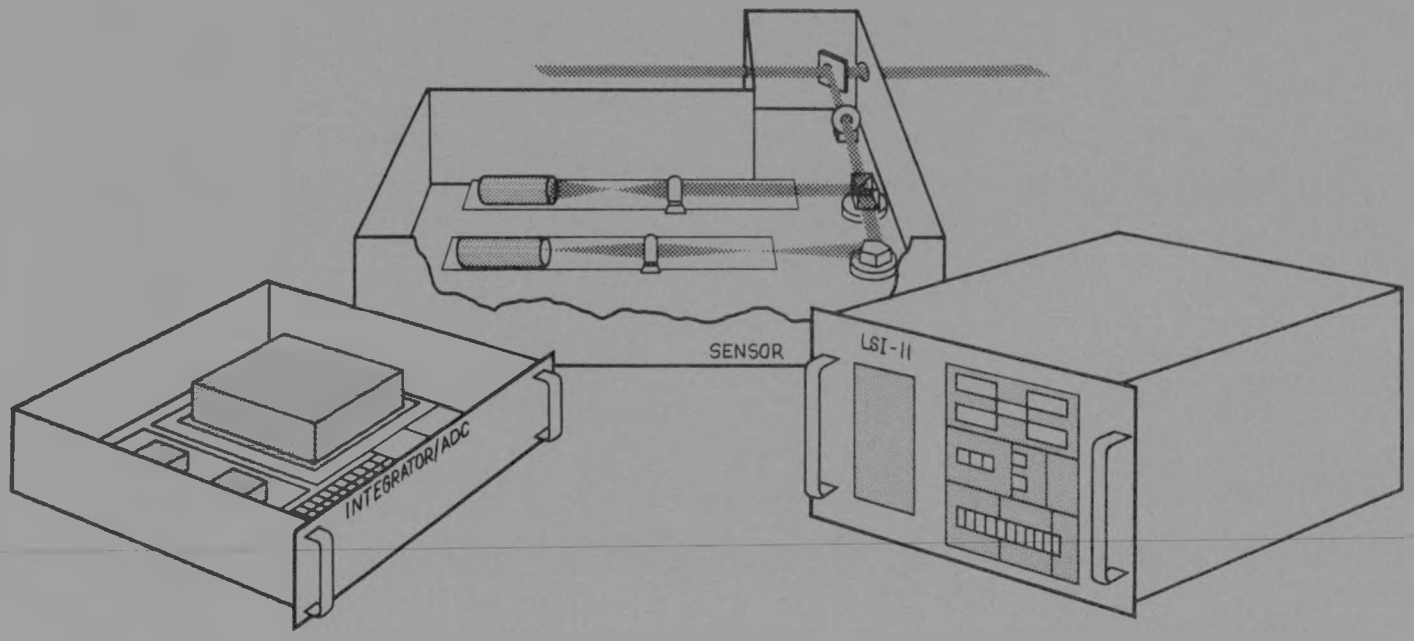
M-114

Shiva Oscillator Alignment System

Reference Manual



May 27, 1980

University of California
Lawrence Livermore National Laboratory

ACKNOWLEDGMENTS

The Oscillator Alignment System described in this document was the result of the combined efforts of many people. The primary design was done by Mark Summers, John Parker, and Bob Cody. Significant contributions were also made by Erlan Bliss, Bob Boyd, Roger Gant, Fred Holloway, Greg Suski, and Paul Van Arsdall. The REBEL/BASIC software was developed by Jim Greenwood.

Much of the artwork for this document was done by Darlene Neidhardt.

Work performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore Laboratory under Contract W-7405-Eng-48.

M-114

Shiva Oscillator Alignment System Reference Manual

John Parker

Manuscript date: May 27, 1980

University of California
Lawrence Livermore National Laboratory

Available from: National Technical Information Service • U.S. Department of Commerce
5285 Port Royal Road • Springfield, VA 22161 • $9.00 per copy • (Microfiche $3.50)

CONTENTS

Shiva Oscillator Alignment System Reference Manual

I. INTRODUCTION

Shiva is a 20-arm Nd:glass laser used for inertial confinement fusion experiments. The broad range of control, data acquisition, and analysis functions required to support the system has resulted in the implementation of a hierarchal computer network in which these functions have been divided into major subsystems: power conditioning, fusion target diagnostics, beam diagnostics, and alignment control. Figure 1 illustrates the control architecture.

The alignment control subsystem includes pointing, centering, and focusing, as well as certain selection operations involving many hundreds of control points from the oscillators to the target. Alignment is accomplished with a chopped CW YAG laser directed from the oscillator through all 20 laser chains to a surrogate target. Sensors placed at strategic locations detect errors and provide feedback through the control system to motor driven gimbals. A typical Shiva beam line is illustrated in Fig. 2.

SYSTEM REQUIREMENTS AND DESCRIPTION

The Oscillator Alignment System (OAS) for the Shiva laser was designed to automatically point and center both a CW alignment laser and any of several pulsed oscillators. The laser beams are pointed and centered with a mirror pair whose axes are driven by error signals derived from a sensor located in the preamplification stage. Figure 3 shows a plan view of the oscillator and preamp tables.

Three oscillators must be coaligned by the OAS: the master pulsed oscillator, a CW alignment laser, and the output of a pulse synchronization system used for temporal alignment of Shiva's 20 beams.

A single pulse from a mode-locked train produced by the master oscillator is selected and injected into a temporal pulse shaper. Typically, the selected pulse has a FWHM (full width at half-maximum) of about 0.1 ns and an energy of 100 μJ. The shaped pulse, for early experiments, will be comprised of a stack of 0.1 ns pulses suitably attenuated and delayed in time.

Alignment subsystems which follow the OAS (chain input pointing and chain output pointing and centering) for directing the beam through the 20 amplifier chains are designed to operate with a high-power CW YAG laser modulated at 630-Hz. Therefore, the OAS must be capable of coaligning the pulsed master oscillator and the modulated CW oscillator. The 630-Hz modulation is achieved with a chopper wheel located near the output of the CW laser.

The pulse synchronization system contains its own CW mode-locked oscillator whose signal is comprised of an infinite train of 200-ps pulses separated by about 10 ns and modulated at 630 Hz. Thus, the OAS detector (response time ~0.5 μs) will see it as another 630-Hz alignment beam, as will all other alignment subsystems.

Alignment errors in the oscillator and preamplification sections are primarily the result of drift in table supports, mirror mounts, and lens mounts, in addition to inherent beam wander. Motions of flat transmitting components such as rod amplifiers, Pockels cells, and polarizers contribute to a lesser degree.

The OAS is a microprocessor-based control system which points and centers any of the previously mentioned laser oscillators. Pointing and centering errors are sensed by imaging the far-field and an appropriate near-field plane, respectively, onto lateral effect photodetectors. Signals from the detectors are buffered, integrated, converted to digital format, and stored in the microprocessor. Calculations are then performed to decouple pointing and centering, and the results are used to command gimbal motors to move,

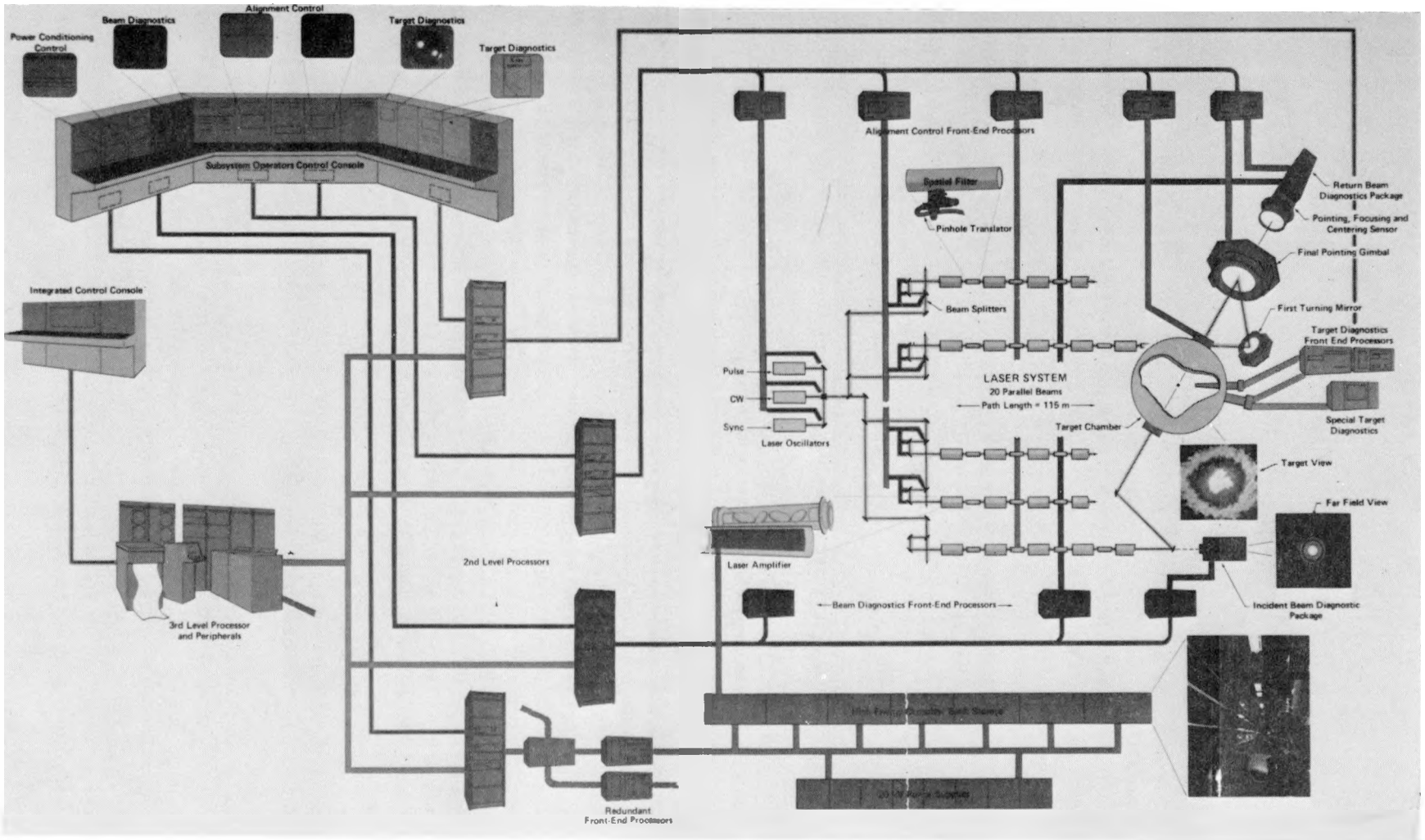


FIG. 1. Shiva control system.

