	uncoated
M1	95%	36.3%	33.3%	68.1%	Aluminium surface
M2	95%	34.6%	31.6%	64.7%	Aluminium surface
M3	95%	32.7%	30.1%	61.4%	Aluminium surface
L3	92%	30.1%	27.6%	56.5%	uncoated
Beam Distribution:					
	30.1%	27.6%	-	(16 mW laser)	
	-	-	56.5%	(8 mW laser)	

The beam intensity at the sensors would be well matched if the secondary laser had a similar intensity profile and about 50% of the power output. Lasers in this class are comparatively cheap.

Appendix C

The Electronic Circuitry

The electronic circuitry for the system is outlined in section 10.3 and described in detail in reference [47]. Figure 62 represents the acquisition electronics and shows the partitioning of the various sub-functions for the system into abridged schematics labelled as sections A to H (figures 63 to 71). The position of the various blocks within the sections shown in figure 62 represents the approximate location of the stated functions within the respective schematic. The interface card consists of sections A to D, the rack distribution card consists of section G, the rack buffer cards, section H and the video acquisition and store cards, sections E to F.

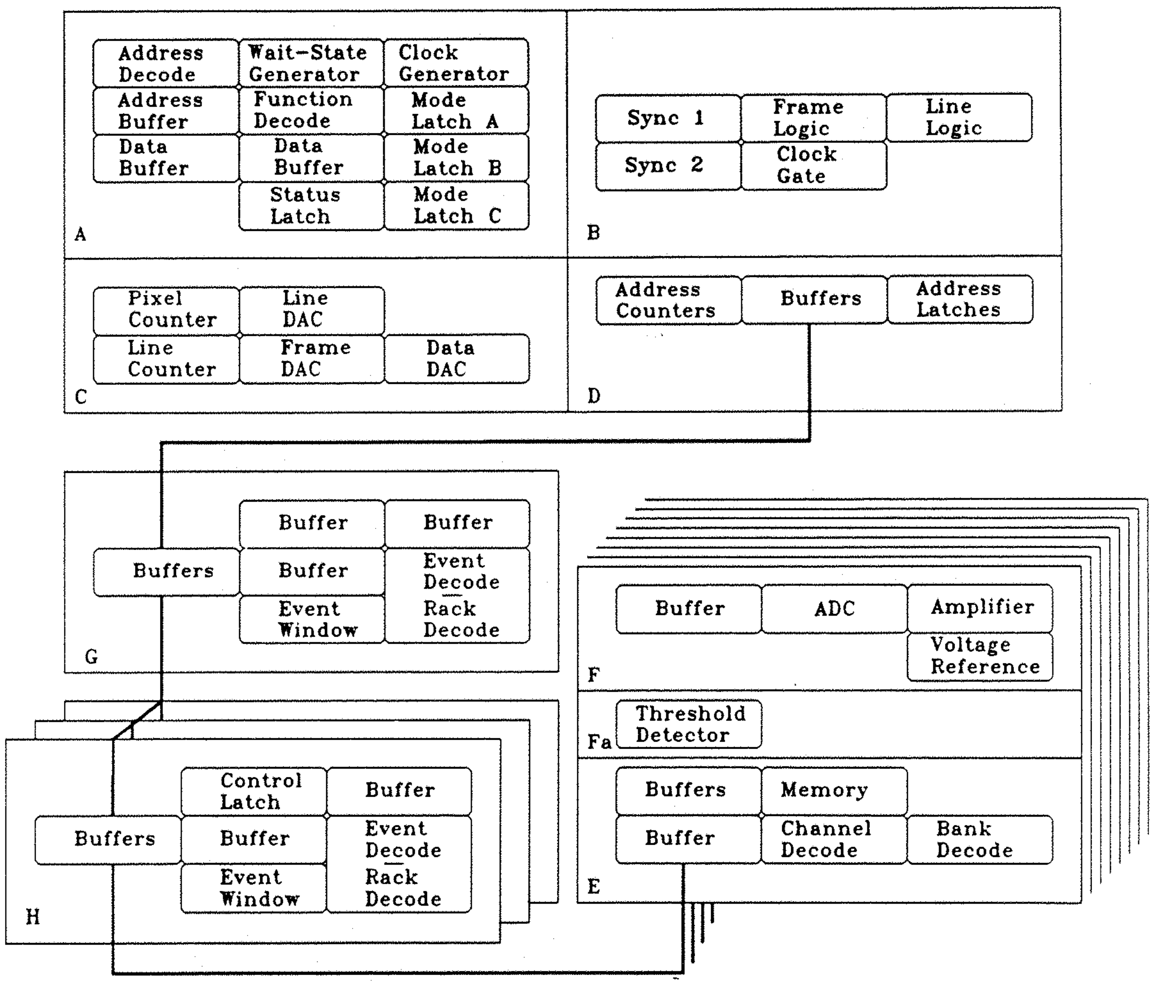


Figure 62: The PC and rack interfaces - block diagram.

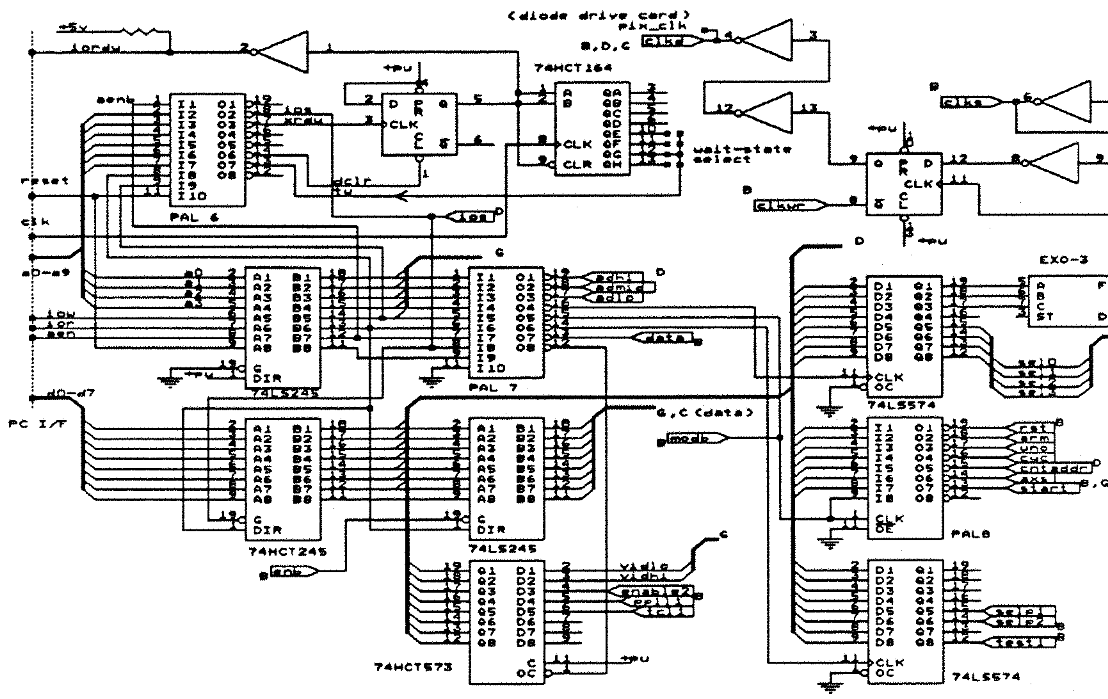


Figure 63: Circuit detail - pc interface card, section A.

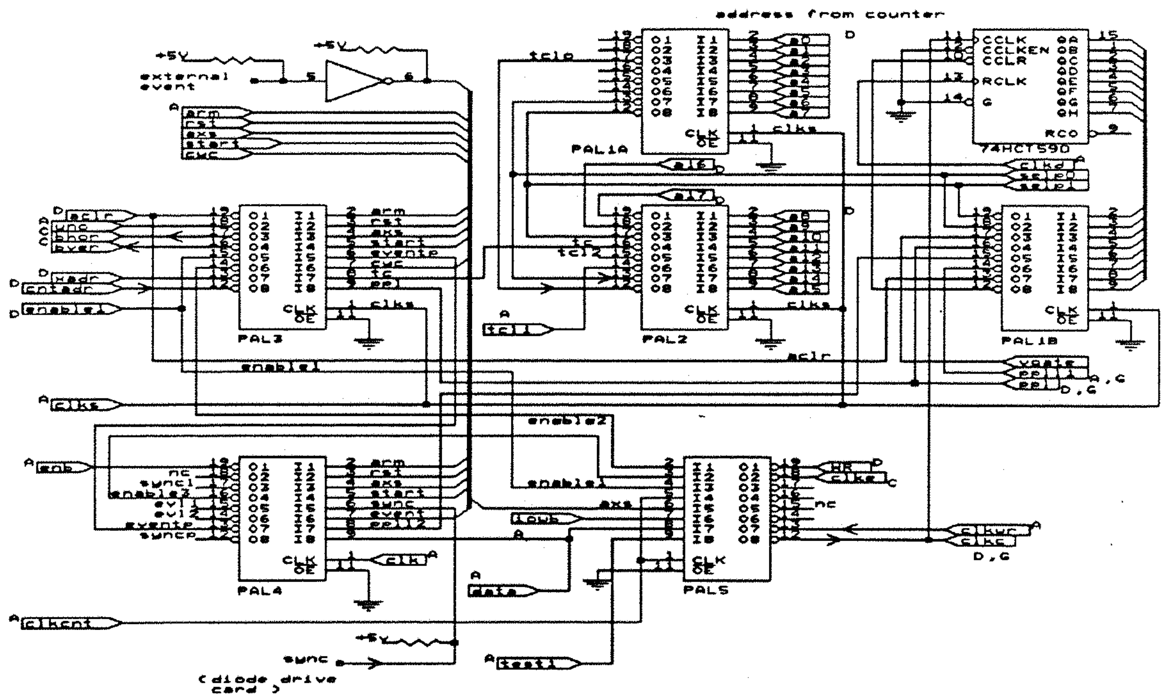


Figure 64: Circuit detail - pc interface card, section B.

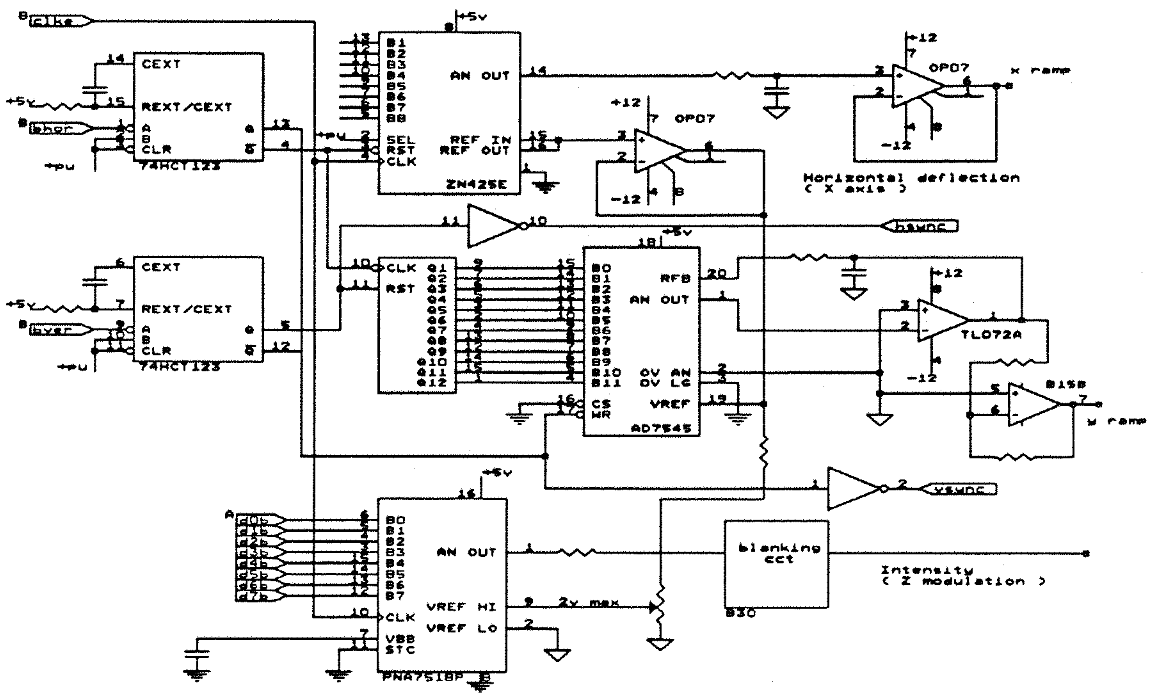


Figure 65: Circuit detail - pc interface card, section C.

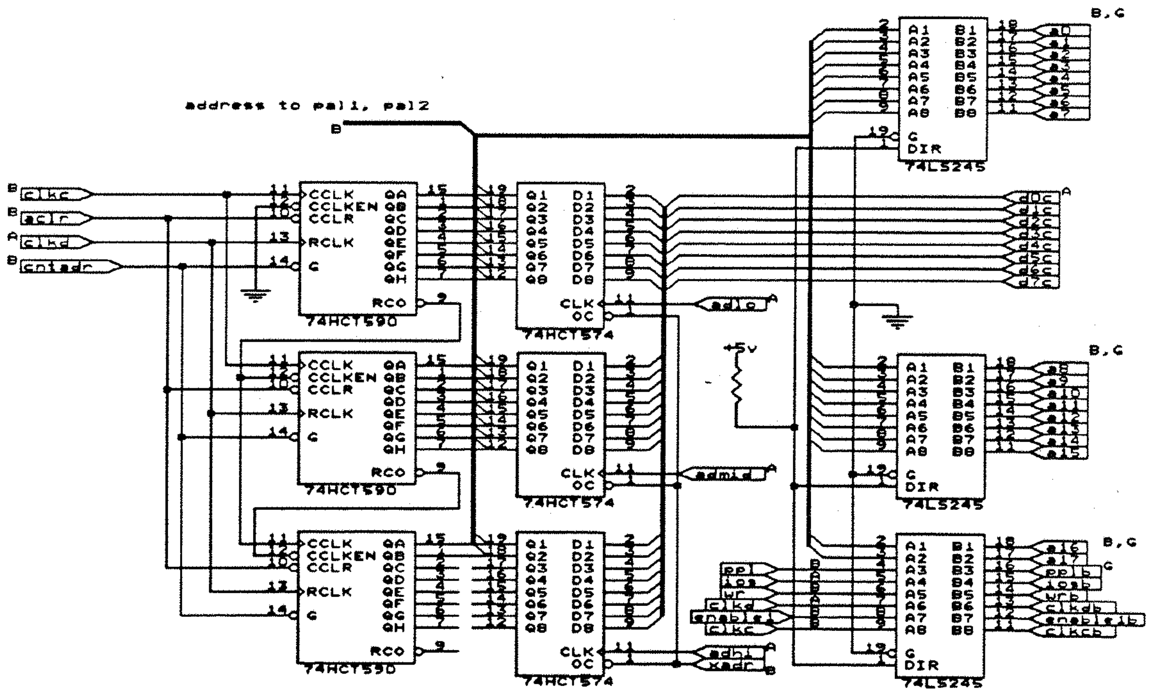


Figure 66: Circuit detail - pc interface card, section D.

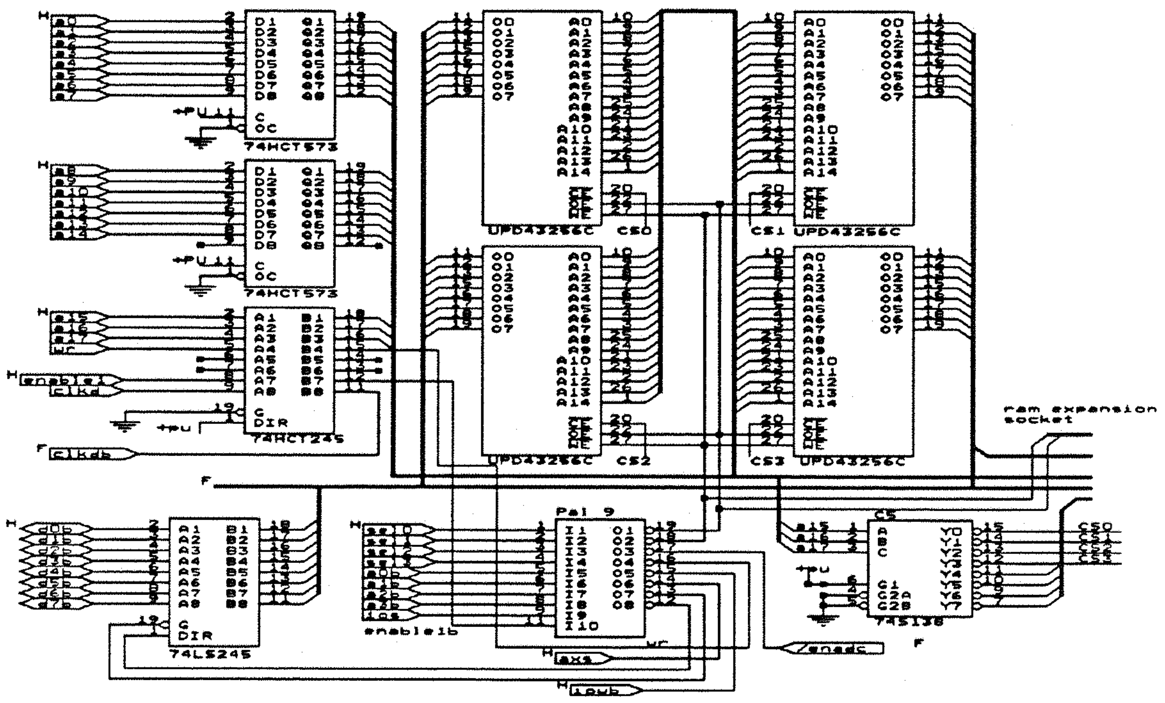


Figure 67: Circuit detail - video acquisition card, section E.