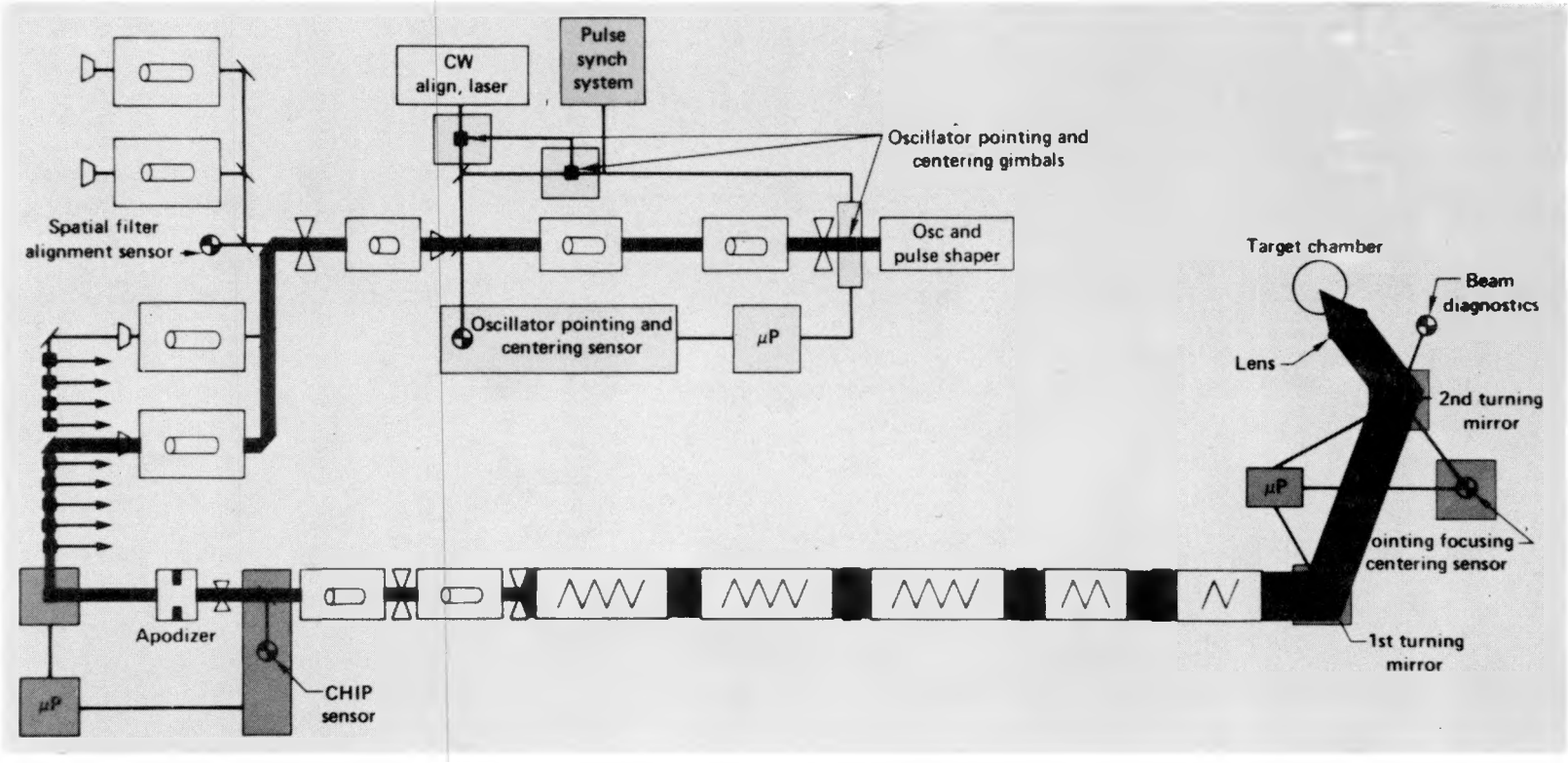


FIG. 2. Shiva beam line.

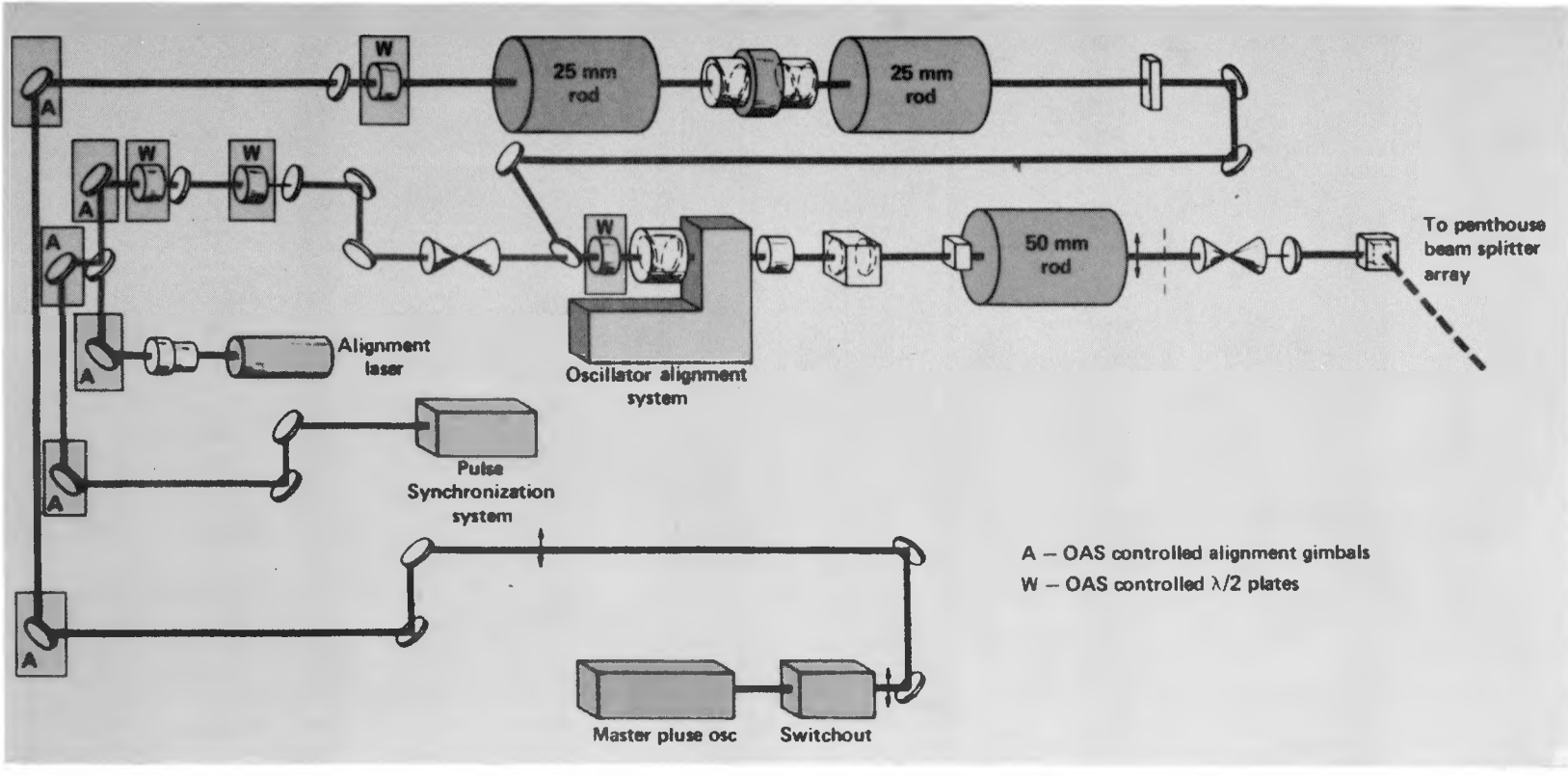


FIG. 3. Shiva oscillator and preamplifier.

correcting the alignment errors. The use of gated integrators with programmable sample widths permits both short-pulsed and relatively longer, chopped CW signals to be processed.

Since the purpose of the OAS was to eliminate long term drift of the oscillators, only a modest repetition rate (~1 Hz) was required. In fact, the CW oscillator is now so stable that, after initial alignment, little or no correction is necessary for hours. A more subtle and very useful characteristic of the system is that one can switch between oscillators and, indeed, align any of them in a matter of minutes.

The OAS allows either closed loop alignment of any of the lasers or open loop operation, through an operator, for monitoring and system adjustment. In the closed loop mode, alignment errors are sampled, gimbals moved, and the process repeated until the alignment errors are sufficiently small. Open loop operation allows the operator to manually move motors (gimbals or waveplates used as attenuators) and to store motor positions for future use. Both modes (closed and open loop) provide the ability to display any of several system parameters, including status, data, and error conditions.

The OAS assures a pointing and centering accuracy at the sensor of $\pm 10\ \mu$rad and ± 10 mils, which implies a chain output error of no greater than approximately 0.125 μrad. This contribution is negligible compared with the alignment error introduced by other system elements and also ensures that the requirements for centering on the apodizing aperture will be satisfied.

Details of optical design, electronics and signal processing, decoupling calculations, and software considerations will be described in the following sections, ending with a discussion of system resolution and response.

II. OPTICS

The first element of the OAS, located on the preamplifier table, is the ALIGNMENT SENSOR (Fig. 4), which splits off a small percentage of the main laser beam and senses pointing and centering deviations from a predetermined nominal position.

Figure 5 shows how pointing and centering errors are sensed. When the far-field pattern is imaged, pointing errors appear as translations in the focal plane of the lens. This focal plane is then imaged onto a lateral effect photodetector. (Any centering error on this lens will not cause beam movement in the far field.) When a plane located between the entrance to the sensor and the last preceding optical element in the laser chain is imaged (so that error contributions from the previous elements are accountable), centering errors are detected, and displacements from nominal are related to the position on the photodetector by the magnification of the intermediate lens.

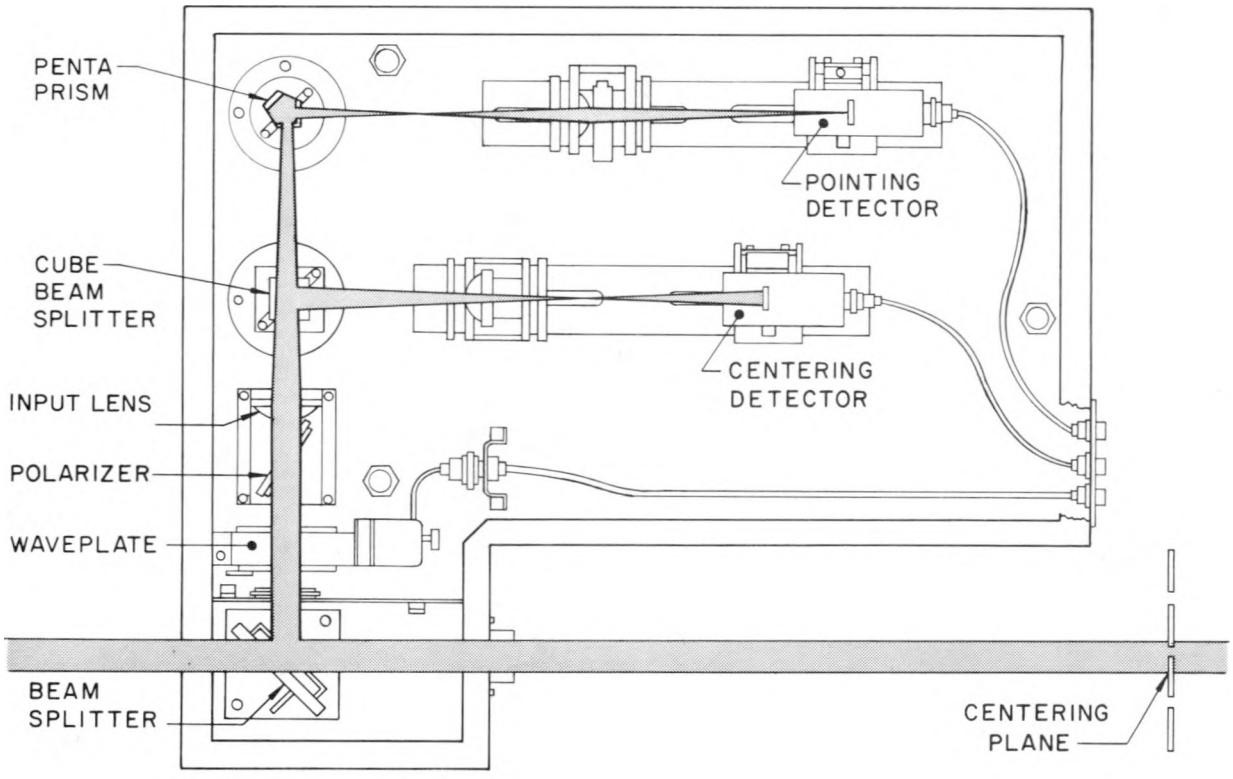


FIG. 4. Shiva oscillator alignment sensor.