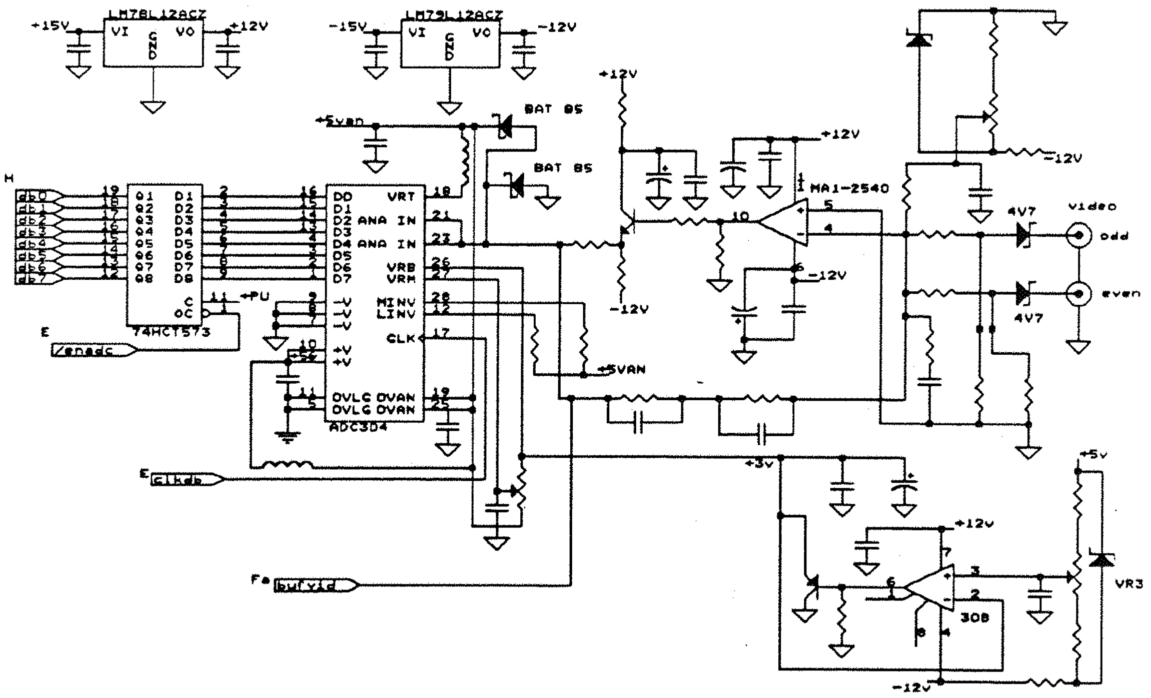
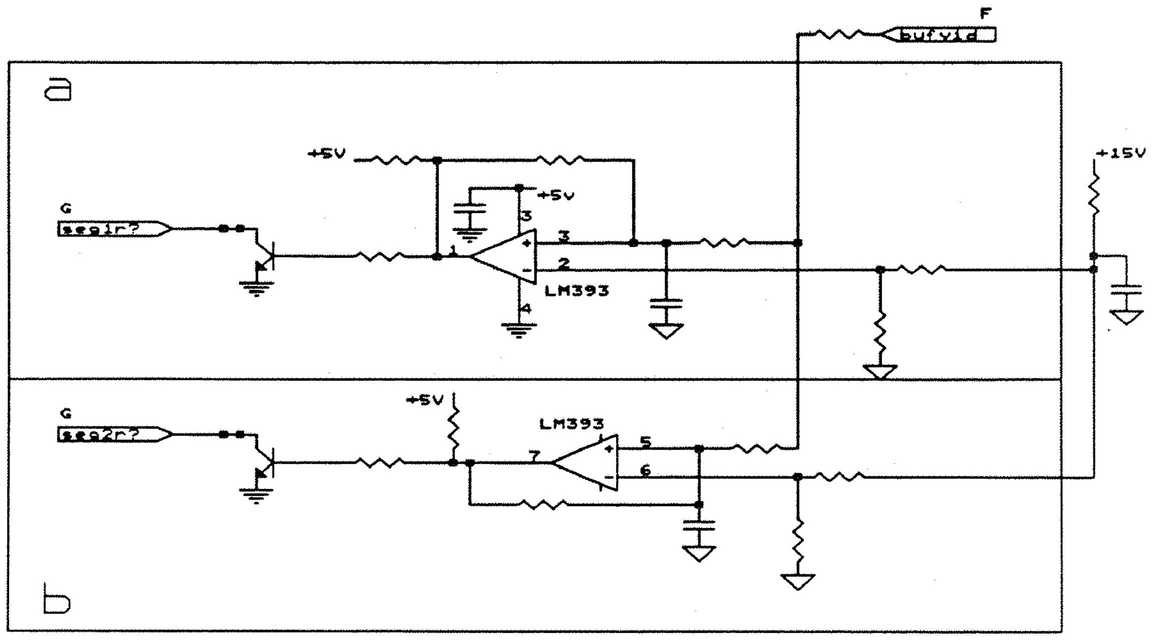


Figure 68: Circuit detail - video acquisition card, section F.



note: components shown in 'a' needed for segment 1 only.
components shown ib 'b' needed for segment 2 only.

Figure 69: Circuit detail - video acquisition card, section Fa.

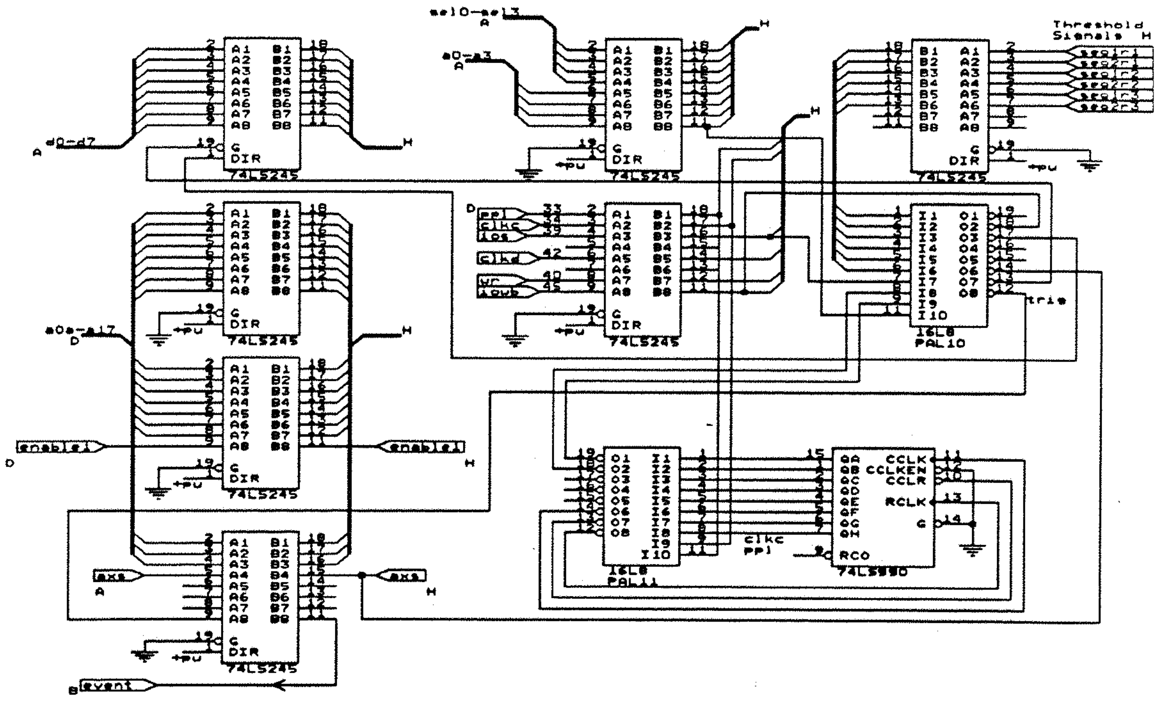


Figure 70: Circuit detail - distribution buffer, section G.

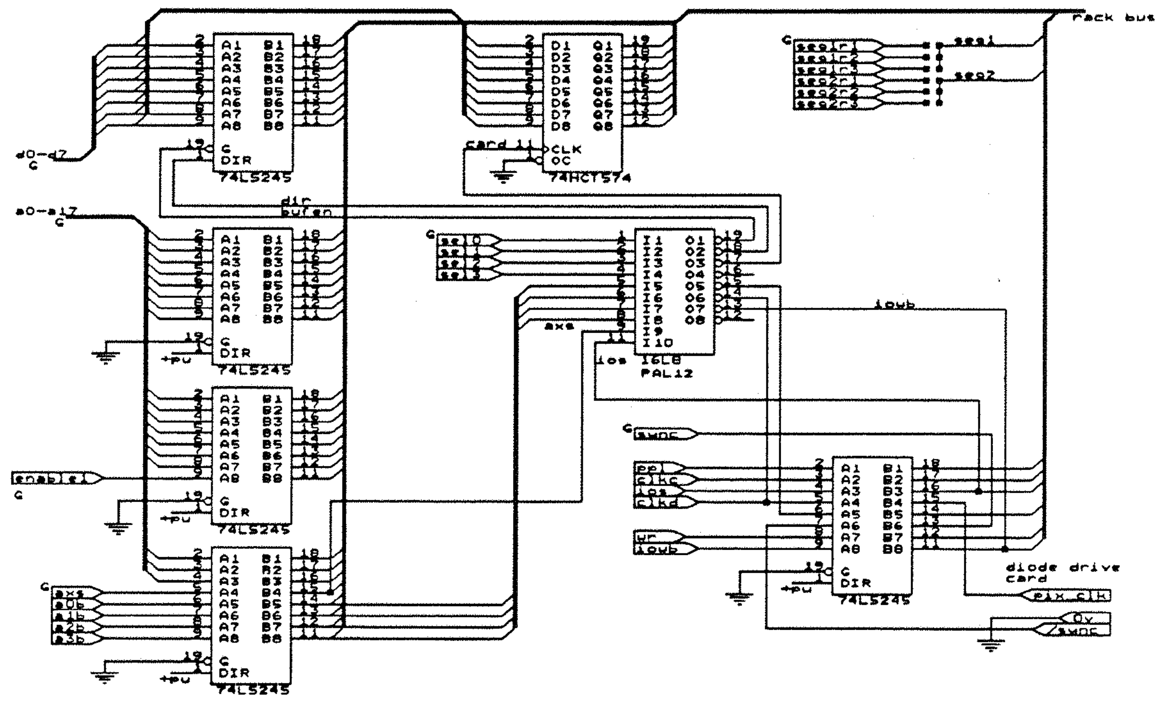


Figure 71: Circuit detail - rack buffer, section H.

Appendix D

Programmable Array Logic

The design of the interface and rack control cards incorporates several programmable array devices (PALs) to reduce the overall chip count, in some cases by up to 4 or 5 ic's per PAL. One distinct advantages of PAL's in experimental design is that major logic changes are facilitated on-chip, without the problems of re-routing external circuitry. Twelve different PAL programmes have been used in this design. The source code is written in Monolithic Memories PALASM2 dialect[48].

```
***********************************
CHIP            HSTPAL1B    PAL16R4
***********************************

PIN MNEMONICS :
INPUTS:
CLK - 12 MHz INDEPENDENT TEST I/F SYSTEM CLOCK
A0-A7 I/F ADDRESS BUS.
EN - ENABLE
D1K  - 1K PIXEL BOUNDARY DETECTED
SELP1 -SELECT TERMINAL PIXEL PER LINE COUNT
SELP2   00- 130, 01- 1K, 10 -256 11- PPL NO EFFECT

OUTPUTS:
PPL - PULSE HIGH, PIXEL PER LINE COUNT DETECTED.
TCLO - LO LEVEL,LOW PART OF TERMINAL PIXEL COUNT
        DETECTED.
PPLL1,PPLL2 INTERMEDIATE LOGIC

2 LATCHES CLOCKED BY CLKS NOT USED

PIN DEFINITION:
CLK A0 A1 A2 A3 A4 A5 A6 A7 GND
EN SELP2 CLKD PPLL1 PPLL2 PPL TCLO D1K SELP1 VCC

EQUATIONS
(PIXEL TERMINAL COUNT-LO BYTE)
INTEGER OF TOTAL ((MEM/130)*130)-1=131039
```

```
1FFEFH OR 0001 1111 1111 1110 1111B
TCLO DETECTS LOWER BYTE OF TERMINAL COUNT

/TCLO:=A7*A6*A5*/A4*A3*A2*A1*A0*CLKD

***********************************
CHIP            HSTPAL1B    PAL16R4
***********************************

PIN MNEMONICS :

INPUTS:
CLK - 12 MHz INDEPENDENT TEST I/F SYSTEM CLOCK
A0-A7 LINE MONITOR ADDRESS COUNTER.
EN - ENABLE
SELP1,SELP0 -SELECT TERMINAL PIXEL PER LINE
 0       0     128
 0       1     129
 1       0     130
 1       1     131


OUTPUTS:
PPL - PULSE HIGH, PIXEL PER LINE COUNT DETECTED.
VGATE - VIDEO EVENT WINDOW GATE
PPLL1,PPLL2 INTERMEDIATE LOGIC

2 LATCHES CLOCKED BY CLKS NOT USED

PIN DEFINITION:

CLK A0 A1 A2 A3 A4 A5 A6 A7 GND
EN CLR ACLR PPLL1 PPLL2 PPL VGATE SELP0 SELP1 VCC

EQUATIONS

PIXELS PER LINE-
TRIAL COUNT VALUE---SUSPECT THIS WILL REQUIRE TRIMMING @@
PIXELS, COUNT= (128 PIXELS +BLACK LEVEL SAMPLES)
128  =0111 1111B
129  =1000 0000B
130  =1000 0001B
131  =1000 0010B

/PPLL1:=/SELP0*/SELP1*A0*A1*A2*A3*A4*A5*A6*/A7+
        SELP0*/SELP1*/A0*/A1*/A2*/A3*/A4*/A5*/A6*A7+
       /SELP0*SELP1*A0*/A1*/A2*/A3*/A4*/A5*/A6*A7+
        SELP0*SELP1*/A0*A1*/A2*/A3*/A4*/A5*/A6*A7
```

```
/PPLL2:=/PPLL1

/PPL:=/SELP0*/SELP1*A0*A1*A2*A3*A4*A5*A6*/A7*PPLL2+
      SELP0*/SELP1*/A0*/A1*/A2*/A3*/A4*/A5*/A6*A7*PPLL2+
     /SELP0*SELP1*A0*/A1*/A2*/A3*/A4*/A5*/A6*A7*PPLL2+
      SELP0*SELP1*/A0*A1*/A2*/A3*/A4*/A5*/A6*A7*PPLL2

/CLR=/PPL+/ACLR

VIDEO EVENT WINDOW DEPENDS ON SEL0,1
SET TO 5/4 PIXELS CLEAR OF VIDEO START/ STOP
SEL1 SEL0     VGATE HI FOR COUNT:
 0    0       4-124
 0    1       4-125
 1    0       4-126
 1    0       4-127
ACTUAL ADDRESS=COUNT MINUS 1 DUE TO CIRCUIT RESPONSE

EXPRESSION ORDER
0-4,123-127,124-127,125-127,126-127,128-255

/VGATE:=/A2*/A3*/A4*/A5*/A6*/A7+
       A0*A1*/A2*A3*A4*A5*A6*/A7+A2*A3*A4*A5*A6*/A7+
       A2*A3*A4*A5*A6*/A7+
       A0*/A1*A2*A3*A4*A5*A6*/A7+A1*A2*A3*A4*A5*A6*/A7+
       A1*A2*A3*A4*A5*A6*/A7+
       A7

**********************************
CHIP            HSTPAL2    PAL16R4
**********************************

INPUTS:
CLKS - 12 MHz INDEPENDENT TEST I/F SYSTEM CLOCK.
A8-A17 I/F ADDRESS BUS.
EN - ENABLE, PPL - PIXEL PER LINE COUNT DETECTED.
TCL0 - TERMINAL COUNT ON A0-7 FROM HSTPAL1


OUTPUTS:
TC - TERMINAL PIXEL COUNT DETECTED.
D1K -DETECT 1K BOUNDARY
2 LATCHES CLOCKED BY CLKS NOT USED
TCL1,TCL2 INTERMEDIATE LOGIC
PIN DEFINITION:

CLK A8 A9 A10 A11 A12 A13 A14 A15 GND
```

```
EN TCLO CLKD TCL1 TCL2 TC D1K A16 A17 VCC

EQUATIONS

PIXEL TERMINAL COUNT-
INTEGER OF TOTAL ((MEM/130)*130)-1=131039
1FFEFH OR 0001 1111 1111 1110 1111B
TCLO DETECTS LOWER BYTE OF TERMINAL COUNT

/TCL1:=/TCLO*/A17*A16*A15*A14*A13*A12*A11*A10*A9*A8*CLKD

/TCL2:=/TCL1

/TC:=/TCLO*/A17*A16*A15*A14*A13*A12*A11*A10*A9*A8*TCL2*CLKD

DETECT 1K BOUNDARY FOR HSTPAL1

/D1K:=/A8*/A9*A10*CLKD

*******************************
CHIP            HSTPAL3 PAL16R4
*******************************

PIN MNEMONICS :

INPUTS:
CLK- 12 MHz TEST I/F CLOCK
ARM - PULSE LO,READY FOR EVENT TRIGGER
RST - PULSE LO,CLEAR TO DEFAULT STATUS
AXS - LEVEL,HIGH FOR ACCESS, LOW FOR ACQUIRE
START-PULSE LO,FORCE ENABLE1,WITHOUT EVENT TRIGGER
EVENTP-PULSE,ENABLE1 IF ARMED
CYC  -HIGH FOR CONTINUOUS CYCLING,LOW FOR NORMAL STOP ON TC
TC   -TERMINAL PIXEL COUNT FROM HSTPAL3
PPL  -PIXEL PER LINE COUNT
UNO  -LOW FOR NORMAL OPERATION, HIGH FOR TERMINATE ON
      SINGLE LINE
CNTADR-SELECT COUNTER FOR ADDRESS, DESELECT LATCHES
    UNO   CYC
     0     0      NORMAL, STOP ON TC
     0     1      CONTINUOUS, IGNORE TC
     1     0      UNO. STOP AFTER ONE LINE, ARM STARTS NEXT
     1     1      RESET COUNTER

OUTPUTS:
CLR  -CLEAR ADDRESS COUNTERS
ENABLE2 -LOW TO ENABLE ADC O/P ONTO MEMORY DATA-BUS AT
         START ACQUIRE
```

```
        -HIGH, DISABLE AT ACQUIRE CYCLE END.
XADR    -LOW TO SELECT ADDRESS LATCHES
BVER    -D-A VERTICAL SCAN CLOCK
BHOR    -D-A HORIZONTAL SCAN CLOCK

PIN DEFINITION:

CLK ARM RST AXS START EVENTP CYC TC PPL GND
EN CNTADR XADR ENABLE2 ENABLE1 BVER BHOR UNO CLR VCC

EQUATIONS

/ENABLE1:=UNO*/PPL+/CYC*/TC+ARM*/ENABLE1+/RST+AXS

/ENABLE2:=/AXS*/START*ENABLE1+/AXS*/EVENTP*RST*ENABLE1+
          /AXS*ARM*/ENABLE2*RST

/XADR=CNTADR

/CLR=/CYC*/TC+/RST+UNO*CYC

/BHOR:=/AXS*/PPL+AXS*UNO

/BVER:=/AXS*/TC+AXS*CYC

*******************************
CHIP            HSTPAL4 PAL16R4
*******************************

PIN MNEMONICS :

INPUT:
CLK 12 MHz I/F CLOCK
SYNC  -PULSE LOW,BEGINNING OF SCAN SYNC FROM DIODE DRIVE
RST   -PULSE LOW, RESET TO DEFAULT
START -FORCE ENABLE1,SET Q8
PPL   -PIXEL PER LINE COUNT, SET Q8
ARM   -ARM ENABLE1, SET Q8
AXS   -LOW, ENABLE ACQUIRE
DATA  -LATCH I/O EVENT
EVENT -HIGH, EVENT TRIGGER TO START ACQUIRE
ENABLE2 -ARMED AN READY TO GO

OUTPUT:
SYNCP    -PULSE ON RISING EDGE OF SYNC
ENABLE3  -ENABLE2, WITH ENABLE1 AT HSTPAL5, ENABLES
         CLOCK ETC
ENB      -ENABLE PC ACCESS BUFFER
```

```
EVENTP   -PULSE DERIVED FROM EVENT I/P
PIN DEFINITION:

CLK ARM RST AXS START SYNC EVENT PPL DATA GND
EN SYNCP EVENTP EVL1 EVL2 ENABLE3 /SYNCL ENABLE2 ENB VCC

EQUATIONS

/EVL1:=/EVENT

/EVL2:=/EVL1

/EVENTP=EVENT*/EVL2

SYNCL:=SYNC

/SYNCP=/SYNCL*SYNC

/ENABLE3:=/ENABLE3*RST*SYNCP+/ARM+/PPL+/START

/ENB=AXS*/DATA

*******************************
CHIP            HSTPAL5 PAL16R4
*******************************

PIN MNEMONICS :

INPUT:
CLK  -CLKD, S/W SELECTABLE MASTER CLOCK -DEFAULT 6 MHz
CLKS -12 MHz I/F SYSTEM CLOCK
AXS  -LOW, ACQUIRE MODE
IOWB -BUFFERED PC /IOWB
DATA -LATCH I/O DATA TO PC I/F
TEST1 -LO,NORMAL HI,IGNORE ENABLE3

OUTPUT:
AD   -ENABLE A/D DURING ACQUIRE CYCLE.
WR   -PULSE LO TO WRITE TO MEMORY DURING ACQUIRE CYCLE
CLKC -CLOCK COUNTER
CLKR -TRANSFER COUNT TO COUNTER O/P
EN3L1,EN3L2 INTERMEDIATE LOGIC

PIN DEFINITION:

CLK ENABLE2 ENABLE3 ENABLE1 CLKCNT AXS IOWB DATA TEST1 GND
EN CLKC CLKWR NC NC EN3L1 EN3L2 CLKE WR VCC
EQUATIONS
```

```
/EN3L1:=/ENABLE3

/EN3L2:=/EN3L1

/WR=ENABLE1*/ENABLE2*EN3L2*ENABLE3*CLKWR*/AXS*/TEST1+
   ENABLE1*/ENABLE2*CLKWR*/AXS*TEST1+
    AXS*/IOWB*/DATA

/CLKC=ENABLE1*/ENABLE2*EN3L2*ENABLE3*CLKCNT*/AXS*/TEST1+
     ENABLE1*/ENABLE2*CLKCNT*/AXS*TEST1+
      AXS*/DATA+
      ENABLE2*/ENABLE1*CLKCNT*/AXS

/CLKE=ENABLE1*/ENABLE2*EN3L2*ENABLE3*CLKCNT*/AXS*/TEST1+
     ENABLE1*/ENABLE2*CLKCNT*/AXS*TEST1+
      AXS*/DATA+
      ENABLE2*/ENABLE1*CLKCNT*/AXS

*******************************
CHIP            HSTPAL6 PAL16L8
*******************************

PIN MNEMONICS :

INPUTS:
A4A-9 -PC ADDRESS BUS
IOWB   -PC I/O WRITE, BUFFERED
IORB   -PC I/O READ, BUFFERED
TW     -WAIT STATE DELAYED I/P
OUTPUTS:
IOS    -I/O PROTOTYPE AREA DECODE
XRDY   -BUFFERED TO IORDY INTO PC
IOWR   -IOW OR IOR ACCESS

PIN DEFINITION:

AENB A9A A8A A7A A6A A5A A4A IORB IOWB GND
RESET NAND1 TW DCLR IN2 IN1 XRDY IOS-NOR1 VCC

EQUATIONS

/IOS=/IOWB*/AENB*/A4A*/A5A*/A6A*/A7A*A8A*A9A+
       /IORB*/AENB*/A4A*/A5A*/A6A*/A7A*A8A*A9A

/XRDY=IOS

/DCLR=TW+RESET
```

```
BLOW SPARE  GATES
/NAND1=IN1*IN2
/NOR1=IN1+IN2

*******************************
CHIP            HSTPAL7 PAL16L8
*******************************

PIN MNEMONICS :

INPUTS:

AOB-A3B -BUFFERED PC ADDRESS, LO BITS
IOWB    -BUFFERED PC I/O WRITE
IORB    -BUFFERED PC I/O READ          (NOT USED HERE )
AEN     -BUFFERED PC ADDRESS ENABLE (NOT USED HERE)
RESET   -BUFFERED PC RESET          (NOT USED HERE )

OUTPUTS:

ENABLE LO:
ADHI   -HI ADDRESS LATCH
ADMID  -MID ADDRESS LATCH
ADLO   -LO ADDRESS LATCH
MODA   -MODE LATCH A
MODB   -MODE LATCH B
MODC   -MODE LATCH C
DATA   -DATA PORT
STAT   -STATUS PORT

PIN DEFINITION:

AOB A1B A2B A3B IOWB IORB AENB IOS RESET GND
SPR1 STAT DATA MODC MODB MODA ADLO ADMID ADHI VCC

EQUATIONS

/ADHI=/AOB*/A1B*/A2B*A3B*/IOS*/IOWB

/ADMID=AOB*/A1B*/A2B*A3B*/IOS*/IOWB

/ADLO=/AOB*A1B*/A2B*A3B*/IOS*/IOWB

/MODA=AOB*A1B*/A2B*A3B*/IOS*/IOWB

/MODB=/AOB*/A1B*A2B*A3B*/IOS*/IOWB

/MODC=AOB*/A1B*A2B*A3B*/IOS*/IOWB
```

```
/DATA=/A0B*A1B*A2B*A3B*/IOS+/A3B*/IOS

/STAT=A0B*A1B*A2B*A3B*/IOS*IOWB

*******************************
CHIP             HSTPAL8 PAL16R4
*******************************

PIN MNEMONICS :

INPUT:
D0C-D7C -BUFFERED PC DATA
MODB   -PULSE LOW I/O SELECT
CLK    =MODB
OUTPUT:
RST  -LOCAL RESET *
ARM  -ARM ENABLE1 *
UNO  -LOW FOR NORMAL, HIGH FOR TERMINATE ON SINGLE LINE
CYC  -HIGH FOR CONTINUOUS, LOW FOR FINISH ON TC
CNTADDR -LOW FOR SELECT COUNTER FOR ADDRESS, HIGH
         FOR EXTERNAL
AXS   -LOW FOR SAMPLE AND DISPLAY, HIGH FOR ACCESS
START -IMMEDIATE SAMPLE START IF AXS LOW *
* INDICATES PULSE LOW, OTHERWISE LEVEL.

PIN DEFINITION:

CLK D0C D1C D2C D3C D4C D5C D6C MODB GND
EN NC START AXS CNTADR CYC UNO ARM RST VCC

EQUATIONS

/RST=/MODB*D0C

/ARM=/MODB*D1C

/UNO:=D2C        CLOCKED IN BY CLK=MODB

/CYC:=D3C         ..      ..        ..

/CNTADR:=D4C      ..      ..        ..

/AXS:=D5C         ..      ..        ..

/START=/MODB*D6C

*******************************
```

```
CHIP            HSTPAL9A PAL16L8
*******************************

PIN MNEMONICS :

INPUT:

SEL0-3, ENABLE1[HI] SELECT EVENT FOR TRIGGER
        CONTROL BUFFER DIRECTION
        ENABLE1[LO] SELECT CARD TO DAC
A0B-A3B SELECT CARD ON PC ACCESS
AXS     [LO] ACQUIRE/DISPLAY [HI] PC ACCESS
IOWB    PC BUFFERED IO WRITE
WR      ACQUIRE WRITE
ENABLE1 ARMED,WAITING EVENT OR START
IOSB    BUFFERED I/O SELECT

OUTPUT:

B_DIR  DATA BUFFER CONTROL
B_ENB
WE     MEMORY CONTROL
OE
EN_ADC enable video buffer onto bus.

PIN DEFINITION:

SEL0 SEL1 SEL2 SEL3 A0B A1B A2B A3B IOSB GND
ENABLE1 B_DIR B_ENB AXS IOWB WR EN_ADC WE OE VCC
EQUATIONS

SEL0-SEL3 COMBINATION DEPENDS ON ABSOLUTE CARD NUMBER
EXAMPLE IS FOR CARD 1

/EN_ADC=ENABLE1

ENABLE HI--- ARMED FOR ACQUIRE  LO--- DISPLAY

/OE=/ENABLE1*WR  CARD ADDRESS IS DECODED BY DATA
                 BUFFER,B_ENB

A0B-A3B COMBINATION AGAIN RELATES TO CARD POSITION

/WE=/AXS*/WR+AXS*/WR*/A0B*/A1B*/A2B*/A3B*/IOSB

/B_ENB=/AXS*/SEL0*/SEL1*/SEL2*/SEL3+
      AXS*/A0B*/A1B*/A2B*/A3B*/IOSB
```

```
/B_DIR=/AXS+AXS*WR

*******************************
CHIP           HSTPAL9B PAL16L8
*******************************

EQUATIONS

/EN_ADC=ENABLE1

/OE=/ENABLE1*WR

/WE=/AXS*/WR+AXS*/WR*A0B*/A1B*/A2B*/A3B*/IOSB

/B_ENB=/AXS*SEL0*/SEL1*/SEL2*/SEL3+
      AXS*A0B*/A1B*/A2B*/A3B*/IOSB

/B_DIR=/AXS+AXS*WR

*******************************
CHIP           HSTPAL9C PAL16L8
*******************************

EQUATIONS

/EN_ADC=ENABLE1

/OE=/ENABLE1*WR

/WE=/AXS*/WR+AXS*/WR*/A0B*A1B*/A2B*/A3B*/IOSB

/B_ENB=/AXS*/SEL0*SEL1*/SEL2*/SEL3+
      AXS*/A0B*A1B*/A2B*/A3B*/IOSB

/B_DIR=/AXS+AXS*WR

*******************************
CHIP           HSTPAL9D PAL16L8
*******************************

EQUATIONS

/EN_ADC=ENABLE1

/OE=/ENABLE1*WR

/WE=/AXS*/WR+AXS*/WR*A0B*A1B*/A2B*/A3B*/IOSB
```

```
/B_ENB=/AXS*SEL0*SEL1*/SEL2*/SEL3+
      AXS*A0B*A1B*/A2B*/A3B*/IOSB

/B_DIR=/AXS+AXS*WR

*******************************
CHIP            HSTPAL9E PAL16L8
*******************************

EQUATIONS

/EN_ADC=ENABLE1

/OE=/ENABLE1*WR

/WE=/AXS*/WR+AXS*/WR*/A0B*/A1B*A2B*/A3B*/IOSB

/B_ENB=/AXS*/SEL0*/SEL1*SEL2*/SEL3+
      AXS*/A0B*/A1B*A2B*/A3B*/IOSB

/B_DIR=/AXS+AXS*WR

*******************************
CHIP            HSTPAL9F PAL16L8
*******************************

EQUATIONS

/EN_ADC=ENABLE1

/OE=/ENABLE1*WR

/WE=/AXS*/WR+AXS*/WR*A0B*/A1B*A2B*/A3B*/IOSB

/B_ENB=/AXS*SEL0*/SEL1*SEL2*/SEL3+
      AXS*A0B*/A1B*A2B*/A3B*/IOSB

/B_DIR=/AXS+AXS*WR

*******************************
CHIP            HSTPAL9G PAL16L8
*******************************

EQUATIONS
/EN_ADC=ENABLE1

/OE=/ENABLE1*WR
```

```
/WE=/AXS*/WR+AXS*/WR*/A0B*A1B*A2B*/A3B*/IOSB

/B_ENB=/AXS*/SEL0*SEL1*SEL2*/SEL3+
      AXS*/A0B*A1B*A2B*/A3B*/IOSB

/B_DIR=/AXS+AXS*WR

*******************************
CHIP           HSTPAL9H PAL16L8
*******************************

EQUATIONS

/EN_ADC=ENABLE1

/OE=/ENABLE1*WR

/WE=/AXS*/WR+AXS*/WR*A0B*A1B*A2B*/A3B*/IOSB

/B_ENB=/AXS*SEL0*SEL1*SEL2*/SEL3+
      AXS*A0B*A1B*A2B*/A3B*/IOSB

/B_DIR=/AXS+AXS*WR

********************************
CHIP            HSTPAL10 PAL16L8
********************************

PIN MNEMONICS :

INPUT:

IOS = VALID PORT ADDRESS FOR I/F
A3 = BUFFERED PC ADDRESS BIT 3
Smn = SEGMENT m COMPARATOR O/P RACK n

OUTPUT:

BUFFEN = ENABLE DATA BUFFER
EV = EVENT TRIGGER

PIN DEFINITION:

S11 S21 S12 S22 S13 S23 IOS W1 W2 GND
A3 EV BUFEN AXS I1 O1 BUFDIR IOWB NC VCC

EQUATIONS
```

```
/BUFEN =/IOS   USE FOR QUIET OPERATION, NO D-A,
               TIE ENABLE LOW TO ENABLE D-A DATA ON
               BUS DURING IDLE.
/BUFDIR =/AXS+AXS*IOWB  DATA NORMALLY OUT FROM BUS,
                        IN ON I/F WRITE

/EV=S11*/S21*S12*/S22*/W1*/W2+
   S12*/S21*S13*/S23*/W1*/W2+
   S22*/S22*S13*/S23*/W1*/W2

/O1=I1

********************************
CHIP            HSTPAL11 PAL16L8
********************************

PIN MNEMONICS :

INPUT:

QA-QH  SCAN WINDOW COUNTER O/P
PPL 1 END OF LINE SCAN (PIXELS PER LINE PULSE)
CLKC  SYSTEM CLOCK

OUTPUT:

WNDW1 LOW AFTER START OF SCAN, (HOLD-OFF)
WNDW2 LOW FOR DURATION OF SCAN, (SPAN)

RCLK REGISTER CLOCK FOR WINDOW COUNTER
CLK WINDOW COUNTER CLOCK
CCLR CLEAR COUNTER
NOTE:- ADJUSTMENTS TO PROGRAMMING OF PAL11 MAY
BE REQUIRED.. FURTHER DEVELOPMENT ON EVENT
TRIGGERING MAY BE REQUIRED.(TESTS NOT COMPLETE 11/12/90)

PIN DEFINITION:

QA QB QC QD QE QF QG QH CLKC GND
PPL RCLK CCLR CCLK NC NC NC WNDW2 WNDW1 VCC

EQUATIONS

HOLD OFF FOR 4 PIXELS - (MAY NEED ADJUSTING)

/WNDW1=QC+QD+QE+QF+QG+QH

SPAN TOTAL 128 PIXELS
```

```
COULD ALL BE DONE WITH WNDW1 BUT SEPARATE
SIGNAL MAY BE USEFUL

/WNDW2=QH

/RCLK=/CLKC

/CCLK=/CLKC

/CCLR=/PPL

*********************************
CHIP            HSTPAL12A PAL16L8
*********************************

PIN MNEMONICS :

INPUT:

IOS = VALID PORT ADDRESS FOR I/F
A0-3 = BUFFERED PC ADDRESS BITS 0-3
AXS = ACCESS/AQUIRE MODE BIT
SR0-3 = SELECT RACK BITS
IOWB = PC WRITE

OUTPUT:

BUFFEN = ENABLE DATA BUFFER
BUFDIR = DIRECTION
CARDCLK = DISPLAY CARD SELECT LATCH CLOCK

PIN DEFINITION:

SR0 SR1 SR2 SR3 A0 A1 A2 A3 AXS GND
IOS NC IOWB  CLK1 CLK2 NC CARDCLK BUFDIR BUFEN VCC

EQUATIONS

/BUFEN = /SR0*/SR1*/SR2*/SR3

/BUFDIR = /AXS+AXS*IOWB

/CARDCLK = /A0*A1*A2*A3*/IOS*/IOWB

/CLK2 = CLK1

*********************************
CHIP            HSTPAL12B PAL16L8
```

```
********************************

EQUATIONS

/BUFEN = SR0*/SR1*/SR2*/SR3

/BUFDIR = /AXS+AXS*IOWB

/CARDCLK = /A0*A1*A2*A3*/IOS*/IOWB

/CLK2 = CLK1

********************************
CHIP            HSTPAL12C PAL16L8
********************************

EQUATIONS

/BUFEN = /SR0*SR1*/SR2*/SR3

/BUFDIR = /AXS+AXS*IOWB

/CARDCLK = /A0*A1*A2*A3*/IOS*/IOWB

/CLK2 = CLK1

**************
end
**************
```

Appendix E

Operating System Software

The interface operating programme has been split into three units to avoid problems with memory restrictions while developing the software. The units are: definitions, miscellaneous and main subroutines.