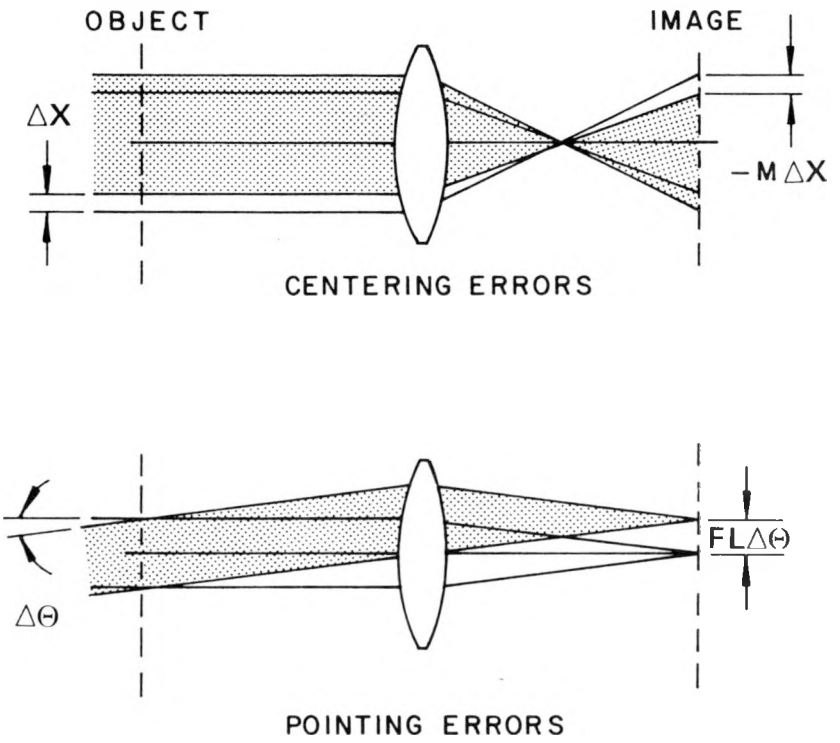


FIG. 5. Sensing alignment errors.

Accurate centering and pointing requires that the error sensitivity match the resolution of the photodetector. The detector used was a United Detector SC/10 lateral effect photodetector with a beam centroid resolution capability of 1 mil.

For small angles the angular error (θ) is related to the translational movement (S) in the focal plane by the focal length (f) of the lens:

$$S = f\theta.$$

With S = 0.001 in. = $25.4(10^{-6})$ m, the resolution of the detector, and $\theta = 4\,\mu r$, the required error sensitivity, the focal length of the lens is

$$f = \frac{S}{\theta} = 6.35 \text{ m}.$$

This long focal length is packaged in a simpler form by assuming a two-lens system, the first lens having a focal length of 600 mm and the second lens magnifying the far field of the first lens by 10.6. This is accomplished by using a microscope objective with an effective focal length of 18.30 mm as the second lens. The lens system is thus reduced in size to 0.8 m, yet the far field has the correct magnification.

Since a single beam-splitter is used to sample the main beam, the first lens (L1) is used for both the centering and pointing portions of the optical system. A cube beam-splitter is subsequently used to divide the beam equally between pointing and centering detectors without introducing astigmatism. The accuracy required in centering is 0.1 mm; the beam diameter at the sensing plane is 21 mm. Hence, the required magnification can be calculated by finding the movement on the detector that corresponds to the required accuracy in the sensing plane, and taking its ratio to the beam diameter in the sensing plane:

$$\frac{0.1 \text{ (accuracy required)}}{21 \text{ mm (beam diameter)}} = \frac{1 \text{ mil (resolution of detector)}}{D \text{ (required diameter of beam on detector)}}$$

$$D = \frac{(1 \text{ mil})\ (21 \text{ mm})}{0.1 \text{ mm}} = 5.33 \text{ mm}.$$

The minimum required magnification is thus

$$M = \frac{5.33}{21} = 0.254.$$

Lens L3 must therefore image the sensing plane onto the centering detector with a magnification of 0.254. Since this distance from L1 to the sensing plane is 1400 mm, selecting the focal length of L3 to be 400 mm and the distance between L2 and L3 to be 350 mm yields the required magnification.

The alignment sensor must also be able to accommodate a wide variation in energy levels, from a fast pulse oscillator to a CW alignment laser. The beam splitter that samples the main beam was designed with a reflectivity of approximately 10%, since the pulse energy at the sensor is 100 μJ and the detectors require at least 500 μJ. An attenuator comprised of a polarizer-half-wave plate combination accommodates a 5- to 10-W CW laser without the physical interchange of neutral density filters. Since the polarization vector rotates 2 θ as the wave plate subtends an angle of θ, the intensity transmitted through the combination varies as $\cos^2\theta$. The maximum attenuation obtained is approximately 10^{-3}. Energy not transmitted is reflected by the polarizer into a power dump consisting of BG-8 glass, thereby reducing any stray light in the system. Since the half wave plate is stepping motor driven, its rotation angle can be controlled by computer. In addition, a shutter was added to the rotation mount so that the detectors will be protected during amplified pulse laser shots.

To aid in maintaining a clean environment for the pulse beam, the sensor is sealed off by an internal wall from the beam splitter that samples the beam. Optical windows are used for the transmitted and sampled beam.

III. DECOUPLING COMPENSATION

Since the Oscillator Alignment System uses a two-gimbal configuration to correct both pointing and centering errors, a combination of motions of both gimbals must be used to correct pointing without decentering, and vice versa.

Figure 6 is a top view illustration of the two-gimbal arrangement.

For pure translation of the beam in the horizontal direction (plane of the page), the centering displacement changes according to

$$\epsilon_x = 2\theta_{21_x}\left(L_1 + L_2\right) - 2\theta_{11_x} L_1 \tag{1}$$

In order that no pointing error be introduced, the two gimbals must be moved in the same direction by the same amount; that is,

$$\theta_{21_x} = \theta_{11_x} = \epsilon_x / 2L_2 \tag{2}$$

for pure horizontal translation.

For pure rotation of the beam in the azimuthal direction (plane of the page), the angular displacement is proportional to twice the net angular movement of the gimbals. In order that no centering error be introduced, the two gimbals must be moved in opposite directions. Thus,

$$\epsilon_\alpha = -2\left(\theta_{11_\alpha} - \theta_{21_\alpha}\right). \tag{3}$$

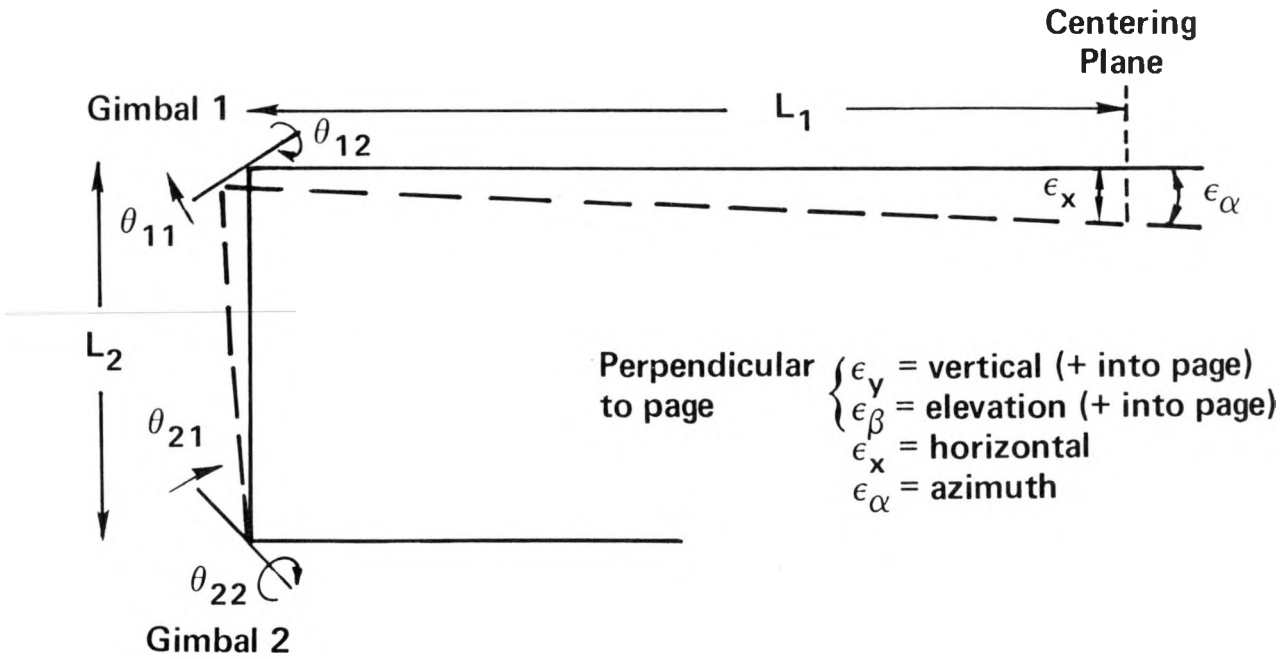


FIG. 6. Geometry for OAS two-mirror decoupling.

Moving gimbal 2 through an angle θ_{21_α} results in an angular displacement on gimbal 1 of $2L_2\theta_{21}$. This produces an angular displacement (far field) of $-\epsilon_\alpha L_1$. Thus,

$$-\epsilon_\alpha L_1 = 2L_2\theta_{21_\alpha} \tag{4}$$

for pure azimuthal rotation due to movement of gimbal 2.

Moving gimbal 1 through θ_{11_α} results simply in an angular displacement of $-\epsilon_\alpha/2$ to which must be added any angular change introduced by gimbal 2.

Combining Eqs. (2) and (4), the expression for θ_{21} becomes

$$\theta_{21} = \theta_{21_x} + \theta_{21_\alpha} = \frac{\epsilon_x}{2L_2} - \frac{\epsilon_\alpha}{2}\left(\frac{L_1}{L_2}\right), \tag{5}$$

since the movement of gimbal 2 must account for both translation and rotation.

Substituting (4) into (3) to eliminate θ_{21_α} and considering Eq. (2) yields

$$\theta_{11} = \theta_{11_x} + \theta_{11_\alpha} = \frac{\epsilon_x}{2L_2} - \frac{\epsilon_\alpha}{2}\left(1 + \frac{L_1}{L_2}\right), \tag{6}$$

since gimbal 1 must also account for both translation and rotation.