E.1 Variable Definition Unit

```
{High Speed Tracking V1.00 in Turbo Pascal V5.1 11/12/90}
{******  system constants & variables  ******}

{$M 32000,0,655360}
{ IMPORTANT - line buffer array is set for 128 pixels and 3 ref elements
  per segment. This must be altered if hardware changes are made. }

unit hstvars;

interface

const
   pixset=128;    {pixels / seg}
   refset=3;      {ref elements / seg}
   qtyset=50;     {lines/ buf, lines displayed}

type
   tenindx = 0..9;
   seg_entry = record
                pix:array[1..pixset] of byte;
                ref:array[1..refset] of byte
               end;
   ln_entry  = record
ln_dig:integer;
                seg:array[1..8] of seg_entry
               end;
   data1 =array[1..qtyset] of ln_entry;
   storevars = record
                vers,
                sv_mode :integer;
                buf_indx:tenindx;
                pixp_seg,       {active pixels per seg}
                refp_seg,       {ref pixels per seg   }
               segp_ln,         {segs per line          }
               lnper_buf,       {lines per buffer       }
               bufp_file,       {buffers per file       }
               filep_rack,      {files per rack         }
               rackp_sys,       {racks total            }
               total_lns,       {total number of lines/rack}
               rec_size         :integer; {total record size}
               pix_clk,
               hold_clk         :byte;
               bufsize          :word;
             end;
 dispvars = record
ln_indx,       {line offset into buffer}
                maxx,           {max screen x,y}
maxy,
                orgx,           {on screen fixed base org}
                orgy,
                refy,
                refx,           {current trace org}
qtylines,      {qty of lines plotted}
                maxln,          {current line buffer limits}
                minln,
                maxlnf,         {current file limits}
                minlnf,
                ln_num,         {absolute line number}
                max_trace,      {display trace length}
                pix_inc,        {pix increment,jump over pixels}
                seg_start,      {current display, start seg}
                qty_seg,        {segs to plot}
                ln_inc,         {line increment, skip lines     }
                x_inc,          {x-axis displacement            }
y_inc,
cmdx1,         {command box co-ords}
                cmdy1,
                cmdx2,
                cmdy2,
                txtx,           {cmd text base}
                txty,
                boxx,           {location box org}
boxy,
                oldpos,
                ln_abs,
                start_seg,
                seg_indx,
                cursx,
                cursy,
                curs_inc,
                o_cursx,
                 o_cursy,
                 frame,
                 lnper_file     :integer;
                 hide_flag,
                 u_mode,
                 f_open         :boolean; {file open flag}
                 rack_indx,
                 file_indx,
                 buf_indx       :tenindx;
                 sizex,sizey    :word;
                 cursyptr,
                 cursxptr       :pointer;
             end;

const
  vers_ok = 10;         {current s/w version}
{i/f ports}
  basep=$300;           {pc prototype card allocation}
  base1=basep+8;
  addhi=basep+8;
  addmid=basep+9;
  addlo=basep+10;
  moda=basep+11;
  modb=basep+12;
  modc=basep+13;
  xdata=basep+14;
  io_status=basep+15;
{ control mask values :}
  b_reset=1;     {rst}
  b_arm=2;       {arm}
  b_clear=12;    {clear counter}
  b_uno=4;       {cycle until tc ( not single line ) }
  b_cyc=8;       {stop on tc ( not continuous cycle ) }
  b_ext_addr=16; {bus addressed by counter ( not latches )}
  b_xdata=32;    {acquire or display mode ( not external data access )}
  b_start=64;    {start  }
  v_reset=61;    {1+4+8+16+32}
  v_arm=2;
  v_start=64;
  v_clear1=0;
  v_clear2=12;
  v_uno=0;
  v_cyc=0;
  v_xdata=0;
  v_int_addr=16;
  v_ext_addr=0;
  init_stat=60;    {initial control status port -modb}
  init_moda=0;     {initial clock, card select - 6MHz and card 1}
  boxwidth=134;    {used by graphics procs}
  grmode_val=3;

 var
   o_count,a1,a2,a3,a4,n3:longint;
   o_color,grdriver,grmode,n1            :integer;
   init_modc,in_int,new_stat,
   hold_stat,s_bit,o_fill,color,
   s_addhi,s_addmid,s_addlo,bz,
   o_stat,o_addhi,o_addmid,o_addlo,
   o_sel,o_moda,o_modb,o_modc,o_clk      :byte;
   o_port,port_num,
   c_red,c_white,c_green                 :word;
   ch                                    :char;
   out_filen,in_filen,
   o_filen,o_string, pathstr             :string;
   out_fbuf,in_fbuf                      :file;
   prompt,g_flag,flag                    :boolean;
   storev                                :storevars;
   dispv                                 :dispvars;
   minpix,o_minpix                       :array[0..639] of integer;
   ln_buf                                :data1;
implementation
end.
```

E.2 Miscellaneous Procedures/ Functions

```
{HST V1.0 misc i/f control in Turbo Pascal V5.1   11/12/90}

 unit hstio;

 interface

 uses crt,dos,graph,hstvars;

procedure wr_str(mesg:string);
procedure write_byte(bt:byte);
procedure write_word(bt:word);
function conv2hex(s1:string;VAR result:longint):boolean;
function get_hex:byte;
function get_num(max_num:integer):integer;
procedure get_io_port;
procedure set_clk(bt:byte);
procedure wr_io_port;
procedure rd_io_port;
procedure set_rack(b:byte);
procedure set_lvl_io;
procedure pulse_io;
procedure init_io;
procedure default_rst;
function enable1:boolean;
function arm_acquire(noisy:boolean):boolean;
procedure start_acquire(noisy:boolean);
procedure clear_count(noisy:boolean);
procedure set_uno_line;
procedure set_ppl;
procedure set_cyc_tc;
procedure sifm_ext_axs;
procedure sifm_int_addr;
procedure sifm_ext_addr;
procedure set_pix_clk;
procedure store_if_addr;
procedure un_store_if_addr;
procedure get_if_addr;
procedure set_if_addr;
procedure adj_if_addr(r1:longint);
procedure bump_if_addr;
procedure write_if_addr;
procedure flip_test1;
procedure set_seg(segno:byte);
procedure gset_seg;
function check_space(size:integer):boolean;
function get_stp(delta:integer):integer;
procedure tick(x,y,length,dir:integer);
procedure DrawLine (x1, y1, x2, y2,f1,f2,dir: integer);
procedure restore_clk;
procedure fetch_data;
function scan_nseg:integer;

implementation

procedure wr_str(mesg:string);
{used by procs in text or graphics mode}
var
  t_str:text;
  size:word;
  p:pointer;
  vp:viewporttype;
begin
   if g_flag then
    begin
       sound(1000);delay(50);nosound;
       getviewsettings(vp);
       setviewport(0,0,getmaxx,getmaxy,true);
       size:=imagesize(0,0,getmaxx-16,14);
```

```
        {fit in graph legend box}
        getmem(p,size);
        getimage(0,0,getmaxx-16,14,p^);
        setviewport(21,1,getmaxx-18,14,clipoff);
        clearviewport;
        setcolor(cyan);
        outtextxy(40,3,mesg);
        delay(1500);
        setviewport(0,0,getmaxx,getmaxy,true);
        putimage(0,0,p^,normalput);
        freemem(p,size);
        setviewport(vp.x1,vp.y1,vp.x2,vp.y2,vp.clip);
     end
     else begin
       sound(500);delay(50);nosound;
       writeln(mesg);
       delay(500)
    end
end;

procedure write_byte(bt:byte);
{writes byte to screen in ascii}
const
    hex_digits:array[0..15] of char = '0123456789ABCDEF';
var
    bz:byte;
begin
   bz:=bt and $0f;
   bt:=bt shr 4;
   write(hex_digits[bt],hex_digits[bz])
end;

procedure write_word(bt:word);
{writes word to screen in ascii}
var
  by,bz:byte;
begin
   by:=bt div 256;
   bz:=bt mod 256;
   write_byte(by);
   write_byte(bz)
end;

function conv2hex(s1:string;VAR result:longint):boolean;
{converts input string to big number}
const
 hex_set = '0123456789abcdefABCDEF';
var
```

```
 s2:string;
 n1,n2,n3:integer;
 n4,n5:longint;

begin
   conv2hex:=false; {set up for fail exit}
   n4:=1;  {multiplier}
   n5:=0;   {accumulator}
   for n2:=1 to length(s1) do
    begin
       s2:=copy(s1,(length(s1)-n2+1),1);
       n3:=pos(s2,hex_set)-1;
       if n3<0 then exit;
       if n3>15 then n3:=n3-6;
       n5:=n5+n4*n3;
       n4:=n4*16
   end;
   result:=n5;
   conv2hex:=true
end;

function get_hex:byte;
{gets hex number from user}
var
  s1:string;
  result:longint;
begin
   repeat
        write(' ');
        readln(s1);
   until (conv2hex(s1,result)=true) and (result<256);
   get_hex:=result
end;

function get_num(max_num:integer):integer;
{gets number within limits from user}
var
 n:integer;
begin
   repeat
        read(n);
   until (n < max_num);
   get_num:=n
end;

procedure get_io_port;
{get io port number from user}
var
```

```
  result:longint;
  bt:byte;
  s1:string;
begin
   repeat
        writeln;
        write('Port no. :');
        readln(s1);
   until (conv2hex(s1,result)=true) and (result<256);
   bt:=result;
   port_num:=basep+bt;
   o_port:=port_num;
   write('New port: $');
   write_word(port_num);
 case bt of
   0:s1:=' segment 1.';
   1:s1:=' segment 2.';
   2:s1:=' segment 3.';
   3:s1:=' segment 4.';
   4:s1:=' segment 5.';
   5:s1:=' segment 6.';
   6:s1:=' segment 7.';
   7:s1:=' segment 8.';
   8:s1:=' high address.';
   9:s1:=' mid address.';
   10:s1:=' low address.';
   11:s1:=' mode A latch.';
   12:s1:=' mode B latch.';
   13:s1:=' mode C latch.';
   14:s1:=' aux. data.';
   15:s1:=' status port.'
  else s1:=' invalid.';
  end;
  writeln(s1)
end;

procedure set_clk(bt:byte);
{set pixel and system clock}
 begin
   {set clock, preserve non clock bits}
   bt:=bt and $07;
   o_moda:=o_moda and $f8; {rst clk bits}
   o_moda:=o_moda or bt; {set clk bits}
   port[moda]:=o_moda;
   o_clk:=o_moda;
   with storev do
   begin
      pix_clk:=bt;
   end
end;


procedure wr_io_port;
{write to preset port}
begin
   set_clk(0);
   write(':');
   port[port_num]:=get_hex;
   set_clk(storev.hold_clk)
end;

procedure rd_io_port;
{read from preset port}
var
 n1:byte;
begin
   set_clk(0);
   n1:=port[port_num];
   write('[ $');
   write_word(port_num);
   write(']=');
   write_byte(n1);
   writeln;
   set_clk(storev.hold_clk)
end;

procedure set_rack(b:byte);
begin
   b:=b-1;
   b:= b shl 4; {get offset into correct field}
   o_moda:=o_moda and $0f; {clear sel0-3 bits}
   o_moda:=o_moda or b;
   port[moda]:=o_moda
end;

procedure set_lvl_io;
{set bit at i/f command port}
begin
   o_stat:=o_stat and (255-s_bit);
   o_stat:=o_stat or new_stat;
   port[modb]:=o_stat;
   o_modb:=o_stat
end;

procedure pulse_io;
{pulse bit at i/f command port}
```

```
var
 n:integer;
begin
   hold_stat:=o_stat;
   set_lvl_io;
   o_stat:=hold_stat;
   port[modb]:=o_stat;
   o_modb:=o_stat;
   delay(100) {***}
end;

procedure init_io;
{initialise io ports}
begin
   port[modb]:=o_modb;
   port[modc]:=o_modc;
   o_stat:=init_stat
end;

procedure default_rst;
{reset hardware}
begin
   init_io;
   new_stat:=v_reset;
   s_bit:=b_reset;
   pulse_io;
   DISPV.u_mode:=false
end;

function enable1:boolean;
{mask of cycle done bit from i/f port}
begin
   enable1:=(port[io_status] and 4)=4
end;

function arm_acquire(noisy:boolean):boolean;
{set up port for acquire cycle}
begin
   arm_acquire:=true;
   port[modb]:=o_stat and (255-b_cyc);
   new_stat:=v_arm;
   s_bit:=b_arm;
   pulse_io;
   if noisy then wr_str('Arming.. ');
   if enable1 then
   begin
     if noisy then wr_str('Ready for start or trigger')
   end
   else begin
     wr_str('HST i/f not responding.');
     arm_acquire:=false
 end;
 {inhibit tc to avoid conflict with arm pulse}
 port[modb]:=o_stat or b_cyc
end;

procedure start_acquire(noisy:boolean);
{force start cycle, overide wait for external event}
begin
   new_stat:=v_start;
   s_bit:=b_start;
   pulse_io;
   if noisy then wr_str('Started acquire.');
   repeat
   until NOT enable1 or keypressed;
   if noisy then wr_str('Done...  (T)ransfer to save to disk. ');
   o_addhi:=0; {prepare for blk re-read if requested}
   o_addmid:=0;
   o_addlo:=0
end;

procedure clear_count(noisy:boolean);
{clear i/f address counters}
begin
   new_stat:=v_clear1;
   s_bit:=b_clear;
   pulse_io;
   if noisy then wr_str('Counters cleared')
end;

procedure set_uno_line;
{set i/f to stop after 1 line}
begin
   new_stat:=v_uno;
   s_bit:=b_uno;
   set_lvl_io;
   DISPV.u_mode:=true
end;

procedure set_ppl;
{select pixels per line via i/f logic}
var
 bt:integer;
 bz:byte;
begin
   write('-no of pixels/line (128-131)');
   readln(bt);
 case bt of
    128:bz:=00;
    129:bz:=$10;
    130:bz:=$20;
    131:bz:=$30;
    else bz:=00
 end;
  port[modc]:=bz or init_modc;  {set test bit send selp0-1}
  o_modc:=bz or init_modc
end;

procedure  set_cyc_tc;
{set stop on terminal count, before memory wraparound}
begin
   new_stat:=v_cyc;
   s_bit:=b_cyc;
   set_lvl_io
end;

procedure sifm_ext_axs;
{set i/f mode for external access}
begin
   new_stat:=v_xdata;
   s_bit:=b_xdata;
   set_lvl_io
end;

procedure sifm_int_addr;
{set i/f mode for internal, counter addressing}
begin
   new_stat:=v_int_addr;
   s_bit:=b_ext_addr;    {?}
   set_lvl_io
end;

procedure sifm_ext_addr;
{set i/f mode for external, latch addressing}
begin
   new_stat:=v_ext_addr;
   s_bit:=b_ext_addr;
   set_lvl_io
end;

procedure set_pix_clk;
var
bz:byte;
begin
```

```
   wr_str('Select 0) to 7) for 6 MHz, 3 MHZ...46.875KHz : ');
   bz:=get_num(8);
   storev.hold_clk:=bz;
   set_clk(bz);
   with storev do
   if (bz<2) then refp_seg:=3 else refp_seg:=2
end;

procedure store_if_addr;
begin
   o_addhi:=s_addhi;
   o_addmid:=s_addmid;
   o_addlo:=s_addlo
end;

procedure un_store_if_addr;
begin
   s_addhi:=o_addhi;
   s_addmid:=o_addmid;
   s_addlo:=o_addlo
end;

procedure adj_if_addr(r1:longint);
{convert a big number into byte-wide address segments}
var
  n1,n2,n3:byte;

begin
   n1:=r1 div 65536;
   r1:=r1-n1*65536;
   n2:=r1 div 256;
   n3:=r1 mod 256;
   s_addhi:=n1;
   s_addmid:=n2;
   s_addlo:=n3
end;

procedure get_if_addr;
{get address from user}
var
  result:longint;
  s1:string;

begin
  repeat
     write('-Address(hex): ');
     write(' ');
     readln(s1);
```

```
 until (conv2hex(s1,result)=true) and (result<131072);
 adj_if_addr(result);
end;

procedure set_if_addr;
{set address at i/f latches}
begin
   port[addhi]:=s_addhi;
   port[addmid]:=s_addmid;
   port[addlo]:=s_addlo
end;

procedure bump_if_addr;
{increment i/f address latches}
begin
  inc(s_addlo);
   if s_addlo=0 then
    begin inc(s_addmid);
      if s_addmid=0 then inc(s_addhi)
   end;
  set_if_addr
end;

procedure write_if_addr;
{display current address}
begin
   write_byte(s_addhi);
   write_byte(s_addmid);
   write_byte(s_addlo);
   write(': ')
end;

procedure flip_test1;
{set test bit, wait /nowait for diode sync..
 used for debugging}
var
bz:byte;
begin
 bz:=o_modc and $80;
 if bz=0 then
  begin
   writeln('Setting Test1 on');
   bz:=o_modc or $80;
  end
  else
  begin
   writeln('Setting Test1 off');
   bz:=o_modc and $7f
  end;
 port[modc]:=bz;
 o_modc:=bz
end;

procedure set_seg(segno:byte);
{select segment/channel accessed via i/f}
begin
   sifm_ext_axs;
   {need to set 'axs' bit to load card register in rack}
   port[xdata]:=segno;
   default_rst
end;

procedure gset_seg;
{get required segment from user}
{set card for display and for block r/w functions}
var
  bt:byte;
begin
   repeat
     writeln;
     write('Which card (1 to 8) ');
     readln(bt);
   until (bt>0) and (bt<9);
   bt:=bt-1;
   set_seg(bt);
   port_num:=basep+bt
end;

function check_space(size:integer):boolean;
{check disk space available for data file storage}
begin
 if ((maxavail)>size) then check_space:=true
 else
 begin
  wr_str('Not enough free ram for data buffer');
  check_space:=false;
 end;
end;

function get_stp(delta:integer):integer;
{used in debug}
begin
  if delta< 0 then get_stp:= -1
  else get_stp:= 1
end;
```

```
procedure tick(x,y,length,dir:integer);
{provides axis tick for FILE display}
{direction variable values as shown..}
{dir :    1  2  3
          4     5
          6  7  8}
var
 dirx,diry:integer;
begin
  case dir of
   1:begin dirx:=-1;diry:=-1;end;
   2:begin dirx:=0;diry:=-1;end;
   3:begin dirx:=1;diry:=-1;end;
   4:begin dirx:=-1;diry:=0;end;
   5:begin dirx:=1;diry:=0;end;
   6:begin dirx:=-1;diry:=1;end;
   7:begin dirx:=0;diry:=1;end;
   8:begin dirx:=1;diry:=1;end;
  else begin dirx:=0;diry:=0;end
  end;
  setcolor(c_white);
  line(x,y,x+length*dirx,y+length*diry);
  setcolor(o_color)
end;


procedure DrawLine (x1, y1, x2, y2,f1,f2,dir: integer);
{Bresenham's Algorithm f1,f2 tick frequency,dir.}
{used to allow later development for plotter drive
 and allow hidden line removal option}
var
  n1,n2,b_tic,l_tic,
  step_x, step_y,
  x,y,dx, dy,
  d, incr1, incr2,
  f1_count,f2_count: integer;
  f1_flag,f2_flag:boolean;
begin
  b_tic:=5;
  l_tic:=2;
  x := x1;
  y := y1;
  dx := x2 - x1;
  dy := y2 - y1;
  step_x:=get_stp(dx);
  dx:=dx*step_x;
  step_y:=get_stp(dy);
  dy:=dy*step_y;
```

```
if f1 >0 then f1_flag:=true
else f1_flag:=false;
if f2 >0 then f2_flag:=true
else f2_flag:=false;
if dir=0 then {auto detect direction}
begin
  dir:=7; {default to down facing ticks }
  if y1>y2 then dir:=5; {set for left facing ticks}
  if y1<y2 then dir:=4; { right }
end;
if (f1_flag) or (f2_flag) then tick(x,y,b_tic,dir);
f1_count:=1;
f2_count:=1;
with DISPV do
begin
if dy < dx then
begin
     incr2 := (dy - dx) shl 1;
     d:= incr2 + dx;
     incr1 := d + dx;
     while x <> x2 do
     begin
         x := x + step_x;
         if d < 0 then d := d + incr1
         else
         begin
            d := d + incr2;
            y := y + step_y
         end;
         {f1 or f2 set, assume draw axis mode}
         if f1_flag or f2_flag then putpixel(x,y,color)
         else begin
              if DISPV.hide_flag then
              begin
                 if (y<o_minpix[x]) then
                 begin
                    putpixel(x,y,color);
                    if (y<minpix[x]) then minpix[x]:=y
                 end
              end
         end;
         if f1_flag then
         begin
           inc(f1_count);
           if (f1_count>f1)  then
           begin
              f1_count:=1;
              tick(x,y,l_tic,dir);
            end
          end;
          if f2_flag then
          begin
            inc(f2_count);
            if (f2_count>f2) then
            begin
               f2_count:=1;
               tick(x,y,b_tic,dir);
            end
          end
      end
end    {end dy<dx}
else
begin  {dy>dx}
     incr2 := (dx - dy) shl 1;
     d:= incr2 + dy;
     incr1 := d + dy;
     while y <> y2 do
     begin
        y := y + step_y;
        if d < 0 then d := d + incr1
        else
        begin
            d := d + incr2;
            x := x + step_x;
        end;
       { putpixel(x,y,color); }
        if f1_flag or f2_flag then putpixel(x,y,color)
        else begin
              if DISPV.hide_flag then
              begin
                 if (y<o_minpix[x]) then
                 begin
                    putpixel(x,y,color);
                    if (y<minpix[x]) then minpix[x]:=y
                 end
              end
        end;
        if f1_flag then
        begin
          inc(f1_count);
          if (f1_count>f1)  then
          begin
             f1_count:=1;
             tick(x,y,l_tic,dir);
          end
        end;
```

```
          if f2_flag then
          begin
            inc(f2_count);
            if (f2_count>f2) then
            begin
               f2_count:=1;
               tick(x,y,b_tic,dir);
            end
          end
      end
  end  {end dy>dx}
  end  {with DISPV do}
end;

procedure restore_clk;
{restores clock to preset value, adjusts ref pixel count
 to compensate for timing anomaly}
begin
   with storev do
   begin
      set_clk(hold_clk);
      if (hold_clk<2) then refp_seg:=3 else refp_seg:=2
   end
end;

procedure fetch_data;
{fetches data for 1 segment only, 10 lines}
{pix clk changed to 6MHz during data transfer}
var
 pix_indx,ref_indx:integer;
begin
   s_addhi:=0;
   s_addmid:=0;
   s_addlo:=0;
   sifm_ext_addr;
   sifm_ext_axs;
   set_clk(0);
   with STOREV do
   begin
      with DISPV do
      begin
         for ln_indx:=1 to 10 do
         begin
            with LN_BUF[ln_indx] do
            begin
       ln_dig:=ln_indx;
       for pix_indx:=1 to pixp_seg do
       begin
```

```
          set_if_addr;
          seg[start_seg].pix[pix_indx]:=port[port_num];
          bump_if_addr
       end;
       for ref_indx:=1 to refp_seg do
       begin
          set_if_addr;
          seg[start_seg].ref[ref_indx]:=port[port_num];
          bump_if_addr
       end
            end   {with LN_BUF}
         end      {In_index loop}
      end         {with DISPV}
   end;           {with STOREV}
   restore_clk;   {restore pixel clock}
   default_rst    {hand back address bus and access to i/f}
end;

function scan_nseg:integer;
{scans all segs in preset rack, checks pixels against
 a threshold}
var
  pix,segn,n,dudsegs,dudpix,
  maxdud,threshold,pixval:integer;
  s:string;
begin
   threshold:=50; {set near saturation value}
   dudsegs:=0;    {counter}
   maxdud:=3;     {number of invalid segments tolerated}
   s:='Beam profile [';
   n:=storev.pixp_seg-1;
   dispv.start_seg:=1;
   for segn:=1 to 8 do
   begin
      port_num:=basep+segn-1; {set port id}
      fetch_data;             {get some data from adc ram}
      dudpix:=0;
      if keypressed then exit;
      for pix:=4 to n do  {start 2nd line in}
        begin
           pixval:=ln_buf[2].seg[1].pix[pix];
           if (pixval>threshold) or (pixval=0)
            then inc(dudpix)
        end;
        if (dudpix<maxdud) then s:=s+'*'
         else begin
                s:=s+'-';
                inc(dudsegs)
          end
      end;
      write(s+']');
      scan_nseg:=dudsegs
   end;
   end.
```