In the orthogonal direction (ϵ_y, vertical; ϵ_β, elevation) a slightly different situation occurs. Assuming that the incident angle of the beam on the gimbal is 45° and the angle through which the gimbal is moved remains small, Fig. 7 illustrates that the gimbal angle and beam reflected angle are related by

$$\gamma = \sqrt{2}\delta \ . \tag{7}$$

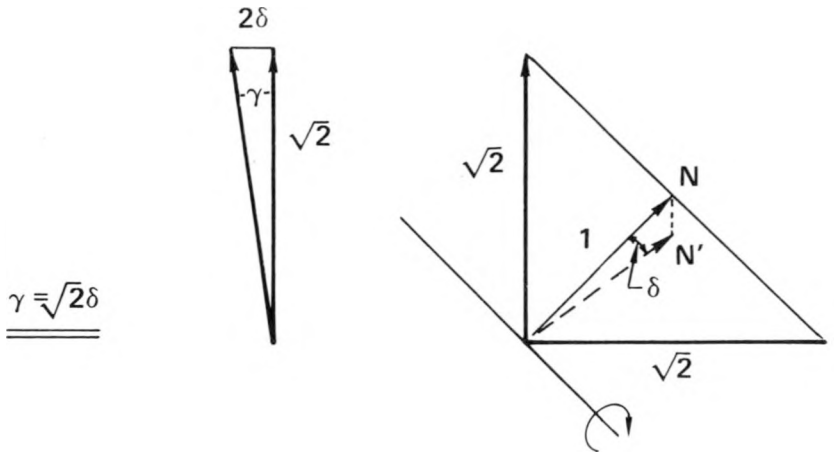


FIG. 7. Tilting gimbal through angle δ results in a change in reflected angle of γ.

Taking (7) into account for vertical and elevation directions, Eqs. (1) and (3) may be rewritten as

$$\epsilon_y = -\sqrt{2}\, L_2\, \theta_{22_y} - \sqrt{2}\, L_1 \left(\theta_{12_y} + \theta_{22_y}\right) \tag{8}$$

and

$$\epsilon_\beta = -\sqrt{2}\left(\theta_{12_\beta} + \theta_{22_\beta}\right). \tag{9}$$

Using arguments similar to those for the horizontal and azimuthal cases above, the expressions for the effect of gimbal motions on vertical and elevation errors become

$$\theta_{12} = \theta_{12_y} + \theta_{12_\beta} = \frac{\epsilon_y}{\sqrt{2}L_2} - \frac{\epsilon_\beta}{\sqrt{2}}\left(1 + \frac{L_1}{L_2}\right) \tag{10}$$

and

$$\theta_{22} = \theta_{22_y} + \theta_{22_\beta} = -\frac{\epsilon_y}{\sqrt{2}L_2} + \frac{\epsilon_\beta}{\sqrt{2}}\frac{L_1}{L_2}. \tag{11}$$

Additional factors are included to account for units, gimbal motor resolution, and magnification in the expressions used in the OAS control software.

IV. SYSTEM ANALYSIS

The OAS performs the necessary alignment functions by moving gimbal motors to correct errors sensed by the two lateral effect photodetectors. The transformation from alignment errors to motor distances involves factors such as detector characteristics, magnification, cross-coupling, angular effects, and motor resolution. The combination of these factors is best described in terms of matrix operations which transform alignment errors back through the system to motor distance.

After the detector signals have been converted to digital format and normalized in software, the data matrix is multiplied by the detector matrix (DTCR). The latter describes the specific detector characteristics in terms of the figure of merit, R/S (Responsivity/Sensitivity), and the magnification of the sensor optics for pointing and centering. For the present OAS, the (DTCR) matrix is given by

$$(\mathrm{DTCR}) = \begin{pmatrix} M_p(\mathrm{FM}) & 0 & 0 & 0 \\ 0 & M_p(\mathrm{FM}) & 0 & 0 \\ 0 & 0 & M_c(\mathrm{FM}) & 0 \\ 0 & 0 & 0 & M_c(\mathrm{FM}) \end{pmatrix}$$

$$= \begin{pmatrix} 4(250) & 0 & 0 & 0 \\ 0 & 4(250) & 0 & 0 \\ 0 & 0 & 3.68(250) & 0 \\ 0 & 0 & 0 & 3.68(250) \end{pmatrix}$$

where M_p is the pointing magnification, M_c is the centering magnification, and (FM) is the figure of merit. With this transformation, alignment errors at the detector are converted to errors in the beam at the sensor.

Next, the magnification of beam-line components must be accounted for. The matrix (MAG) performs this transformation:

$$(\mathrm{MAG}) = \begin{pmatrix} M_{HAZ} & 0 & 0 & 0 \\ 0 & M_{VEL} & 0 & 0 \\ 0 & 0 & 1/M_{HAZ} & 0 \\ 0 & 0 & 0 & 1/M_{VEL} \end{pmatrix}.$$

M_{HAZ} denotes the horizontal or azimuth magnification; M_{VEL} denotes the vertical or elevation magnification. Pointing errors are increased and centering errors decreased in GOING BACK through the system.

The final step in converting to errors at the gimbals is transformation via a cross-coupling matrix (CC). The elements of this matrix were derived in section III.

$$(CC) = \begin{pmatrix} -\left(1 + \frac{L_1}{L_2}\right) & 0 & \frac{10^6}{L_2} & 0 \\ 0 & -\left(1 + \frac{L_1}{L_2}\right) & 0 & \frac{10^6}{L_2} \\ -\frac{L_1}{L_2} & 0 & \frac{10^6}{L_2} & 0 \\ 0 & \frac{L_1}{L_2} & 0 & -\frac{10^6}{L_2} \end{pmatrix}.$$

The relationship between beam movement, gimbal movement, and motor resolution must be considered in converting from errors at the gimbals to motor distances. A gimbal movement of θ in the horizontal plane results in a beam movement of 2θ, while a movement of $\sqrt{2}\theta$ results in the vertical plane. The (ANG) matrix accounts for this:

$$(ANG) = \begin{pmatrix} 1/2 & 0 & 0 & 0 \\ 0 & 1/\sqrt{2} & 0 & 0 \\ 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1/\sqrt{2} \end{pmatrix}.$$

Each gimbal is driven by a stepper motor of known resolution in microradians per step. In addition, the clockwise-counterclockwise motion of the motor must be reconciled with the predefined gimbal angles. Multiplication by the (GMB) matrix is the final step in converting alignment errors at the detector to motor distance required to correct the errors:

$$(GMB) = \begin{pmatrix} 1/1.45 & 0 & 0 & 0 \\ 0 & -1/1.45 & 0 & 0 \\ 0 & 0 & 1/1.45 & 0 \\ 0 & 0 & 0 & -1/1.45 \end{pmatrix}.$$

With the matrices defined above, the total transformation with the intermediate results obtained is illustrated below:

$$(\text{DATA})\,(\text{DTCR})\,(\text{MAG})\,(\text{CC})\,(\text{ANG})\,(\text{GMB}) = (\text{Motor Distances})\ .$$

Errors at detector — Errors at sensor — Errors at gimbals

Figure 8 shows how centering errors, ϵ_x and ϵ_y, and pointing errors, ϵ_α and ϵ_β, are transformed from the main beam to the detectors.

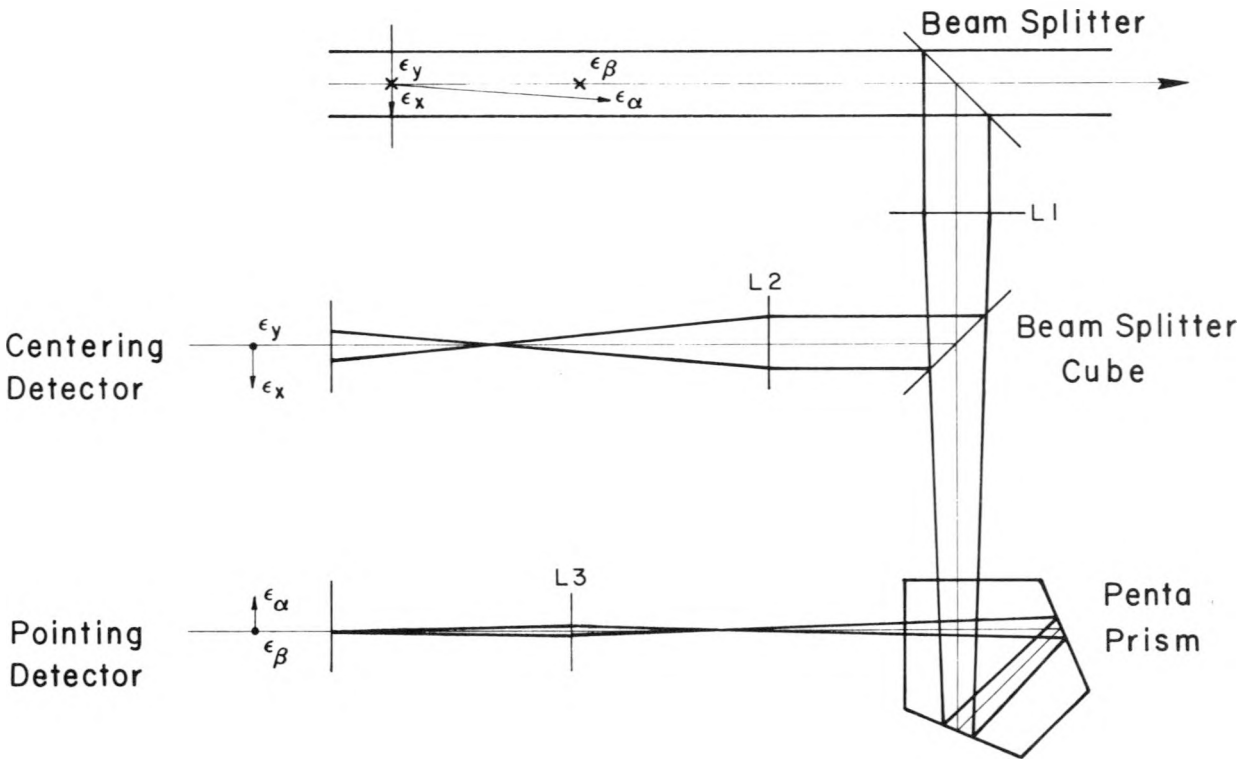


FIG. 8. OAS sensor optics.

V. ELECTRONICS

After receiving analog signals from continuous position sensing photodetectors located in the oscillator alignment sensor, the Oscillator Alignment System Integrator/ADC package integrates, multiplexes, A/D converts, buffers, and transfers them to an LSI-11 for processing. Figure 9 illustrates this signal processing.

The package consists of three functional units: the Integrator/ADC, which performs the integration, analog multiplexing, and A/D conversion of the input signals; the Sequencer, which provides timing and control signals for the Integrator/ADC, in addition to FIFO buffering and handshaking circuitry for interface with the LSI-11; and the Power Supply, which supplies local +15 V and +5 V for the Integrator/ADC and Sequencer. The Integrator/ADC utilizes approximately 7.5 W of 5-V power and 1.7 W of combined ±15-V power. The 5-V and ±15-V supplies are rated at 2.5 A and 0.5 A, respectively.

The Oscillator Alignment System Integrator/ADC package is controlled by means of the software routine OASIS, which specifies integrator sampling time and trigger selection in addition to providing timing and control for the transfer of alignment data to the LSI-11 for processing.