E.3 The Main Program

```
{HST main programme V1.00 in Turbo Pascal V5.1  11/12/90}
{NOTE - refp_seg dec. for pix clock below 3MHz}

uses crt,dos,graph,hstvars,hstio;

procedure  acquire_cycle(flag:boolean);
{runs acquire cycle}
var
  bz:byte;
begin
{clear TEST1 bit-- wait for diode sync}
   bz:=o_modc and $7f;
   port[modc]:=bz;
   if arm_acquire(flag) then
   begin
      clear_count(flag);
      start_acquire(flag)
  end
end;

procedure chk_vid;
{checks video levels on racks 1 to 2, tries to
 establish which clock frequency works, indicates saturation
 status as it goes}
var
 n1,n2,result:integer;
begin
   for n1:=1 to 3 do
   begin
       set_rack(n1);
       writeln;
       writeln('Video Level-Check, Rack [',n1,'] :');
       n2:=0;
       repeat
           storev.hold_clk:=n2;
           set_clk(n2);
           with storev do
               if (n2<2) then refp_seg:=3 else refp_seg:=2;
           acquire_cycle(false);
           inc(n2);
           result:=scan_nseg;
           write(' Sampling at ');
           write(12000/(exp(ln(2)*(STOREV.pix_clk+1))):4:2);
           writeln (' KHz.');
       until (result=0) or (n2>7) or keypressed;
       if keypressed then ch:=readkey;
       if (result>7) and (n2>7) then
```

```
        begin
           write('No saturation level found..');
           write('check alignment');
           writeln('or sensor video output.')
       end;
       if (result>1) and (n2<8)
        then writeln('Poor saturation response.. check alignment.')
   end
end;

function blk_head:longint;
{sets up address,returns count reqd}
var
 n:longint;
begin
   readln(n);
   blk_head:=n;
   o_count:=n;
   get_if_addr;
   writeln;
   set_if_addr;
   sifm_ext_addr;
   sifm_ext_axs
end;

procedure blk_glitch;
{look for specified delta, 1 byte to the next-
  most effective with test ramp i/p}
var
  bv,bx,by,bz:byte;
  ba,bd:integer;
  bb,bc:real;
  lp_count,count,e_count:longint;
begin
   write('Glitch delta (hex)? :');
   bx:=get_hex;
   lp_count:=1;
   {count=round(ramsize/(pix+ref per line))}
   count:=131052;
   o_count:=count;
   e_count:=0;
   s_addhi:=0;
   o_addhi:=0;
   s_addmid:=0;
   o_addmid:=0;
   s_addlo:=0;
   o_addlo:=0;
   writeln;
   set_if_addr;
   sifm_ext_addr;
   sifm_ext_axs;
   writeln('Checking ram...');
   set_clk(0);
   by:=port[port_num];
   repeat
        bz:=port[port_num];
        ba:=(abs(bz-by));
        if ba>bx then
         begin
            ba:=ba*8;
            write_if_addr;
            writeln(' ',by,' >>',bz,' delta- ',ba,' mV');
            inc(e_count)
         end;
        bump_if_addr;
        by:=bz;
        inc(lp_count);
   until (lp_count > count) or keypressed ;
   restore_clk;
   writeln('Total count [',lp_count,']');
   if e_count>0 then writeln('Total found [',e_count div 2,']')
   else writeln('None found');
   default_rst
end;

procedure blk_read_body(count:longint);
{common part of block read from ram,
 display on screen}
var
  bz:byte;
  col_count:integer;
  lp_count:longint;
begin
   write_if_addr;
   lp_count:=1;
   col_count:=1;
   sifm_ext_axs; {use a0-a3 to select card in rack, not sel0-3}
   set_clk(0);
   repeat
        bz:=port[port_num]; {read data}
        write_byte(bz);     {display}
        write(' ');
        inc(col_count);
        bump_if_addr;
        inc(lp_count);
        if col_count=17 then
```

```
           if lp_count <= count then
            begin
               col_count:=1;
               writeln;
               write_if_addr
            end;
   until (lp_count > count) or keypressed;
   restore_clk;
   default_rst
end;

procedure blk_read;
begin
   write('Read block size (decimal)? :');
   blk_read_body(blk_head)
end;

procedure blk_write;
{block write to ram}
var
  fill_byte:byte;
  bz:byte;
  lp_count,count:longint;
begin
  lp_count:=1;
  write('Write block size? (decimal)');
  count:=blk_head;
  write('-Fill block byte? ');
  fill_byte:=get_hex;
  write('  processing...');
  sifm_ext_axs; {use a0-a3 to select card in rack}
  set_clk(0);
  repeat
       port[port_num]:=fill_byte;
       bump_if_addr;
       inc(lp_count);
  until (lp_count > count) or keypressed;
  restore_clk;
  default_rst;
  writeln('done')
end;

function check_dsk(size:longint):boolean;
{check space on disk, requirement depend
 on number of racks involved}
begin
   check_dsk:=true;
   wr_str('Checking disk space');
```

```
    if ((diskfree(0))<size) then
     begin
        write(' Insufficient disk space....');
        writeln('require ',size,' bytes free.');
        check_dsk:=false;
        exit
     end
end;

function get_filen(o_filen:string):string;
{get filename from user}
var
  filen,fn:string;
  fdir:dirstr;
  fname:namestr;
  fext:extstr;
begin
   fsplit(o_filen,fdir,fname,fext);
   {dump file extension
   add new extension}
   filen:=fdir+fname+'.M11';
   write('Type "r" to use TEST or recent filename [',filen,'] : ');
   readln(fn);
   if (fn='r') or (fn='R') then get_filen:=filen
    else
       if (length(fn)<3) or (pos(' ',fn)>0) then
        begin
           writeln('No file name supplied...quitting.');
           delay(500);
           get_filen:='';
           exit
        end
      else get_filen:=fn
end;

function open_out_file(filen:string;rec_size:word):boolean;
{opens file;if entry flag true and files exist, prompts for
 overwrite..otherwise overwrites without prompt}
begin
   open_out_file:=false;
   assign(out_fbuf,filen);
   {$I-}
   reset(out_fbuf);
   {$I+};
   if (ioresult=0) then
    if prompt then
     begin
        close(out_fbuf);
        write('File Exists...OK to erase? (Y/N) ');
        ch :=readkey;
        writeln;
        if NOT (ch='y') and NOT (ch='Y') then
         begin
            writeln('User abort.');
            exit
        end
        else erase(out_fbuf)
   end;
   {$I-}
   rewrite(out_fbuf,rec_size);
   {$I+};
   if ioresult<>0
    then writeln('Cant open file  ['+filen+']')
   else open_out_file:=true
end;

function open_in_file(filen:string;rec_size:word):boolean;
var
  I:integer;
begin
   assign(in_fbuf,filen);
   {$I-}
   reset(in_fbuf,rec_size);
   I:=ioresult;
   {$I+};
   if I<>0 then
    begin
       writeln('Cant find file :'+filen);
       open_in_file:=false;
       readln
    end
    else open_in_file:=true
end;

function adj_filen(filevar:string;indx1,indx2:tenindx):string;
{strips extension, adds new one with type and appended indices}
var
  s1,s2:string;
  fdir:dirstr;
  fname:namestr;
  fext:extstr;
begin
   fsplit(filevar,fdir,fname,fext);
   {dump extension
    add type suffix to extension}
   s1:=fdir+fname+'.M';
   str(indx1,s2); {convert index to ascii, append}
   s1:=s1+s2;
   str(indx2,s2);
   adj_filen:=s1+s2;
   o_filen:=s1+s2;
   wr_str('File : '+o_filen)
end;

function wr_fhead(filen:string):boolean;
{generates file header... 1st line in file }
var
  n1,n2:integer;
  i1:word;
begin
    {prime header...excess items not in STOREV
     but in header are un-initialised}
    i1:=ioresult;
    blockwrite(out_fbuf,STOREV,1); {rec_size=1 line}
    {$I+}
    if (i1<>0) or (ioresult<> 0) then
    begin
       write('File error during header generation, file:- ');
       writeln(filen);
       wr_fhead:=false
    end
    else wr_fhead:=true
end;

procedure do_line;
{fetches data from ram, loads to line records}
var
  port_indx,pix_indx,port_val:integer;
begin
   with LN_BUF[DISPV.ln_indx] do
   begin
      ln_dig:=DISPV.ln_abs;
      {fetch set of segs from basep+port_indx-1}
      for port_indx:=1 to STOREV.segp_ln do
      begin
         {recover start addr for next seg}
 un_store_if_addr;
     port_val:=basep+port_indx-1;
         for pix_indx:=1 to STOREV.pixp_seg do
         begin
            {loop, fetching pix data,incrementing i/f address}
    set_if_addr;
    seg[port_indx].pix[pix_indx]:=port[port_val];
            bump_if_addr
```

```
        end;
        for pix_indx:=1 to STOREV.refp_seg do
begin
           {loop, similarly for ref. element data}
           set_if_addr;
           seg[port_indx].ref[pix_indx]:=port[port_val];
           bump_if_addr
        end
      end
   end;
   store_if_addr {store last addr for next cycle}
end;

function do_buf(filen:string):boolean;
{fills a buffer with lines, writes it out}
var
  y:byte;
  n1:integer;
begin
   do_buf:=false;
   with STOREV do
   begin
      y:=wherey;
      {fetch line of segs to buffer}
      for DISPV.ln_indx:=1 to lnper_buf do
      begin
 gotoxy(12,y);
 write(DISPV.ln_abs:4);
 do_line;
         if keypressed then exit;
         inc(DISPV.ln_num);
         inc(DISPV.ln_abs)
      end
   end;
   {write out buffer load of line records}
   {$I-}
   for n1:=1 to qtyset do
     blockwrite(out_fbuf,ln_buf[n1],1); {rec_size is 1 line}
   {$I+}
   if ioresult <>0 then
   begin
      writeln;
      writeln('Problem writing out buffer.');
      do_buf:=false
   end
   else do_buf:=true
end;
```

```
function do_file(filen:string):boolean;
{writes out several buffers of data, closes file}
begin
   do_file:=false;
   write(filen,' : ');
   {last buffer in last file has less lines}
   with STOREV do
     if (total_lns-DISPV.ln_abs<lnper_buf) then
      lnper_buf:=total_lns-DISPV.ln_abs;
   {write file header}
   if NOT wr_fhead(filen) then exit;
   with STOREV do
     begin
        set_clk(0); {prepare for data transfer}
        DISPV.ln_num:=1;
        for DISPV.buf_indx:=1 to bufp_file do
        begin
           {get buffer of lines, write out}
           if NOT do_buf(filen) then
            begin
               writeln;
               write('Closing file.');
               close(out_fbuf);
               restore_clk;
               if keypressed then ch:=readkey;
            exit
           end
        end;              {got all buffers}
        close(out_fbuf); {close file}
        do_file:=true
   end;  {with STOREV}
   restore_clk;
   writeln
end;

function do_rack(filen:string;rack_indx:tenindx):boolean;
{opens file, builds set of files for single rack}
var
  recsize:word;
  file_indx:tenindx;
begin
   do_rack:=false;
   for file_indx:=1 to STOREV.filep_rack do
   begin
      {adjust file ext for new file}
      filen:=adj_filen(filen,rack_indx,file_indx);
      if NOT open_out_file(filen,STOREV.rec_size) then exit;
      if NOT do_file(filen) then  exit;
      prompt:=false
   end;
   do_rack:=true;
   writeln
end;

procedure init_storev;
begin
   with STOREV do
   begin
      vers:=10; {software version 1.0}
      sv_mode:=1;
      segp_ln:=8;
      lnper_buf:=qtyset;
      bufp_file:=5;
      filep_rack:=4;
      total_lns:=round(131072/(pixp_seg+refp_seg));
      if total_lns>1000 then total_lns:=1000;
      rec_size:=2+(pixp_seg+refp_seg)*segp_ln;
      bufsize:=sizeof(LN_BUF);
   end
end;

{main block store routine}
{suggested mods for later:-
 store mode flag:
     true -  single segment store:
             form segment at port[num] in [rack]
     false - array store:
             for [num] racks,starting at [rack]
storage format is 1 line header,then 1000/ 250
lines in 5 blocks per file, 1st 4 bytes in each
line = rack,segment,buf line count}

{sv_mode :- 01   line number,pixels,dark ref
            02   line number,pixels
            03   line number,pixels-ref
 other numbers used to indicated processed data files.}

procedure blk_store;
var
  n:longint;
  start_rack,in_n:integer;
  rack_indx,file_indx:tenindx;
begin
   clrscr;
   write('Type rack number to store or "a" for all :');
   readln(ch);
```

```
if (ch='a') or (ch='A') then in_n:=3 else in_n:=pos(ch,'123');
if in_n<1 then in_n:=1;
start_rack:=in_n;
with storev do
 begin
    rackp_sys:=1;
    if (in_n<1) or (in_n>3) then
     begin
        start_rack:=1;
        rackp_sys:=3
    end
end;
g_flag:=false;
writeln('Enter filename (no extension required).');
{get filen}
out_filen:=get_filen(o_filen);
if out_filen = '' then exit;
{adjust file extension}
file_indx:=1;
rack_indx:=start_rack;
out_filen:=adj_filen(out_filen,rack_indx,file_indx);
o_filen:=out_filen;
init_storev; {initialise store variables}
writeln('[Any key]-->stop.  Minimum valid line count is 50.');
writeln;
with STOREV do
 begin
    n:=1;
    n:=(n*rec_size*lnper_buf);
    n:=n*(bufp_file)+rec_size;
    n:=n*filep_rack*rackp_sys; {total disc space required }
end;
{check disk space}
if NOT check_dsk(n) then exit;
{init i/f}
sifm_ext_addr; {set i/f for external addressing}
sifm_ext_axs;  {set i/f for external data access}
writeln('Transferring....');
prompt:=true; {turn on query overwrite for 1st file}
DISPV.ln_num:=1;
DISPV.ln_abs:=1;
for rack_indx:=start_rack to (start_rack+STOREV.rackp_sys) do
 begin
    s_addhi:=0;
    s_addmid:=0;
    s_addlo:=0;
    store_if_addr;
    set_rack(rack_indx);  {select rack to access}
```

```
        {get set of files for rack}
        if NOT do_rack(out_filen,rack_indx) then exit;
    end;  {get rack data loop}
    writeln('...done')
end;

procedure get_text1;
{label FILE display axis}
var
  text_inf:textsettingstype;
  s1:string;
  n1:real;
begin
   with DISPV do
    begin
       gettextsettings(text_inf);
       setcolor(c_white);
       outtextxy(300,maxy-60,chr(27)+' Segment(s) '+chr(26)) ;
       outtextxy(500,maxy-63,'File:-'+o_filen);
       n1:=1;
       with STOREV do
         n1:=n1*total_lns*rec_size / 1000;
       str(n1:5:2,s1);
       str(start_seg,s1);
       outtextxy(orgx+12,orgy+7,s1+' '+chr(26));
       outtextxy(525,maxy-110,'Line Number');
       settextstyle(text_inf.font,vertdir,text_inf.charsize);
       outtextxy(5,25,'('+chr(27)+' Light)     Sensor Response  ');
       settextstyle(text_inf.font,text_inf.direction,text_inf.charsize);
       setcolor(o_color)
    end
end;

procedure make_axis;
{draw FILE display axis}
var
  x,y,x1,y1,x2,y2,f1,f2,f3,f4,pixp_ln:integer;
begin
   with DISPV do
   begin
      setviewport(0,0,maxx,maxy,clipoff);
      setviewport(orgx,20,maxx,maxy-21,clipon);
      clearviewport;
      setcolor(c_white);
      get_text1;
      with STOREV do
         x:=(pixp_seg+refp_seg)*x_inc;  {line span for trace 1 full seg}
      f1:=x div qty_seg+1;
      f2:=0;
      f3:=64*y_inc;  {data range on ega display =256 pixels}
      f4:=128*y_inc;
      {scale ticks}
      {plotting pixels and refs to start with....}
      with STOREV do pixp_ln:=(pixp_seg+refp_seg)*qty_seg;
      x:=DISPV.orgx+(pixp_ln*x_inc div pix_inc)+x_inc;
      y:=orgy+y_inc;
      color:=c_white;   {drawline color}
      {draw axis with ticks}
      x1:=orgx-x_inc;
      y1:=y;
      x2:=x;
      y2:=y1;
 {x}  drawline(x1,y1,x2,y2,f1,f2,0);
      x1:=x+1;
      x2:=x+x_inc+(qtylines*x_inc);
      y2:=orgy-(qtylines*y_inc);
      f1:=10*x_inc;
 {y}  drawline(x1,y1,x2,y2,0,f1,0);
      x1:=orgx-x_inc;
      y1:=orgy;
      x2:=orgx-X_inc;
      y2:=y1-255;
 {z}  drawline(x1,y1,x2,y2,f3,f4,4)