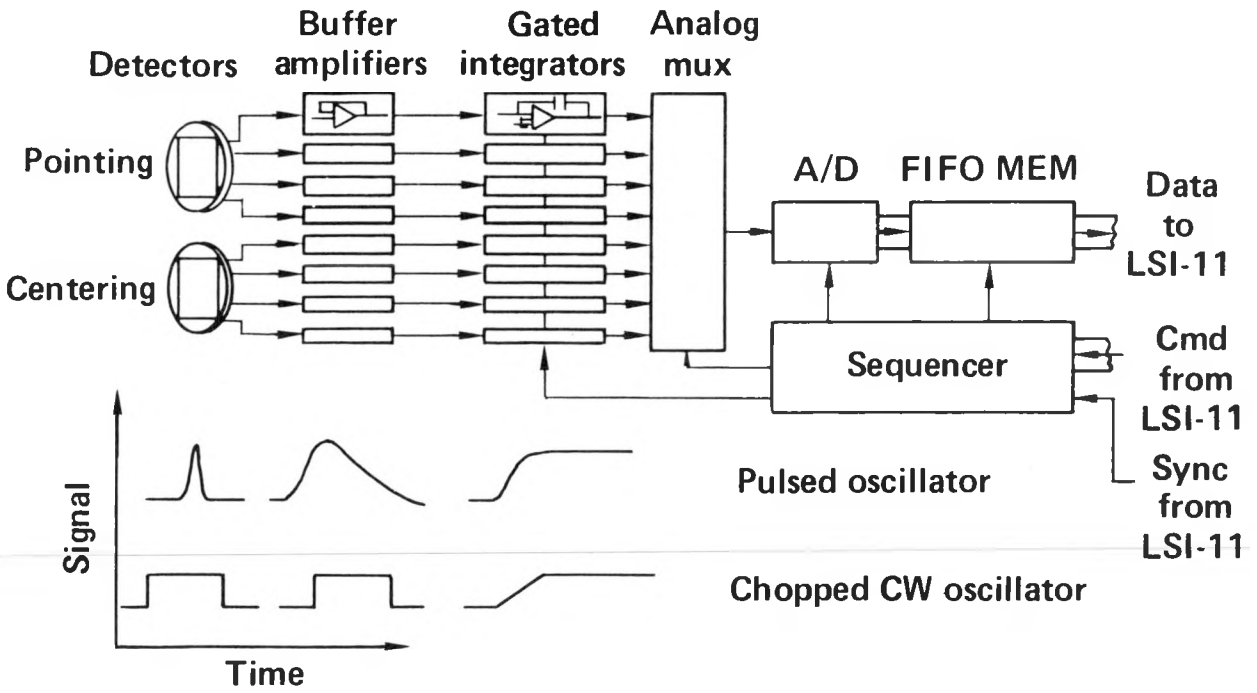


FIG. 9. OAS signal processing.

GENERAL DESCRIPTION

The optical signals are focused or imaged onto two (pointing and centering) lateral-effect photodiodes, each of which outputs two pairs of analog signals. The difference in the current signals is a function of the location of the beam. Calculations based on these data (digitization of the signals) are used to generate commands to stepping motor driven gimbal pairs which move to correct any alignment error.

The closed loop control sequence is initiated when the LSI-11 issues a reset command (DTARCDN) to the Integrator/ADC thereby initializing all Sequencer logic. This actually serves as a redundant check on the initial state of the Sequencer since a reset is done on power-up of the Integrator/ADC chassis (PWRRST). Simultaneously, a gate-width generator is loaded with an appropriate value so as to provide a desired integration time upon reception of a delayed trigger pulse; an appropriate trigger channel is also addressed on the Trigger Multiplexer (located on the back panel of the LSI-11). Finally, a KWV11-A programmable clock is reset and loaded with a value representing a time delay to be inserted between the actual trigger signal and the trigger signal sent to the Integrator/ADC.

The system is then in its initial state, ready to receive a trigger signal which will initiate all subsequent functions. The selected trigger (via the addressable trigger multiplexer) is optically isolated through the trigger multiplexer, and input to the KWV11-A programmable clock. After the programmed time delay has elapsed, a signal is sent to the Integrator/ADC initiating the Sequencer which provides all the timing and control for the package.

The output signals of the integrators are subsequently multiplexed, buffered, converted to 12-bit words, and stored in a FIFO buffer. The A/D converter is unipolar with an input range of 0 to 10 volts.

DETECTION

Figure 10 shows the OAS detection circuitry, and a typical integration circuit.

An LM 310 is used as a buffer amplifier and the bias is chosen to be 1K, so that there is sufficient signal from the detector diode without producing a nonlinear response. The integrator resistance R was then chosen so that the overall gain produced a nominal 5-V signal out of the integrators. It should be kept in mind that the gain of the system is *programmable* since the gating time is controlled from software.

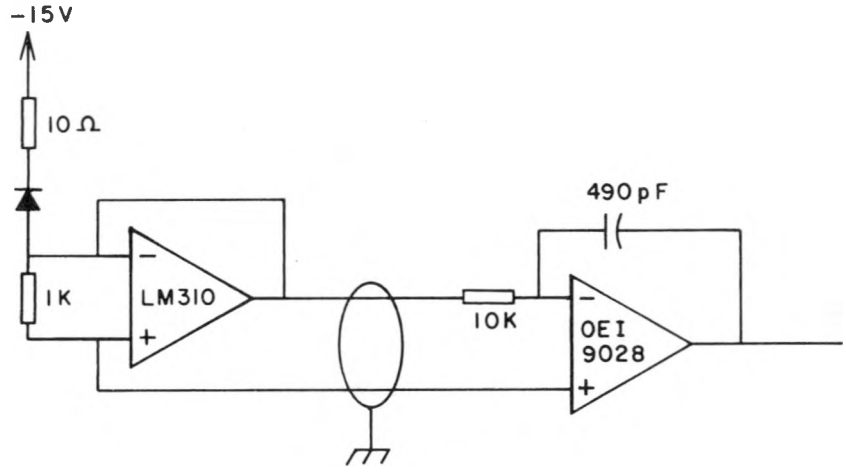


FIG. 10. OAS detection circuitry, including a typical integration circuit.

The detected error can be related to the current flowing in the detector diodes by means of the responsivity and sensitivity of the detector. If R is the responsivity in mA/mW, S is the sensitivity in mA/mW-mil, and P is the power on the detector in mW, then the difference in the detector currents, Δi, is given by

$$\Delta i = PS\Delta x,$$

where Δx is the distance (error) from nominal (0,0) on the detector, and the total current is

$$\Sigma i = RP.$$

Since Δi changes with signal level, it is necessary to normalize by Σi so that the normalized error is

$$\frac{\Delta i}{\Sigma i} = \frac{PS\Delta x}{RP} = \frac{S}{R}\Delta x.$$

Therefore, the error may be determined by measuring the detector currents according to:

$$\Delta x = \frac{R}{S}\frac{\Delta i}{\Sigma i}.$$

The sum and differencing of currents is easily performed in software. R/S is the figure of merit for the SC-10 photodetector used and has a value of 250 at $\lambda = 1.06\ \mu m$. Any drift in the electronic circuitry is thus compensated for, and the electronic "zero" position of the detectors may be defined.

INTEGRATION

The Integrator/ADC receives eight analog signals (four from each photodetector) and performs a gated integration of each. Figure 11 depicts the circuit.

Inherent offsets in the detectors and the integrator modules are adjusted out by means of the Offset and Slope adjustment networks.

RESET L sets up an initial condition (offset adj.) prior to integration of the signal for a time determined by SAMPLE L. This mode of operation permits the isolation of a single oscillator pulse in time or sampling of a CW oscillator.

The output voltage is given by

$$e_o(t) = -\frac{1}{RC}\int_0^t e_i(t)\,dt.$$

With $e_i(t)$ being a DC level, for example, with the CW oscillator, and RC given as (10K) (490pF), the expression for the output voltage simplifies to

$$E_o = -2(10^5)\,E_i\Delta t$$

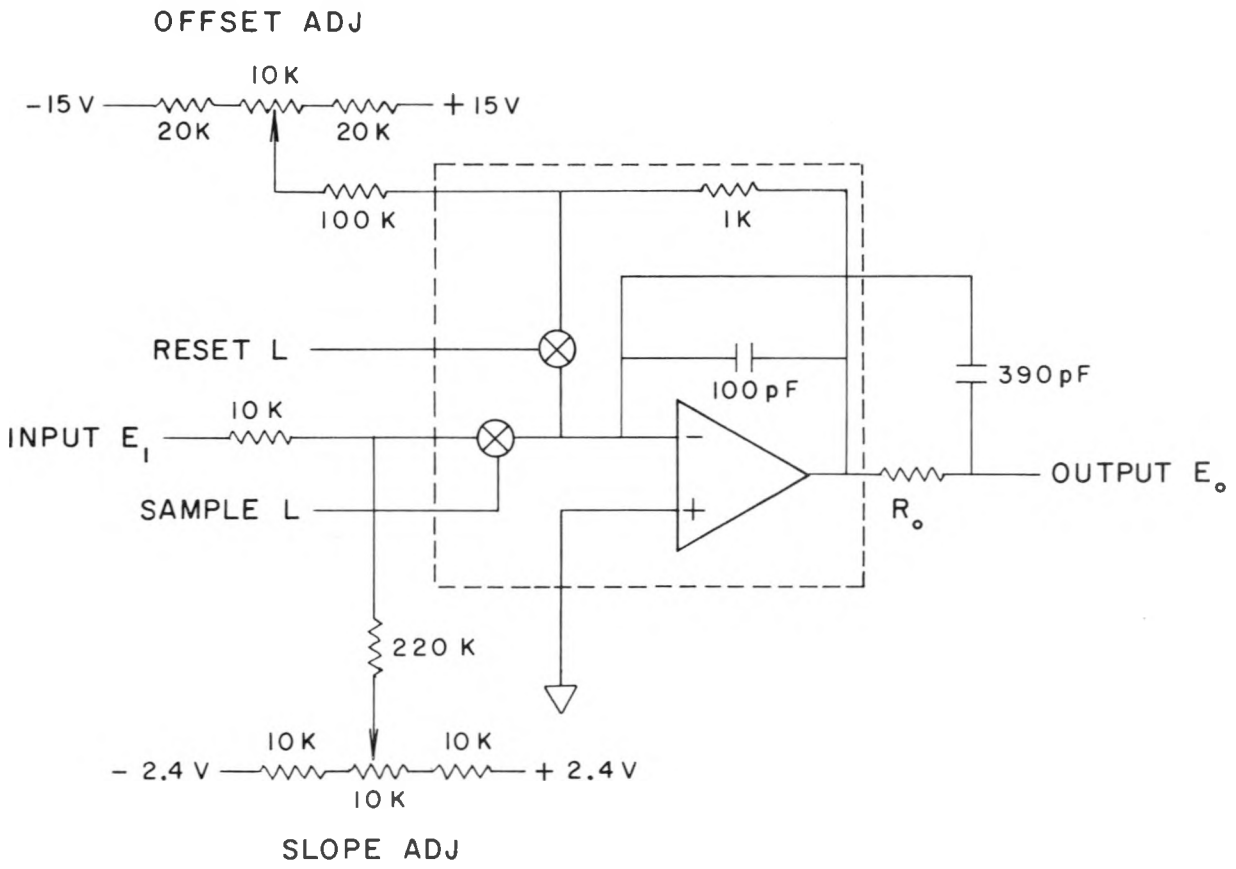


FIG. 11. Integration circuit detail.

where Δt is the sampling or integration time.