   end;
   setcolor(o_color);
   color:=o_color
end;

function get_hdr(filen:string):boolean;
{open file in 1st instance with rec_size=1,
load header data into STOREV, check version number.
Verify line buffer is correct size to load line records.}
begin
   get_hdr:=false;
   if NOT open_in_file(filen,1) then exit;
   {read in storage data from file header}
   {$I-}
   blockread(in_fbuf,STOREV,sizeof(STOREV));
   {$I+}
   if ioresult<>0 then
    begin
       wr_str('Problem reading header from file :-');
       close(in_fbuf);
       exit
   end;
   {check buffer size matches original.. blockreads
   using multiple line recs}

   if (STOREV.bufsize <> sizeof(LN_BUF)) then
     begin
       wr_str('Buffer size missmatch..wrong s/w version or corrupt data.');
       exit
    end;

    {check version number}
    if (STOREV.vers <> vers_ok) then
      begin
         wr_str('Stored using wrong version of software.');
         wr_str('Attempt display (Y/N) ? ');
         ch:=readkey;
         if NOT (ch in ['y','Y']) then
         begin
            close(in_fbuf);
            exit
         end
    end;
    get_hdr:=true;
    close(in_fbuf)
end;

function rd_buf(lncount:integer):word;
var
n1,n2:integer;
begin
    {$I-}
    for n1:=1 to lncount do
     blockread(in_fbuf,LN_BUF[n1],1);
    {$I+}
    rd_buf:=ioresult
end;

function mov_updn:boolean;
{move up or down a file in data set}
var
  i1,i2:word;
  n1,n2,n3:integer;
begin
   with DISPV do
    begin
       n1:=lnper_file;
       n2:=n1*2;
       n3:=n1*3;
       if ln_num <n1 then file_indx:=1
          else if ln_num <n2 then file_indx:=2
```

```
            else if ln_num <n3 then file_indx:=3
              else file_indx:=4;
        maxlnf:=lnper_file*file_indx;
        minlnf:=maxlnf-STOREV.lnper_buf*STOREV.bufp_file+1;
        minln:=minlnf;   {set to bottom buffer in file}
        maxln:=minln+STOREV.lnper_buf-1
      end;    {with DISPV}
      {adjust filename for next file, file_indx incremented}
      in_filen:=adj_filen(in_filen,DISPV.rack_indx,DISPV.file_indx);
      {read in new STOREV values}
      if NOT get_hdr(in_filen) then exit;
      {read in file}
      if NOT open_in_file(in_filen,STOREV.rec_size) then exit;
      {$I-}
      {step past header}
      seek(in_fbuf,1);
      i1:=ioresult;      {load in bottom buffer for file}
      {$I+}
      i2:=rd_buf(qtyset);
      if (i1<>0) or (i2<>0) then
      begin
        close(in_fbuf);
        wr_str('Problem moving down-file, filename.');
        closegraph;
        g_flag:=false;
        exit
      end;
      mov_updn:=true
   end;

   function check_ln:boolean;
   {checks if line is in current buffer, if not loads
    required buffer, switching files in the data set if necessary}

   label at_bufend;
   var
     i1,i2:word;
     n1,n2:integer;
     s1:string;
   begin
      with DISPV do
       begin
         check_ln:=true;
         {quick exit if line in buffer}
         if NOT (ln_num <minln) and (ln_num <= maxln) then exit;
         {end of buffer? use mov_up to handle}
         if (ln_num=STOREV.total_lns+1) then goto at_bufend;
         {seek and load if in current file}
        if (ln_num >=minlnf) and (ln_num <= maxlnf) then
         begin
            check_ln:=false;
    ln_indx:=ln_num-maxln;   {offset into buffer}
            {$I-}
            n1:=(ln_num-minlnf+1);   {index into file}
    seek(in_fbuf,n1);        {rec 0 is header}
            i1:=ioresult;
            {handle case of less than buffer full of lines}
            with STOREV do
             begin
                n2:=lnper_buf*(bufp_file-1); {work out residual}
                if n1>n2 then n2:=lnper_file-n1 {n1 is index into file}
                else n2:=lnper_buf
            end;
            {$I+}
            i2:=rd_buf(n2);
            if (i1<>0) or (i2<>0) then
             begin
                wr_str('Cant find line in file.. <CR to exit.');
                readln;
                closegraph;
                g_flag:=false;
                exit
            end;
            {update limits for new file..min,maxln updated
             for any buffer changes, min,maxlnf updated for
             file changes};
            DISPV.minln:=DISPV.ln_num;
            DISPV.maxln:=DISPV.ln_num+qtylines-1;
            check_ln:=true;
            exit
        end; {if ln_num proc}
        {otherwise load next file up/down}
        at_bufend:
        check_ln:=false;
        close(in_fbuf);
        if NOT mov_updn then exit;
         {use recursion to seek correct buffer in file}
        if NOT check_ln then
         wr_str('Failed to find required line')
    end;  {with DISPV do}
    check_ln:=true
end;

procedure draw_span;
{draw FILE display data span bar}
var
```

```
 vp:viewporttype;
 n1,n2,n3:integer;
 s1:string;
begin
  with DISPV do
   begin
     getviewsettings(vp);
     setviewport(0,0,maxx,maxy,true);
     setviewport(30,maxy-10,maxx-100,maxy,clipoff);
     clearviewport;
     setcolor(c_white);
     {400 pixels for approx screen span required}
     n1:=STOREV.total_lns div 400;
     n2:=STOREV.total_lns div n1;   {scale down}
     rectangle(5,2,5+n2,4);
     setcolor(c_green);
     n2:=ln_num div n1;
     {cursor}
     line(5+n2,  0, 5+n2+2, 2);
     line(5+n2+2,2, 5+n2+4, 0);
     line(5+n2+4,0, 5+n2, 0);
     {location and span};
     n3:=qtylines*ln_inc;
     n2:=oldpos+n3;
     if (n2>=STOREV.total_lns) then oldpos:=STOREV.total_lns-n3;
     rectangle(5+(oldpos div n1),5,(5+((oldpos+n3) div n1)),7);
     setcolor(c_white);
     setviewport(vp.x1,vp.y1,vp.x2,vp.y2,vp.clip)
   end;
   setcolor(o_color)
end;

procedure draw_seg;
{draw FILE display segment size/position bar}
var
  vp:viewporttype;
  n1,n2,n3:integer;
begin
   with DISPV do
   begin
      getviewsettings(vp);
      setviewport(0,0,maxx,maxy,true);
      setviewport(30,maxy-20,maxx-300,maxy-11,clipoff);
      clearviewport;
      setcolor(c_white);
      for n1:=1 to 8 do
        rectangle(5+((n1-1)*20),0,25+((n1-1)*20),4);
      setcolor(c_green);
      n1:=(DISPV.start_seg-1)*19;
      n2:=(DISPV.start_seg-1)*1;
      n3:=(DISPV.qty_seg-1)*20;
      rectangle(5+n2+n1,1,24+n2+n1+n3,3);
      setcolor(c_white);
      setviewport(vp.x1,vp.y1,vp.x2,vp.y2,vp.clip)
   end;
   setcolor(o_color)
end;

procedure draw_dig(num,colr:integer);
{put some digits in a box for FILE display}
var
  vp:viewporttype;
  n1,n2:integer;
  s1:string;
begin
   with DISPV do
    begin
       getviewsettings(vp);
       setviewport(0,0,maxx,maxy,true);
       setviewport(maxx-90,maxy-12,maxx-40,maxy,clipoff);
       clearviewport;
       setcolor(colr);
       rectangle(0,0,50,10);
       str(num:4,s1);
       outtextxy(15,2,s1);
       setviewport(vp.x1,vp.y1,vp.x2,vp.y2,vp.clip)
   end;
   setcolor(o_color)
end;

function plot_ln:boolean;
{plots line trace for FILE display}
var
  pix_indx,ref_indx,first_pix,
  adj_pix, x,y,o_x,o_y,data:integer;
  s1:string;
  segno,n1:integer;
begin
   with DISPV do
    begin
       plot_ln:=false;
       {check line in current buffer ,fetch if required}
       if NOT check_ln then exit;
       ln_indx:=ln_num-minln+1;
       with LN_BUF[ln_indx] do
        begin
           ln_dig:=ln_num;
        draw_dig(ln_dig,green);
        segno:=start_seg;
data:=seg[segno].pix[1];
o_y:=refy-data;    {make 1st'last value' dummy}
        o_x:=refx;
        x:=refx;
        {dump first pixel for particular clock
         frequencies--- hardware anomaly}
        if storev.pix_clk=1 then
         begin
            first_pix:=1;
            adj_pix:=1;
        end
        else begin
                first_pix:=2;
                adj_pix:=0;
        end;
        for n1:=1 to qty_seg do
         begin
            pix_indx:=first_pix;
            data:=seg[segno].pix[pix_indx];
            o_y:=refy-data;
            o_x:=x;
            {plot 1 segment of pixels}
            repeat
 data:=seg[segno].pix[pix_indx];
                 y:=refy-data;
                 if hide_flag then drawline(o_x,o_y,x,y,0,0,0)
                  else line(o_x,o_y,x,y);
                 o_y:=y;
                 o_x:=x;
                 inc(x,x_inc);
                 inc(pix_indx,pix_inc);
            until pix_indx> STOREV.pixp_seg-adj_pix;
            {may want to plot ref pixels}
            {inhibit for now
            ref_indx:=1;
            repeat
                 data:=seg[segno].ref[ref_indx];
                 y:=refy-data;
                 if hide_flag then drawline(o_x,o_y,x,y,0,0,0)
                 else line(o_x,o_y,x,y);
                 o_y:=y;
                 o_x:=x;
                 inc(x,x_inc);
                 inc(ref_indx,pix_inc);
            until ref_indx> STOREV.refp_seg;}
            inc(segno)
```

```
          end {end of segs loop}
       end;  {end of with ln_buf do }
       plot_ln:=true
   end {with DISPV do}
end;

procedure plot_ln_debug(pixone:integer;plotref:boolean);
{plots line trace for DEBUG display mode}
var
  pix_indx,ref_indx,
  x,y,o_x,o_y,data:integer;
  s1:string;
  segno,n1:integer;
begin
   with DISPV do
    begin
      ln_indx:=ln_num-minln+1;
      with LN_BUF[ln_indx] do
       begin
          ln_dig:=ln_num;
          draw_dig(ln_dig,green);
          segno:=start_seg;
          data:=seg[segno].pix[1];
          o_y:=refy-data;    {make 1st'last value' dummy}
          o_x:=refx;
          x:=refx;
          for n1:=1 to qty_seg do
           begin
              {plot 1 segment of pixels}
              pix_indx:=pixone;
              repeat
                   data:=seg[segno].pix[pix_indx];
                   y:=refy-data;
                   line(o_x,o_y,x,o_y);
                   line(x,o_y,x,y);
                   o_y:=y;
                   o_x:=x;
                   inc(x,x_inc);
                   inc(pix_indx,pix_inc);
             until (pix_indx> STOREV.pixp_seg)
                  or (x>orgx-1+boxwidth*3);
             {plot ref pixels as well}
             ref_indx:=1;
             if plotref then
              begin
                 repeat
                      data:=seg[segno].ref[ref_indx];
                      y:=refy-data;
```

```
                        line(o_x,o_y,x,o_y);
                        line(x,o_y,x,y);
                        o_y:=y;
                        o_x:=x;
                        inc(x,x_inc);
                        inc(ref_indx,pix_inc);
                   until (ref_indx> STOREV.refp_seg)
                      or (x>orgx-1+boxwidth*3);
               end;
               inc(segno);
            end        {end of segs loop}
        end;        {end of with ln_buf do }
    end {with DISPV do}
end;

function plot_ntrace:boolean;
{plot a group of traces for FILE display}
var
  n1,n2:integer;
  s1:string;
begin
   with DISPV do
    begin
       plot_ntrace:=false;
       refx:=orgx;
       refy:=orgy;
       setcolor(c_green);
       n1:=1;
       {check line in current buffer ,fetch if required}
       if (ln_num>STOREV.total_lns-(qtylines*ln_inc)) then
        begin
           wr_str('Top of buffer');
           ln_num:=STOREV.total_lns-(qtylines*ln_inc);
           if ln_num<1 then ln_num:=1
       end;
       if NOT check_ln then exit;
       {label line axis start/end}
       setcolor(c_white);
       str(ln_num:4,s1);
       outtextxy(430,orgy,s1);
       str((ln_num-1+qtylines*ln_inc):4,s1);
       outtextxy(580,170,s1);
       if (ln_num<STOREV.total_lns-qtylines*ln_inc) then draw_span;
       repeat
            if NOT plot_ln then exit;
            {initialise arrays, used in hidden line removal}
            for n2:=0 to 639 do o_minpix[n2]:=minpix[n2];
            inc(ln_num,ln_inc);
            inc(refx,x_inc);
            {translate}
            inc(refy,-y_inc);
            inc (n1);
            {stop when display done or no more lines in file}
       until (n1> qtylines)
          or (ln_num>STOREV.total_lns) or keypressed;
       plot_ntrace:=true
   end;  {with DISPV do}
   setcolor(o_color)
end;

function start_graph_mode:boolean;
{initialise graphics}
var
  grdriver,
  grmode,
  errorcode:integer;
begin
   grdriver :=detect;
   grmode:=grmode_val;
   g_flag:=false;  {global flag}
   start_graph_mode:=false;
   initgraph(grdriver,grmode,'c:');
   errorcode:=graphresult;
   if errorcode <> grok then
    begin
       writeln('Graphics error: ',grapherrormsg(errorcode));
       exit
   end;
   g_flag:=true;
   start_graph_mode:=true
end;

procedure setspan(sign:integer);
{adjust span bar for FILE display}
var
  n1:integer;
begin
   with DISPV do
    begin
       n1:=STOREV.total_lns div qtylines;
       if sign <0 then
          begin
             ln_inc:=ln_inc div 2;
             if ln_inc<1 then ln_inc:=1;
             draw_span;
             exit
          end
          else begin  {sign >=0}
             if ((ln_num+ln_inc)>STOREV.total_lns) then exit;
             ln_inc:=ln_inc*2;
             if ln_inc>n1 then ln_inc:=n1;
             draw_span
        end
    end
end;

procedure mvdnln(num:integer);
{adjust variables and check limits for FILE display}
begin
  with DISPV do
  begin
     dec(ln_num,num);
     if ln_num<1 then ln_num:=1;
     draw_span;
     draw_dig(ln_num,white)
  end
end;

procedure mvupln(num:integer);
{adjust line variables and check limits for FILE display}
begin
   with DISPV do
   begin
      inc(ln_num,num);
      if ln_num>STOREV.total_lns
       then ln_num:=STOREV.total_lns-1;
      draw_span;
      draw_dig(ln_num,white)
   end
end;

procedure zoomup;
{adjust segment variables and check
limits for FILE display}
var
s1:string;
begin
  with DISPV do
  begin
   if pix_inc <8 then
   begin
      inc(pix_inc);
      inc(qty_seg);
      if (start_seg>(qty_seg-pix_inc))
```

```
       then start_seg:=qty_seg-pix_inc+1;
      draw_seg
   end
 end
end;

procedure zoomdn;
{adjust segment variables and check
limits for FILE display}
var
s1:string;
begin
  with DISPV do
  begin
    if pix_inc>1 then
    begin
       dec(pix_inc);
       dec(qty_seg);
       draw_seg
    end
  end
end;

procedure mvzmrt;
{adjust segment variables and check
limits for FILE display}
var
s1:string;
begin
   with DISPV do
   begin
    if start_seg>1 then
    begin
       dec(start_seg);
       draw_seg
    end
  end
end;

procedure mvzmlft;
{adjust segment variables and check
limits for FILE display}
var
s1:string;
begin
   with DISPV do
   begin
    if start_seg<8 then
```

```
      if (start_seg<STOREV.segp_ln-qty_seg+1) then
      begin
         inc(start_seg);
         draw_seg
      end
    end
 end;

 procedure init_dispv;
 {initialise display variables, used ONLY
 when in graphics mode}
 begin
    with DISPV do
     begin
        maxx:=getmaxx;
        maxy:=getmaxy;
        orgx:=20;
        orgy:=maxy-70;
        x_inc:=3;
        y_inc:=2;
        cmdx1:=orgx;
        cmdy1:=0;
        cmdx2:=maxx-15;
        cmdy2:=15;
        txtx:=38;
        txty:=4;
        ln_num:=1+(STOREV.lnper_buf*STOREV.bufp_file*(file_indx-1));
        ln_indx:=1;
        qtylines:=qtyset;
        max_trace:=420;
        pix_inc:=8;
        ln_inc:=1;
        seg_indx:=1;
        buf_indx:=1;
        start_seg:=1;
        lnper_file:=STOREV.lnper_buf*STOREV.bufp_file;
        minlnf:=1+(lnper_file*(file_indx-1));
        maxlnf:=minlnf+lnper_file-1;
        minln:=minlnf;
        maxln:=minln+STOREV.lnper_buf-1;
        hide_flag:=true;
        start_seg:=1;
        qty_seg:=STOREV.segp_ln
    end
 end;

 procedure blk_screen;
 {display file contents for FILE display}
```

```
label getfloop;
var
  ch:char;
  i1,i2:word;
  n1,n2:integer;
  fdir:dirstr;
  fname:namestr;
  fext:extstr;
  s1,s2,s3:string;
  r1:real;
begin
    clrscr;
    getfloop:
    {open file WITH user provide extension}
    writeln('Run-set/ file name to load, ( extension optional ).');
    in_filen:=get_filen(o_filen);
    if in_filen =''' then exit;
    {determine rack & file number from file extension}
    fsplit(in_filen,fdir,fname,fext);
    if (fext='') or NOT (length(fext)=4) then
     begin
        DISPV.rack_indx:=1;
        DISPV.file_indx:=1
    end
    else begin
            s1:=copy(fext,3,1);
            val(s1,n1,n2);
            if n2 <>0 then goto getfloop; {get file loop}
            DISPV.rack_indx:=n1;
            s1:=copy(fext,4,1);
            val(s1,n1,n2);
            if n2 <>0 then goto getfloop;
            DISPV.file_indx:=n1
    end;
    in_filen:=adj_filen(in_filen,DISPV.rack_indx,DISPV.file_indx);
    {get file header}
    if NOT get_hdr(in_filen) then exit;
    with STOREV do
     begin
       writeln;
       writeln('Information extracted from file:-');
       writeln;
       write('Data generated with a sampling frequency of ');
       r1:=12000/(exp(ln(2)*(STOREV.pix_clk+1)));
       writeln(r1:4:2,' KHz');
       writeln(pixp_seg:4,' pixels per segment.');
       writeln(refp_seg:4,' ref-samples per segment.');
  writeln(lnper_buf:4,' lines per buffer.');
  writeln(bufp_file:4,' buffers per file.');
  writeln(filep_rack:4,' files per rack.');
  writeln(rackp_sys:4,' rack(s) in system.');
  write('Total number of integral lines :- ',total_lns,'.');
  case STOREV.sv_mode of
     1:s1:='raw data.';
     2:s1:='raw pixels only.';
     3:s1:='pixels minus averaged ref. samples.';
     4:s1:='processed, ( smoothed, compensated pixel data ).'
     else s1:='non specific.'
  end;
  writeln(' Storage format :- ',s1);
  readln;
end;  {with storev}
{re-open file with line/record}
if NOT open_in_file(in_filen,STOREV.rec_size) then exit;
{seek past header}
{$I-}
seek(in_fbuf,1);
i1:=ioresult;
{$I+}
i2:=rd_buf(qtyset);
if (i1<>0) or (i2<>0) then
begin
   writeln('Problem seeking 1st line in filename :- ',in_filen);
   close(in_fbuf);
   closegraph;
   g_flag:=false;
   readln;
   exit
end;
{set graphics}
if NOT start_graph_mode then
begin
   close(in_fbuf);
   exit
end;
{init display vars}
init_dispv;
o_color:=c_green;
{ready for hidden line proc}
{draw instruction box}
with DISPV do
begin
   rectangle(cmdx1,cmdy1,cmdx2,cmdy2);
   s1:='['+chr(30)+chr(31)+'] lines,';
   s2:='['+chr(17)+chr(16)+'] shift,';
```

```
        s3:='[+/-] zoom, [<>] seg, [h] hide [s] show, [q] quit.';
        outtextxy(txtx,txty,s1+s2+s3);
        {outer loop: plot num_lns of trace}
repeat
             for n1:=0 to 639 do
              begin
                 o_minpix[n1]:=DISPV.orgy;
                 minpix[n1]:=DISPV.orgy
             end;
             oldpos:=ln_num; {save start for draw_span etc}
             draw_seg;
             make_axis;
     if NOT plot_ntrace then
              begin
                 closegraph;
                 exit
             end;
     draw_span;
             with DISPV do
              begin
                 repeat
                      ch:=readkey;
                      if ch=#0 then ch:=readkey;
        case (ch) of
        #80:setspan(-1);      {decrease jump}
        #72:setspan(1);       {increase jump}
        #75:mvdnln(1);        {jump back}
        #77:mvupln(1);        {jump forward}
        #73:mvupln(25);
        #81:begin mvdnln(25);ch:=#1;end;
        {ch=81 is also 'Q'}
        #71:DISPV.ln_num:=1;
        #79:DISPV.ln_num:=STOREV.total_lns-STOREV.lnper_buf*DISPV.ln_inc;
        'h','H':begin hide_flag:=true;wr_str('Hide lines.');end;
        's','S':begin hide_flag:=false;wr_str('Show lines.');end;
        '+':zoomdn;          {decrease zoom}
        '-':zoomup;          {increase zoom}
        '<':mvzmrt;          {move zoom zone 'left'}
        '>':mvzmlft;         {move zoom zone 'right'}
        '1':qtylines:=1;
        '2':qtylines:=20;
        '3':qtylines:=30;
        '4':qtylines:=40;
        '5':qtylines:=qtyset;

                      end;
                      {replot on carriage return}
                 until (ch in [#13,'q','Q']);
              end; {with DISPV}
           until (ch in ['q','Q'])
        end; {with DISPV do}
        close(in_fbuf);
        closegraph;
        g_flag:=false
end;

function quiet_sample:boolean;
{start acquire cycle, check to see if
 i/f responds}
var
  n1:integer;
begin
   quiet_sample:=false;
   {arm i/f}
   new_stat:=v_arm;
   s_bit:=b_arm;
   pulse_io;
   if NOT enable1 then
   begin
      wr_str('HST i/f not responding to arm signal.');
      exit
   end;
   {clear i/f counter for correct terminal count}
   new_stat:=v_clear1;
   s_bit:=b_clear;
   pulse_io;
   {start sampling}
   new_stat:=v_start;
   s_bit:=b_start;
   pulse_io;
   {wait for done flag}
   quiet_sample:=true;
   setcolor(c_green);
   outtextxy(8,40,'Sampling...');
   {wait in loop for acquire cycle to clear}
   repeat until (NOT enable1) or keypressed;
   if enable1 then {must be keypressed}
   begin
      wr_str('HST i/f sample-done signal not detected.');
      quiet_sample:=false
   end;
   outtextxy(100,40,'..done');
   setcolor(o_color)
end;

procedure set_pat;
```

```
{set a ramp pattern in selected rack/seg ram
 for DEBUG display test}
var
  pix_indx,ref_indx:integer;
begin
   with STOREV do
    begin
       with DISPV do
        begin
           for ln_indx:=1 to 10 do
            begin
               with LN_BUF[ln_indx] do
                begin
                   ln_dig:=ln_indx;
                   for pix_indx:=1 to pixp_seg do
                     seg[start_seg].pix[pix_indx]:=(pix_indx-1)*2;
                  for ref_indx:=1 to refp_seg do
                    seg[start_seg].ref[ref_indx]:=(ref_indx-1)*8;
               end    {with LN_BUF}
           end     {ln_indx loop}
       end          {with DISPV}
   end;             {with STOREV}
end;


procedure init_curs;
{preserve graphics under curser, place cursor}
var
  n1,n2,n3:integer;
begin
  with DISPV do
   begin
      n1:=cursx*x_inc;
      getimage(n1,0,n1+1,255,cursxptr^);
      n2:=cursy;
      n3:=(pixset+refset)*x_inc;
      getimage(0,n2,n3,n2+1,cursyptr^);
      setcolor(c_green);
      line(n1,0,n1,255);
      line(0,n2,n3,n2)
   end;
   setcolor(o_color)
end;

procedure draw_debug;
{DEBUG display}
var
  x1,y1,x2,y2,ix,iy:integer;
```

```
   s1,s2:string;
   n1,n2:integer;
 begin
    with DISPV do
     begin
        with STOREV do
         begin
            {command box as in blk_screen}
            setviewport(0,0,getmaxx,getmaxy,clipoff);
            setcolor(c_white);
            n1:=(pixset+refset)*x_inc;  {ideal 128 pixels+6 ref}
            n1:=boxwidth*x_inc;
            {force to wide view to allow some adjustment to pix frame}
            x1:=cmdx1;y1:=cmdy1+20;
            x2:=cmdx1+n1;
            y2:=cmdy1+275;
            {instruction box}
            ix:=x1;iy:=y2+5;
            rectangle(ix,iy,x2+160,y2+50);
    s1:='(P)ixels,    ref.(E)elements,   (H)ardware frame,  pixel (C)lock ';
    s2:='(R)ack number,    (S)eg number,  (1 to 5) lines displayed';
           ix:=ix+10;
           outtextxy(ix,iy+4,s1);
           outtextxy(ix,iy+14,s2);
    s1:='(A)cquire samples, (F)etch data,  (T)est, (Z)ap ,single sample';
          outtextxy(ix,iy+24,s1);
    s1:='('+chr(24)+chr(25)+chr(27)+chr(26)+')';
    s2:='(G)et adjacent, (I)nterface reset,'+s1+' cursor, (+/-) curs.sens';
          outtextxy(ix,iy+34,s2);
          {numbers box}
          setviewport(x2+10,y1,x2+10+200,y2,clipoff);
          rectangle(0,0,150,255);
          {wave form box}
          setviewport(x1,y1,x2,y2,clipoff);
          clearviewport;
          rectangle(0,0,n1,255); {elements in segment,data resolution}
          orgx:=0;refx:=0;orgy:=255;refy:=255;
          {end of line marker}
          setcolor(c_red);
          n1:=(STOREV.pixp_seg)*x_inc;
          line(n1,1,n1,254)
        end
    end;
    setcolor(o_color);
 end;

 procedure draw_curs;
 {cursor for DEBUG display}
```

```
var
 vp:viewporttype;
 n1,x1,y1,x2,y2:integer;
 n2:real;
 s1:string;
begin
  with DISPV do
  begin
     with STOREV do
       x1:=cmdx1+18+boxwidth*x_inc;
     y1:=cmdy1+245;
     x2:=x1+135;
     y2:=y1+25;
     getviewsettings(vp);
     setviewport(0,0,getmaxx,getmaxy,true);
     setviewport(x1,y1,x2,y2,clipoff);
     clearviewport;
     setcolor(c_white);
     rectangle(0,0,135,25);
     outtextxy(10,4,'Signal:-');
     with STOREV do
       if cursx<= pixp_seg then s1:='Pixel :-'
         else if cursx<=pixp_seg+refp_seg then  s1:='Ref.  :-';
     outtextxy(10,15,s1);
     setcolor(c_green);
     with LN_BUF[ln_indx] do
     begin
      if (cursx <= STOREV.pixp_seg) then n1:=seg[start_seg].pix[cursx]
      else  n1:=seg[start_seg].ref[cursx-STOREV.pixp_seg]
     end;
     n2:=2/ 256; {2 volts fscale}
     n2:=n2*n1;
     str(n2:4:2,s1);
     outtextxy(80,4,s1+' V');
     n1:=cursx;
     with STOREV do
       if cursx>pixp_seg then
        if cursx<=pixp_seg+refp_seg then n1:=cursx-pixp_seg;
     str(n1:4,s1);
     outtextxy(80,15,s1);
     setviewport(vp.x1,vp.y1,vp.x2,vp.y2,vp.clip)
  end;
  setcolor(o_color)
end;

procedure draw_vars;
{draw DEBUG display variables}
var
```

```
  vp:viewporttype;
  n1,x1,y1,x2,y2:integer;
  n2:real;
  s1:string;
begin
   with DISPV do
    begin
       with STOREV do
         x1:=cmdx1+18+boxwidth*x_inc;
       y1:=cmdy1+25;
       x2:=x1+135;
       y2:=y1+215;
       getviewsettings(vp);
       setviewport(0,0,getmaxx,getmaxy,true);
       setviewport(x1,y1,x2,y2,clipoff);
       clearviewport;
       setcolor(c_white);
       rectangle(0,0,135,215);
       n2:=12000/(exp(ln(2)*(STOREV.pix_clk+1)));
       str(n2:4:2,s1);
       outtextxy(10,4,'Pixel clock :- ');
       outtextxy(10,14,s1+'KHz.' );
       str(STOREV.pixp_seg:4,s1);
       outtextxy(10,34,'Pixels :- '+s1);
       str(STOREV.refp_seg:4,s1);
       outtextxy(10,44,'Refels :- '+s1);
       str(DISPV.rack_indx:4,s1);
       outtextxy(10,54,'Rack :- '+s1);
       str(DISPV.start_seg:4,s1);
       outtextxy(10,64,'Seg :- '+s1);
       str((DISPV.frame+128),s1);
       outtextxy(10,74,'Frame :- '+s1);
       with DISPV do
         if u_mode then s1:='Single sample.'
          else s1:='Multi-sample.';
       outtextxy(10,94,s1);
       outtextxy(10,114,'Status :-');
       outtextxy(10,124,'   0:');
       outtextxy(10,134,'   1:');
       outtextxy(10,144,'   2:');
       outtextxy(10,154,'   3:');
       outtextxy(10,164,'   4:');
       outtextxy(10,174,'   5:');
       outtextxy(10,184,'   6:');
       outtextxy(10,194,'   7:');
       setviewport(vp.x1,vp.y1,vp.x2,vp.y2,vp.clip)
   end;
   setcolor(o_color)
```

```
end;

procedure mvcurs;
{when either cursor moves, both under image must be
 restored and both lines redrawn}
var
  n1,n2,n3:integer;
begin
   with DISPV do
    begin
       {restore old image under cursor}
       n1:=o_cursx*x_inc;
       putimage(n1,0,cursxptr^,normalput);
       n3:=o_cursy;
       putimage(0,n3,cursyptr^,normalput);
       {get image under new position}
       n1:=cursx*x_inc;
       getimage(n1,0,n1+1,255,cursxptr^);
       n2:=cursy;
       n3:=boxwidth*x_inc;
       getimage(0,n2,n3,n2+1,cursyptr^);
       setcolor(c_green);
       line(n1,0,n1,255);
       line(0,n2,n3,n2);
       o_cursx:=cursx;
       o_cursy:=cursy;
       draw_curs
   end;
   setcolor(o_color)
end;

function fetch_ch(s1,s2,s3:string):char ;
{fetch user response, using box in graphics window}
var
  size:word;
  p:pointer;
  vp:viewporttype;
  x,y:integer;
begin
   getviewsettings(vp);
   x:=DISPV.cmdx1+10;y:=DISPV.cmdy1+40;
   setviewport(0,0,getmaxx,getmaxy,true);
   setviewport(x,y,x+380,y+50,clipoff);
   size:=imagesize(0,0,380,50);
   getmem(p,size);
   getimage(0,0,380,50,p^);
   clearviewport;
   rectangle(0,0,380,50);
   setcolor(c_green);
   outtextxy(8,10,s1);
   outtextxy(8,20,s2);
   outtextxy(8,30,s3);
   fetch_ch:=readkey;
   putimage(0,0,p^,normalput);
   freemem(p,size);
   setviewport(vp.x1,vp.y1,vp.x2,vp.y2,vp.clip);
   setcolor(o_color)
end;

procedure g_rack;
var
  s1,s2,s3:string;
  ch:char;
  n1,n2:integer;
begin
   s2:='Rack number (1 to 3) ? :- ';
   s1:='';s3:='';
   if g_flag then ch:=fetch_ch(s1,s2,s3)
     else begin
             write(s2,' ');
             readln(ch)
   end;
   val(ch,n1,n2);
   if (n1<3) and (n1>0) then
    begin
       dispv.rack_indx:=n1;
       set_rack(n1)
    end
end;


procedure get_21;
{get two adjacent segments for DEBUG display}
var
  n1,pixone:integer;
  last_seg:boolean;
begin
   with DISPV do
    begin
       with STOREV do
        begin
           if start_seg=8
            then last_seg:=true else last_seg:=false;
           if last_seg then
           begin
              start_seg:=7;
              port_num:=basep+6
           end;
           if quiet_sample then
           begin
              fetch_data;
              n1:=1;
              ln_num:=2;
              draw_debug;
              pixone:=64;
              refx:=orgx;
              refy:=orgy;
              repeat
                   setcolor(17-ln_num);
                   o_color:=17-ln_num;
                   color:=ln_num;
                   plot_ln_debug(pixone,false);
                   inc(ln_num,ln_inc);
                   inc(n1);
              until (n1>qtylines)
           end;
           setcolor(c_red);
           line(orgx+64*x_inc,1,orgx+64*x_inc,254);
           setcolor(c_white);
           n1:=1;
           ln_num:=2;
           pixone:=1;
           refx:=orgx+64*x_inc;
           refy:=orgy;
           inc(start_seg);
           port_num:=basep+start_seg-1;
           fetch_data;
           repeat
                setcolor(17-ln_num);
                o_color:=17-ln_num;
                color:=ln_num;
                plot_ln_debug(pixone,false);
                inc(ln_num,ln_inc);
                inc(n1);