The sampling time is programmable in 1-μs increments from 1 μs to 256 μs; times on the order of <10 μs are avoided, however, due to settling time considerations of the internal analog switches.

When the FIFO is full, the Sequencer sends an interrupt request to the LSI-11; upon servicing the interrupt, the data is transferred from the FIFO to the LSI-11 parallel interface (DRV11) via a specially designed isolator (DRV11 Driver-Receiver LEA 77-1257). Upon reception of the digitized signals, the LSI-11 sends an indication signal (DTARCD) to the Integrator/ADC. This signal, once again, initializes all the Sequencer logic so that the procedure may repeat. With the digitized detector signals in the LSI-11, the alignment errors can be determined and the gimbal commands calculated. These commands are issued via Stepping Motor Logic and Drive Circuitry (LEA 76-1344) to gimbal stepping motors effecting closed loop control and correction of pointing and centering alignment errors.

SEQUENCER

The Sequencer's function is to generate all the timing and control signals for the Integrator/ADC package including Reset and Sample signals for the integrators, addressing for the analog multiplexer, and convert command for the A/D converter. In addition, the Sequencer controls the FIFO buffer and performs handshaking functions for interface with the LSI-11.

The Sequencer is divided into the following functional sections:

- Generation of SAMPLE L
- Convert control of A/D converter
- Address sequencing of multiplexer
- FIFO buffer storage
- Interface (handshaking) with LSI-11.

Currently, 12 control bits are used by the Oscillator Alignment System; 10 by the Integrator/ADC package; and 2 by the Trigger Multiplexer Board (located on the rear panel of the LSI-11) which uses bits 10 and 11 of the DRV11 output word to select one of four trigger sources. The breakdown of control bit usage by the Integrator/ADC package is as follows:

15	14	13	12	11	10	9	8
Not Used	Not Used	Not Used	Not Used	Trig Mux 1	Trig Mux 2	DTARCDN	TSTMDE

7	6	5	4	3	2	1	0
PN128	PN64	PN32	PN16	PN8	PN4	PN2	PN1

The eight least significant bits determine the width of the SAMPLE L pulse for the integrators. Bit 8, TSTMDE, determines the mode of operation, either normal or test, for the Integrators. Bit 9, DTARCDN, is a control bit which, when asserted (low), indicates that 12 *words* of data from the FIFO have been transferred to the LSI-11.

Generation of Integrator Sample Width

The OAS must align a variety of CW and pulsed laser oscillators. In the detection and data conversion process, integrators with programmable integration widths were chosen to provide a convenient method of gain adjustment.

The circuitry shown in Figs.12 and 13 generate a gate for the integrators dependent on the eight-bit value received from the control program via the DRV-11 parallel interface. The circuit in Fig. 12 provides a stable clock source for the SAMPLE generation circuitry. A 20-MHz oscillator is divided and used as the input to an up/down counter. A combination of the MSB and LSB is then used to gate a triplet of J/K flip-flops, thus generating three phases of a 1-MHz clock. As shown in Fig. 13, the eight bits of sample width information are used as the input to an up/down counter. The circuit is initiated upon the reception of a trigger signal TRIGI. If an integrator sampling is not already in progress, the counter is LOADed with the specified gate width and the count-down clock enabled. The timing shown in Fig. 13 considers that a 1-μsec pulse has been specified. In this case, the counter counts down once, at which time the terminal count, BORW, is reached which terminates the SAMPLE L pulse and disables the counting clock. When the processing of the current set of data is completed, that is, the new data is stored in the FIFO buffer, an internal reset, RSTI, is generated and used to enable further triggering of the system internal circuitry. The complete cycle time for the processing electronics is approximately 400 μsec from the end of sampling to the FIFO full condition. The triggers used by the OAS Integrator/ADC are received and multiplexed initially by the Trigger Multiplexer Board (LEA 78-1216) located on the rear panel of the OAS LSI-11 and routed via a KWV11-A programmable clock. Thus, under software control, one of four triggers may be selected and be given a programmable time delay of anywhere from 1 μs to approximately 65 ms before being sent to the Integrator/ADC.

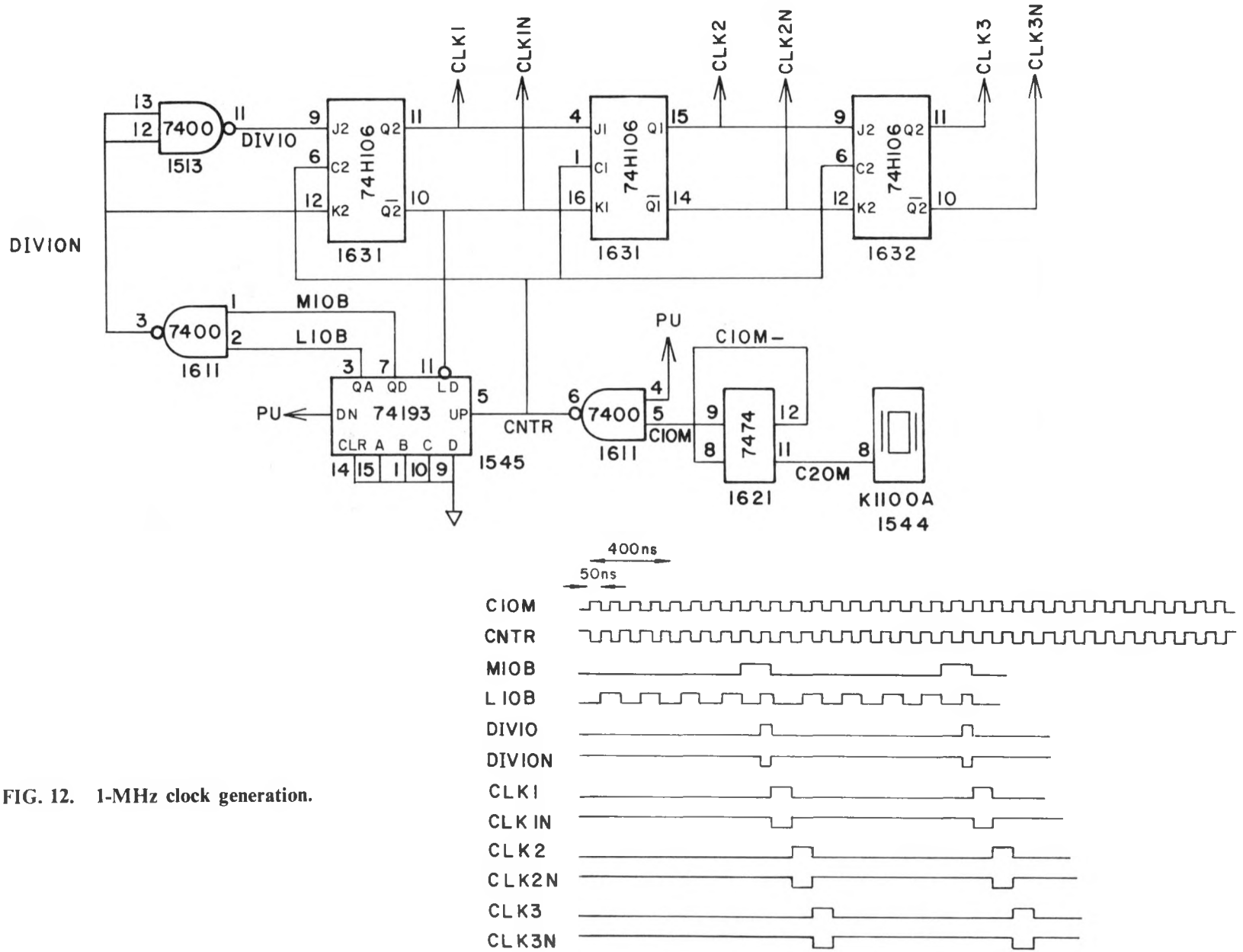


FIG. 12. 1-MHz clock generation.

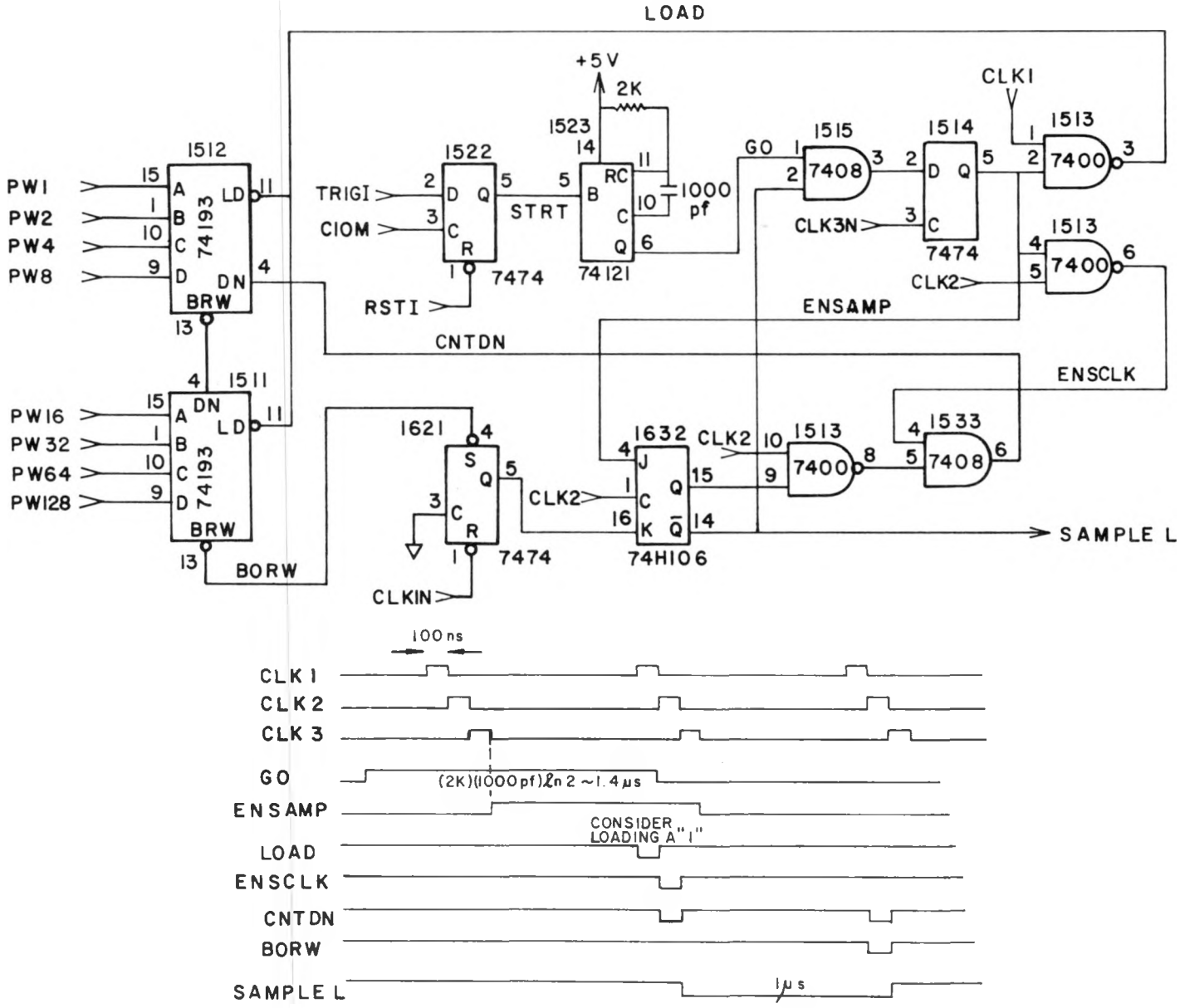


FIG. 13. Generation of programmable sample width.

Analog Multiplexing and A/D Converter Timing

The output of each integrator is connected to an input of a 16-channel analog multiplexer, the output of which is a 12-bit A/D converter. At the end of each integration period (SAMPLE), the multiplexer is sequenced and each channel converted to digital format and stored in a FIFO buffer. Figures 14 and 15 illustrate the circuits used to perform these functions.

The throughput of the A/D converter is given in the data sheets (DATEL ADC-M12B2B1) as 13 μsec; it was convenient, however, to convert at a 39-KHz rate without any degradation in performance. (The conversion must be fast enough so that the last data to be converted does not droop appreciably.) Since the signal format of the converter was such that a low state initiated conversion while a high state reset the device, it was necessary to ensure that, with the free running 39-KHz clock, the conversion began in a high state following the end of the SAMPLE period. Thus, as shown in Fig. 14, the 39-KHz clock is sampled and a decision made on its state so that the start conversion pulse (CONV L) to the A/D always starts in a high (reset) state. At the end of each conversion, the A/D returns an end-of-conversion pulse (EOC L) which is subsequently used to increment the analog multiplexer channel. When all 16 channels are completed (note that only 8 are actually used), the counter incrementing the multiplexer channel reaches its terminal count and generates an internal reset (RSTI) pulse so that the unit is ready for the next trigger.

FIFO Buffer Operation and I/O Control

To store alignment data until a computer request, the Integrator/ADC uses a 9403 First-in First-out (FIFO) buffer memory which is an expandable fall-through type optimized for communication buffer applications. It is organized as 16 words by 4 bits and expanded horizontally in this application to 12-bit words in order to match the A/D converter.

Functionally the 9403 consists of three parts: input and output registers with parallel (or serial) inputs, control inputs and outputs for handshaking and expansion, and a 4-bit-wide, 14-word-deep fall-through stack with self-contained control logic.

Figure 16 shows the horizontal expansion scheme, and timing diagrams for several phases of operation. The circuit shown in Fig. 17, which generates control signals for the FIFO, is used with both the interrupt service and handshaking routines of OASIS in order to interface with the LSI-11.

The input register can receive data in either bit-serial or 4-bit parallel form, store data until it is sent to the fall-through stack, and generate and accept the necessary status and control signals.

In the configuration shown, parallel entry is performed as follows: a HIGH level on the PL input (EOC) loads the D_0-D_3 data inputs which forces $\overline{\text{IRF}}$ LOW, indicating "input register full." During parallel entry, the $\overline{\text{IES}}$ input should be LOW while the $\overline{\text{CPSI}}$ input may be either HIGH or LOW.

Transfer to the fall-through stack is initiated by a LOW level on the $\overline{\text{TTS}}$ input. If the top location of the stack is empty, data is loaded into the stack and the input register reinitialized. This initialization is postponed until PL is LOW again so that, when the $\overline{\text{IRF}}$ output is connected to the $\overline{\text{TTS}}$ input, FIFO action is automatic.

Data falls through the stack automatically, pausing only when it is necessary to wait for an empty next location. When the FIFO is empty after a LOW pulse is applied to $\overline{\text{MR}}$, the Output Register Empty ($\overline{\text{ORE}}$) Output is LOW. After data has been entered into the FIFO and has fallen through to the bottom Stack location, it is transferred into the output register, provided the "Transfer Out Parallel" (TOP) Input is HIGH, and the $\overline{\text{OES}}$ Input is LOW. As a result of the data transfer $\overline{\text{ORE}}$ goes HIGH, indicating valid data on the data outputs (provided the 3-state buffer is enabled). TOP can now be used to clock out the next word. When TOP goes LOW, $\overline{\text{ORE}}$ will go LOW indicating that the output data has been extracted, but the data itself remains on the output bus until the next LOW-to-HIGH transition of TOP transfers the next word (if available) into the output register, as explained above. During parallel data extraction, $\overline{\text{TOS}}$, $\overline{\text{CPSO}}$, and $\overline{\text{OES}}$ should be LOW.

Most conventional FIFO designs provide status signals analogous to $\overline{\text{IRF}}$ and $\overline{\text{ORE}}$. When these devices are operated in arrays, however, variations in unit to unit operating speed require external gating to assure all devices have completed an operation. The 9403 incorporates simple but effective "master/slave" interlocking circuitry to eliminate the need for external gating (Table 1).

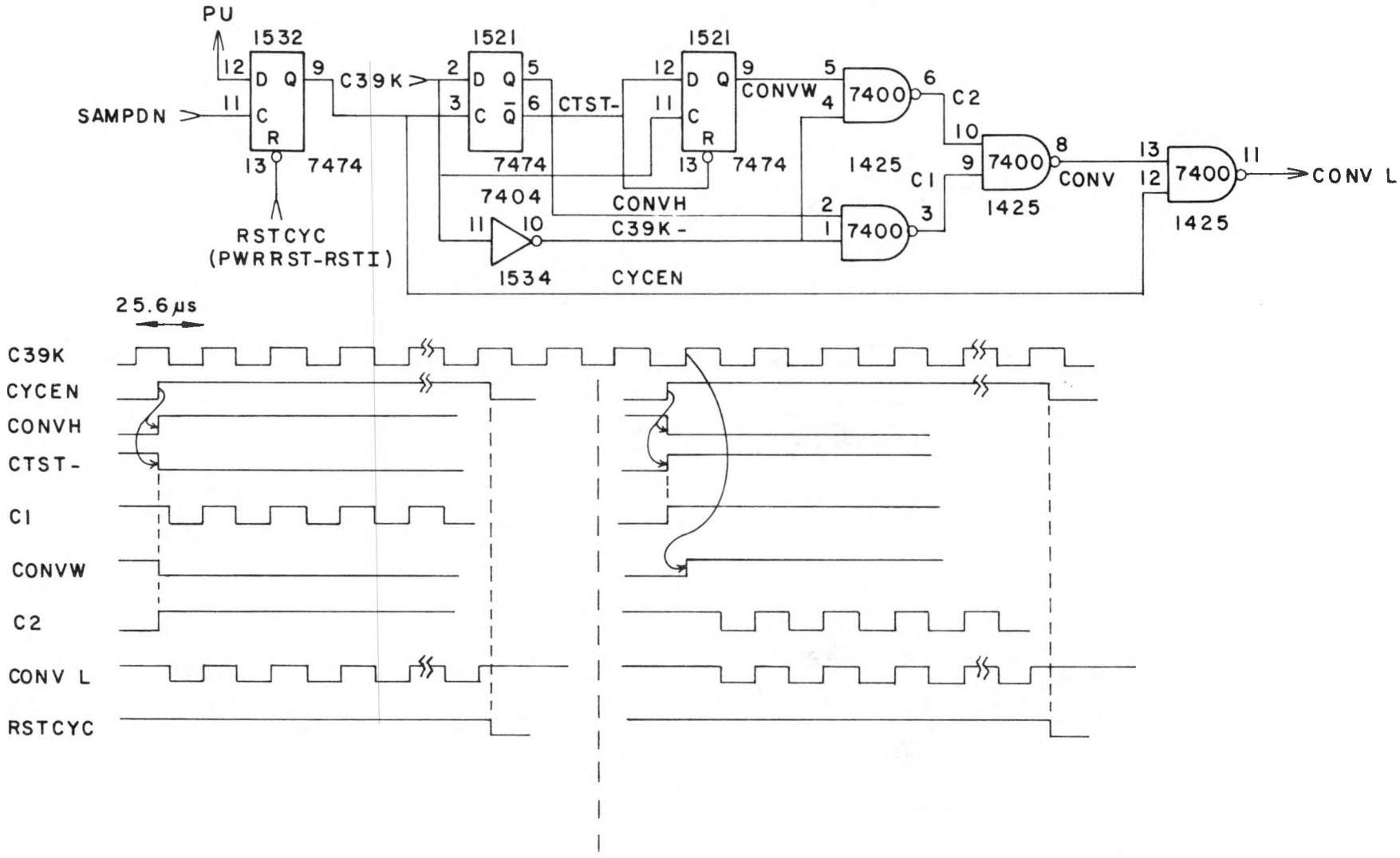


FIG. 14. A/D conversion control circuitry.

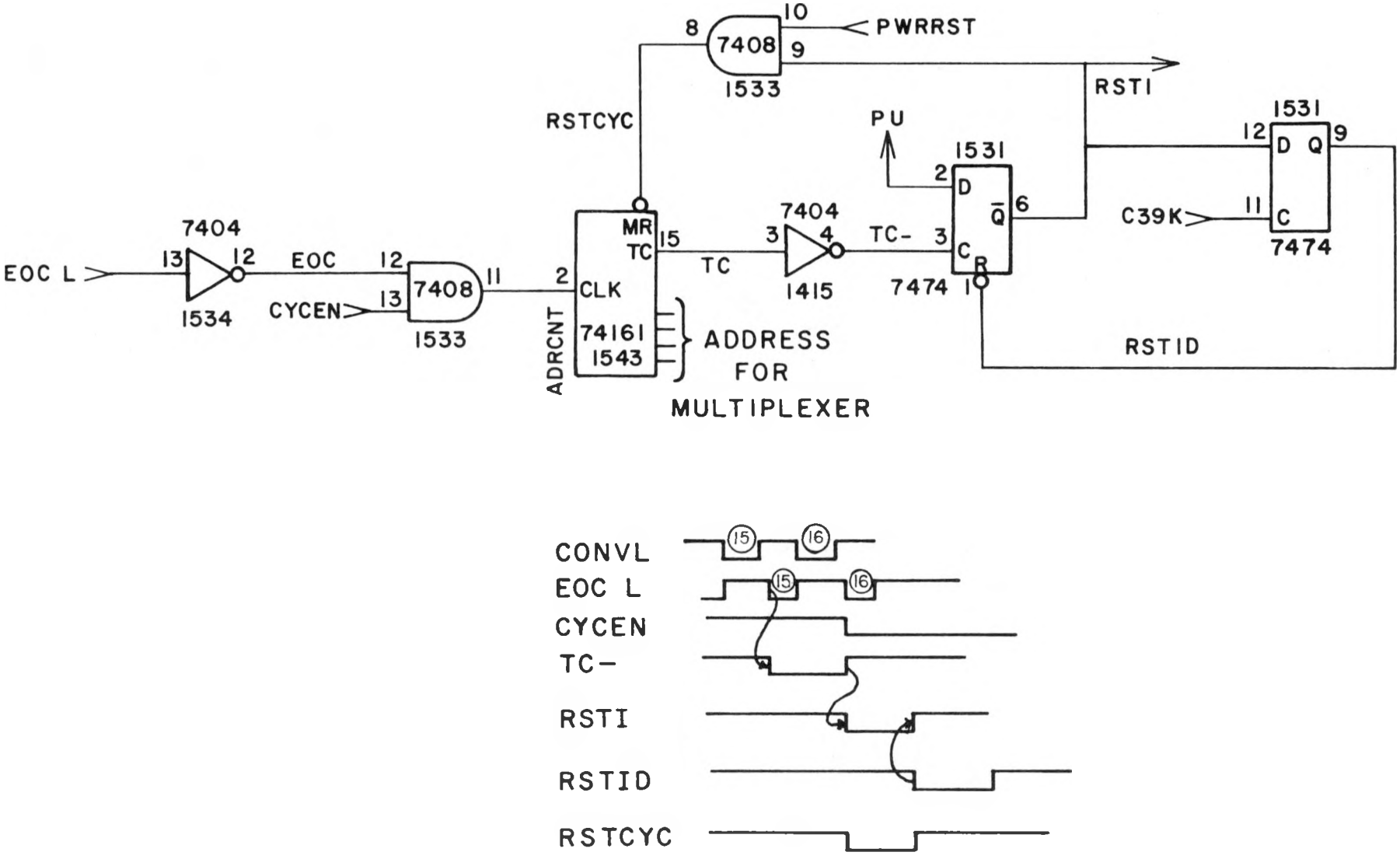


FIG. 15. Multiplexer sequencing and system reset.