           until (n1>qtylines);
           if last_seg then start_seg:=8 else dec(start_seg);
           port_num:=basep+start_seg-1
       end
    end
end;


procedure direct_screen;
{debug version of blk screen: works directly
from acquire ram data}
```

```
var
  ch:char;
  i1:word;
  n1,n2,pixone:integer;
  fdir:dirstr;
  fname:namestr;
  fext:extstr;
  s1,s2,s3:string;
  r1:real;
  bz:byte;
  last_seg:boolean;
begin
   clrscr;
   {set graphics}
   if NOT start_graph_mode then
    begin
       close(in_fbuf);
       exit
    end;
    {clear TEST1 bit-- wait for diode sync}
    bz:=o_modc and $7f;
    port[modc]:=bz;
    o_modc:=bz;
    {grab some memory for cursor}
    {init vars}
    with STOREV do
      begin
         segp_ln:=1;
 total_lns:=5;
 lnper_buf:=5;
 bufp_file:=1;
 filep_rack:=1;
    end;
    with DISPV do
     begin
        maxx:=getmaxx;
        maxy:=getmaxy;
        orgx:=20;
        orgy:=maxy-70;
        refx:=orgx;
        refy:=orgy;
        x_inc:=3;
        y_inc:=10;
        cmdx1:=orgx;
        cmdy1:=0;
        cmdx2:=maxx-15;
        cmdy2:=15;
        txtx:=38;
      txty:=4;
      ln_num:=1;
      ln_indx:=1;
      qtylines:=1;
      max_trace:=420;
      pix_inc:=1;
      ln_inc:=1;
      rack_indx:=1;
      seg_indx:=1;
      buf_indx:=1;
      start_seg:=1;
      minlnf:=1;
      maxlnf:=10;
      minln:=1;
      maxln:=10;
      hide_flag:=false;
      u_mode:=false;
      start_seg:=1;
      qty_seg:=1;
      {outer loop: plot num_lns of trace}
      cursx:=x_inc*5;
      cursy:=128;   {base line approximating black level}
      o_cursx:=cursx;
      o_cursy:=cursy;
      n1:=cursx*x_inc;
      sizex:=imagesize(n1,0,n1+1,255);
      getmem(cursxptr,sizex);
      n1:=cursy;
      n2:=(pixset+refset)*x_inc;
      sizey:=imagesize(0,n1,n2,n1+1);
      getmem(cursyptr,sizey);
      curs_inc:=1;
      o_color:=c_green;
      set_rack(1);
      draw_vars;
      repeat      {outer loop}
           n1:=1;
           ln_num:=2;
           draw_debug;
           pixone:=1;
           refx:=orgx;
           refy:=orgy;
           repeat {inner loop}
                setcolor(16-ln_num);
                o_color:=16-ln_num;
                color:=ln_num;
plot_ln_debug(pixone,true);
                inc(ln_num,ln_inc);
```

```
                 inc(n1);
            until (n1>qtylines);
            init_curs;
            draw_curs;
            with STOREV do
             begin
                repeat
                    ch:=readkey;
                    if ch=#0 then ch:=readkey;

case (ch) of
#75:begin        {left arrow}
       dec(cursx,curs_inc);
       if cursx<1 then cursx:=1;
       mvcurs
       end;
#77:begin        {right arrow}
      inc(cursx,curs_inc);
      with STOREV do
         n1:=(pixp_seg+refp_seg);
      if cursx>n1 then cursx:=n1;
      mvcurs
    end;
#80:begin        {down arrow}
       inc(cursy,curs_inc);
       if cursy>255 then cursy:=255;
       mvcurs
    end;         {up arrow}
#72:begin
       dec(cursy,curs_inc);
       if cursy<1 then cursy:=1;
       mvcurs
    end;
'a','A':if quiet_sample then
      begin draw_vars; fetch_data;ch:=#13;end;
'f','F':begin fetch_data;ch:=#13;end;
't','T':begin set_pat;ch:=#13;end;
'-':if curs_inc>1 then curs_inc:=curs_inc div 2;
'+':if curs_inc<32 then curs_inc:=curs_inc*2;
'1':begin qtylines:=1;draw_vars;end;
'2':begin qtylines:=2;draw_vars;end;
'3':begin qtylines:=3;draw_vars;end;
'4':begin qtylines:=4;draw_vars;end;
'5':begin qtylines:=5;draw_vars;end;
 'r','R':begin g_rack;draw_vars end;
's','S':begin
        s1:='Segment number? :- ';
        s2:='';s3:='';
        ch:=fetch_ch(s1,s2,s3);
        val(ch,n1,n2);
        if (n1>0) and (n1<9) then
          begin
            start_seg:=n1;
            port_num:=basep+n1-1
          end;
        draw_vars
    end;
'c','C':begin
       s1:='0)     6MHz  1)      3MHZ   2)   1.5MHz  ';
       s2:='3)   750KHz  4)    375KHz   5) 187.5KHz  ';
       s3:='6) 93.75KHz  7) 46.875KHz';
       ch:=fetch_ch(s1,s2,s3);
       vaL(ch,n1,n2);
       if (n1>=0) and (n1<8) then
       begin
          hold_clk:=n1;
          set_clk(n1);
          if (n1<2) then storev.refp_seg:=3 else refp_seg:=2;
       end;
       draw_vars
    end;
'z','Z':begin
      default_rst;
      set_uno_line;
      u_mode:=true;
      draw_vars;
      if quiet_sample then
      begin
          fetch_data;
          ch:=#13
      end;
      u_mode:=false;
    end;
'r','R':begin default_rst;delay(1000);end;
'h','H':begin
     s1:='Line frame count :-';
     s2:='0) 128, 1) 129, 2) 130, 3) 131 ?';
     s3:='';
     ch:=fetch_ch(s1,s2,s3);
     val(ch,n1,n2);
     if NOT (n1<0) and (n1<4) then
     begin
        frame:=n1;
        bz:=n1;
        case bz of
         128:bz:=00;
```

```
          129:bz:=$10;
          130:bz:=$20;
          131:bz:=$30;
         else bz:=00
        end;
       port[modc]:=bz or init_modc;  {set test bit send selp0-1}
       o_modc:=bz or init_modc;
       draw_vars
     end
   end;
   'p','P':begin
          s1:='Number of pixels:';
          s2:='    1) 126, 2) 127, 3)128';
          s3:='';
          ch:=fetch_ch(s1,s2,s3);
          val(ch,n1,n2);
          if (n1 in [1..3]) then
          begin
              pixp_seg:=n1+125;
              draw_vars
          end
     end;
 'e','E':begin
        s1:='Number of ref. elements:';
        s2:='    1 to 3 ';
        s3:='';
        ch:=fetch_ch(s1,s2,s3);
        val(ch,n1,n2);
        if (n1 in [0..3]) then
        begin
            refp_seg:=n1;
            draw_vars
        end
    end;
 'g','G':get_21
                    end; {case}
                    {replot on carriage return}
                until (ch in [#13,'q','Q']); {inner loop}
                end {with STOREV}
     until (ch in ['q','Q'])  {outer loop}
 end; {with DISPV do}
 closegraph;
 g_flag:=false;
 with DISPV do freemem(cursxptr,sizex)
end;

procedure set_up_wr(n1:longint;n2,n3:byte);
{used in ram test}
```

```
begin
   {convert to 3 byte address in vars s_addhi etc}
   adj_if_addr(n1);
   set_if_addr;
   port[basep+n2]:=n3
end;

function set_up_rd(n1:longint;n:integer):byte;
{used in ram test}
begin
  adj_if_addr(n1);
  set_if_addr;
  set_up_rd:=port[basep+n]
end;

function ram_test(a1,a2,a3,a4:longint;v1,v2,v3,v4:integer):boolean;
var
  n:integer;
begin
   for n:=0 to 7 do   {do it for each card}
     begin
        set_up_wr(a1,n,v1);  {access 1st byte in each 32k ram chip}
        set_up_wr(a2,n,v2);
        set_up_wr(a3,n,v3);
        set_up_wr(a4,n,v4)
     end;
     ram_test:=false;
     for n:=0 to 7 do
     begin
       if set_up_rd(a1,n)<>v1 then ram_test:=true; {failed}
       if set_up_rd(a2,n)<>v2 then ram_test:=true;
       if set_up_rd(a3,n)<>v3 then ram_test:=true;
       if set_up_rd(a4,n)<>v4 then ram_test:=true;
    end
end;

procedure chk_ram;
var
  bz:byte;
  n1:integer;
  n3,a1,a2,a3,a4:longint;
  flag:boolean;
begin
   {spot ram checks.. 1 byte in each bank of ram is checked}
   {write bytes}
   sifm_ext_axs;        {set external access mode}
   sifm_ext_addr;       {set external addressing}
   a1:=0;a2:=$8000;a3:=$10000;a4:=18000;
   {base address of each ram chip}
   for n1:=1 to 3 do
   begin
     set_rack(n1);
     write('Ram Spot-Check, Rack [',n1,'] :');
     flag:=false;
     if ram_test(a1,a2,a3,a4,11,22,33,44) then flag:=true;
     if ram_test(a1,a2,a3,a4,$aa,$55,$AA,$55) then flag:=true;
     if ram_test(a1,a2,a3,a4,0,255,0,255) then flag:=true;
     if ram_test(a1,a2,a3,a4,255,0,255,0) then flag:=true;
     if flag then
       writeln('Memory test fail.. use (D)ebug to isolate fault.')
     else writeln('OK');
     delay(2000);
   end;
end;

procedure chk_sys;
var
 ch:char;
begin
   writeln('Starting basic checks.. ensure system power is on. ');
   writeln;
   writeln('Any key when ready ([w] if using screen-GRAB).');
   writeln;
   ch:=readkey;
   if (ch='w') or (ch='W') then
   {all white graphics to aid screen capture tsr}
   begin
     c_green:=white;
     c_red:=white
   end
   else begin
     c_green:=green;
     c_red:=red
   end;
   chk_ram; {spot check ram & rack link ok}
   default_rst;
   chk_vid;
   default_rst
end;

procedure help_main;
begin
  clrscr;
  writeln('Select function:');
  writeln('    (A)cquire');
  writeln('    (T)ransfer, pack and file');
  writeln('    (F)ile display');
  writeln('    (I)mmediate display');
  writeln('    (D)ebug functions');
  writeln('    (C)heck (auto) ram /video levels');
  writeln('    (H)elp, ( this table )');
  writeln('    (Q)uit');
  writeln
end;

procedure help_debug;
begin
  clrscr;
  writeln('Debug -select function:');
  writeln('    (A)cquire functions');
  writeln('    (C)ard');
  writeln('    (S)elect rack');
  writeln('    (D)irect manual i/o');
  writeln('    (R)ead block');
  writeln('    (W)rite block');
  writeln('    (L)ook for glitch');
  writeln('    (H)elp, ( this table )');
  writeln('    (Q)uit');
  writeln
end;

procedure help_io;
begin
 clrscr;
 writeln;
 writeln('Direct i/o -- Select function:');
 writeln('    o[xx]: set port offset to xx decimal.');
 writeln('        r: read from port.');
 writeln('    w[xx]: write xx decimal to port.');
 writeln('        q: quit to main menu.');
 writeln('        h: help, (this table).');
 writeln;
 writeln('xx in o[xx]:');
 writeln('     0-7: data port, cards 1 to 8');
 writeln('       8: address, high byte');
 writeln('       9: address, mid  byte');
 writeln('      10: address, low  byte');
 writeln('      11: mode A, bit 0-2=clock, bits 4-6=select video card');
 writeln('      12: mode B, bits 0-6: rst, arm, uno, cyc, cntaddr, start');
 writeln('      13: mode C, bit  3,4 =# of pixels per line 130/1024.');
 writeln('                  bit  7 =test1, set to ignore diode sync');
 writeln('      14: data');
 writeln('      15: status :- armed etc')
end;
```

```
procedure help_access;
begin
 clrscr;
 writeln('Select i/f status:');
 writeln('    (A)rm:- select acquire/display mode & arm (note TEST1');
 writeln('            not cleared).Use "E" to sync to diode array.');
 writeln('    (C)lear counter');
 writeln('    (S)tart sample (must be armed first)');
 writeln('    (E)xecute sample sequence ( sync to diode array )');
 writeln('    (U)no line mode, (sample for 1 line only');
 writeln('    (R)eset, set normal terminate on tc,restore multi-mode');
 writeln('    (P)ixels per line, hardware count: 128, 129, 130 or 131');
 writeln('    (I)gnore tc, set continuous cycling');
 writeln('   e(X)ternal data access mode');
 writeln('    (L)ocal bus addressing');
 writeln('    (D)irect bus addressing, (programmed)');
 writeln('    (F)requency: 0. 6MHz     1.   3MHz  2.   1.5MHz');
 writeln('                 3. 750KHz  4. 375KHz  5. 187.5KHz');
 writeln('                 6. 93.75KHz           7. 46.875KHz');
 writeln('    (T)oggle Test1, (set=ignore diode sync)');
 writeln('    (H)elp, ( this table )');
 writeln('    (Q)uit to main menu');
 writeln
end;

procedure access;
var
 bz:byte;
begin
 help_access;
 repeat
  write('-');
  ch:=readkey;
  case ch of
    'r','R':begin default_rst;delay(1000);end;
    'a','A':if arm_acquire(true) then begin end;
    'c','C':clear_count(true);
    's','S':start_acquire(true);
    'e','E':begin
           {clear TEST1 bit-- wait for diode sync}
           bz:=o_modc and $7f;
           port[modc]:=bz;
           if arm_acquire(true) then
           begin
              clear_count(true);
              start_acquire(true)
          end
```

```
        end;
    'u','U':set_uno_line;
    'p','P':set_ppl;
    'i','I':set_cyc_tc;
    'x','X':sifm_ext_axs;
    'i','I':sifm_int_addr;
    'd','D':sifm_ext_addr;
    'f','F':set_pix_clk;
    't','T':flip_test1;
    'h','H':help_access;
    else writeln('-');
  end
 until (ch in ['q','Q']);
 help_debug;
  ch:=' '
end;

procedure direct_io;
begin
 help_io;
 repeat
   write('-');
   ch:=readkey;
   write(ch);
   case ch of
     'o','O':get_io_port;
     'r','R':rd_io_port;
     'w','W':wr_io_port;
     'h','H':help_io;
    else writeln
   end;
 writeln;
 until (ch in ['q','Q']);
 help_debug;
 ch:=' '
end;

procedure debug;
var
bz:byte;
begin
  help_debug;
    repeat
      write('-');
      ch:=readkey;
      case ch of
      'a','A':access;
      'd','D':Direct_io;
      'c','C':gset_seg;
      's','S':g_rack;
      'r','R':blk_read;
      'w','W':blk_write;
      'l','L':blk_glitch;
      'h':help_debug
      else writeln('-')
      end;
   writeln;
   until (ch in ['q','Q']);
   help_main;
   ch:=' '
end;

procedure main_loop;
begin
   init_io;
   help_main;
   repeat
        writeln;
        writeln('Main menu -select a function (h- help :)');
        write('-');
        ch:=readkey;
  case ch of
    'a','A':begin
              write('Sampling Frequency is ');
              writeln((12000/exp(ln(2)*(storev.pix_clk+1))):4:2,' KHz');
              writeln('Do you wish to modify?');
              writeln;
              ch:=readkey;
              if (ch='y') or (ch='Y') then
              begin
                 writeln('0)     6MHz  1)      3MHZ   2)   1.5MHz   ');
                 writeln('3)   750KHz  4)    375KHz   5) 187.5KHz   ');
                 writeln('6) 93.75KHz  7) 46.875KHz');
                 writeln;
                 set_pix_clk;
              end;
              acquire_cycle(true);
         end;

      't','T':begin blk_store;default_rst;delay(1000);end;
      'f','F':blk_screen;
      'i','I':direct_screen;
      'd','D':debug;
      'c','C':chk_sys;
      'h','H':help_main;
   else writeln('-')
  end;  {case}
  until (ch in ['q','Q'])
end;


{main}

begin
   clrscr;
   writeln('High Speed Passive Tracking System -- Operating System');
   writeln;
   pathstr:='c:';  {used by graphics}
   detectgraph(grdriver,grmode);
   if (grdriver=ega) or (grdriver=ega64) or (grdriver=egamono)
   or (grdriver=vga) then
   else begin
           writeln('Sorry, requires EGA or VGA to run.');
           exit
   end;
   {initialise variables}
   with storev do
   begin
      hold_clk:=0;
      pixp_seg:=128;
      refp_seg:=3;
      pix_clk:=0;
   end;
   g_flag:=false;
   o_modc:=init_modc;
   o_modb:=v_reset;
   o_moda:=0 ; {start with clk=0, rack sel0-3 =0}
   port[moda]:=o_moda;
   o_clk:=o_moda;
   port_num:=basep;
   o_filen:='TEST.M11';
   c_green:=green;
   c_red:=red;
   c_white:=white;
   set_clk(0);
   {initialise hardware}
   init_io;
   with DISPV do
   begin
      bz:=$30;
      frame:=3;
      port[modc]:=bz or init_modc;  {set test bit send selp0-1}
      o_modc:=bz or init_modc;
      start_seg:=1;
   end;
   init_storev; {set default store variables}
   init_dispv;
   default_rst;
   repeat
      main_loop;
      writeln;
      write('Sure (y/n)? ');
      readln(ch);
   until (ch='y') or (ch='Y') or (ch=#27)
end.
```