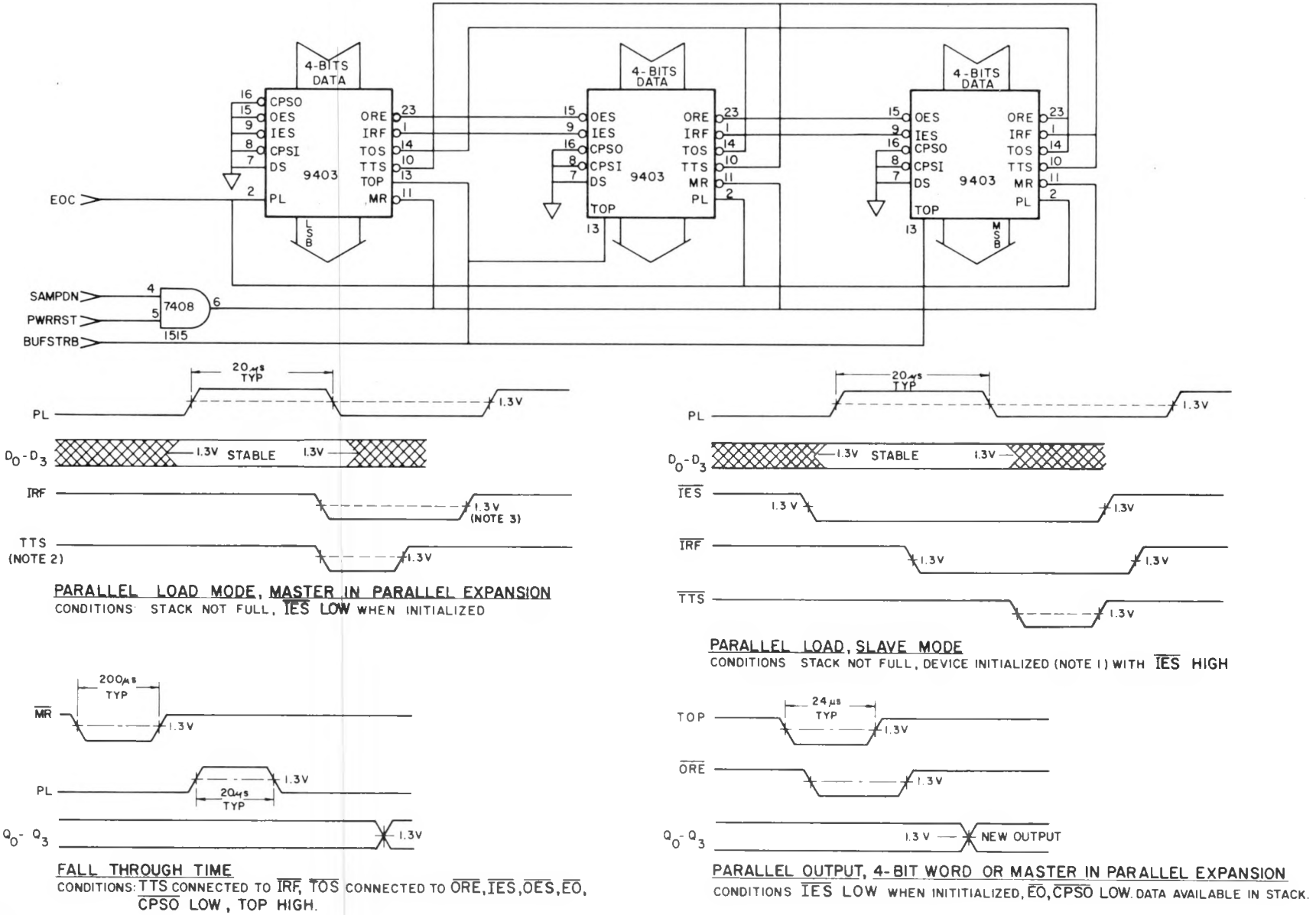
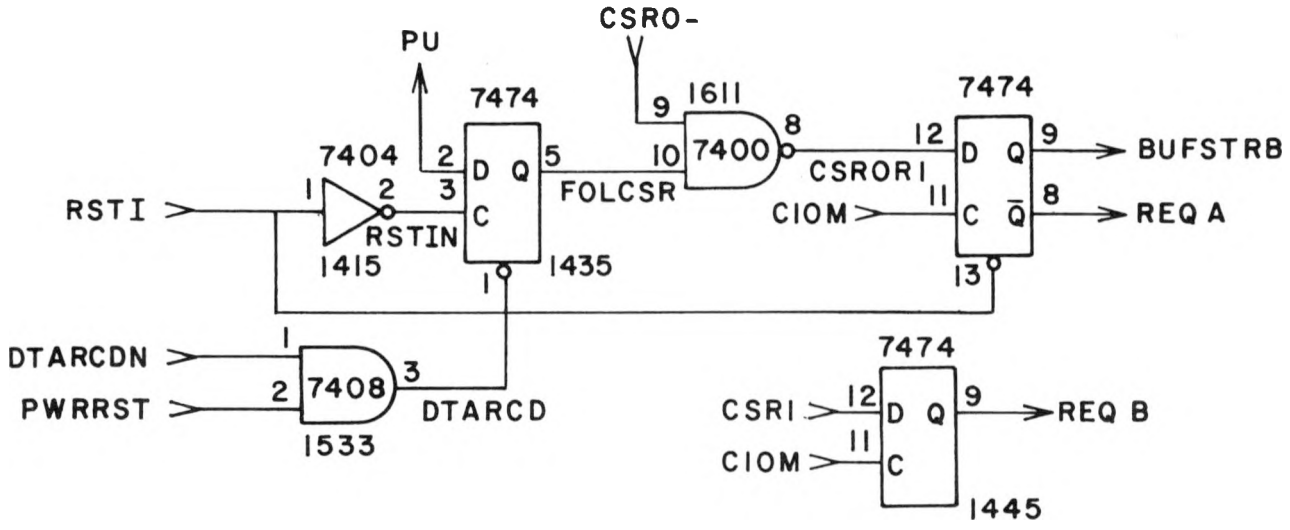


FIG. 16. FIFO buffer internal operation.



NOTE: THIS CIRCUIT IS USED WITH BOTH THE INTERRUPT SERVICE AND HANSHK ROUTINES IN OASIS.

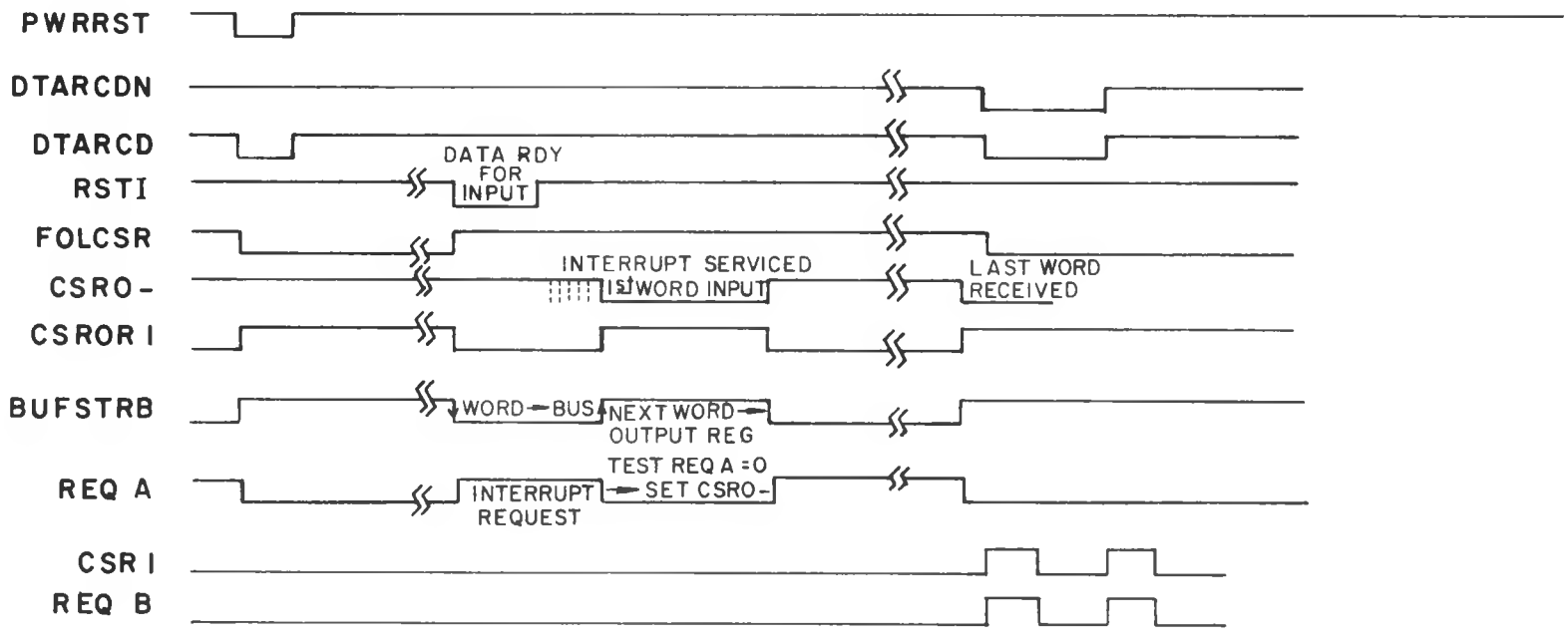


FIG. 17. FIFO buffer control.

In the 9403 array shown in Fig. 16, the leftmost device is defined as "row master" with all the other (2) devices being slaves. No slave will initialize its input register until it has received a LOW on its IES input from the row master or a slave of higher priority.

In a similar fashion, the $\overline{ORE}$ outputs of slaves will not go HIGH until their $\overline{OES}$ input has gone HIGH. This interlocking scheme ensures that new input data may be accepted by the array when the $\overline{IRF}$ output of the final slave in that row goes LOW, and that output data for the array may be extracted when the $\overline{ORE}$ of the final slave in the output row goes HIGH.

The row master is established by connecting its $\overline{IES}$ input to ground; a slave receives its $\overline{IES}$ input from the $\overline{IRF}$ output of the next higher priority device. When an array of 9403 FIFOs is initialized with a LOW on the $\overline{MR}$ inputs of all devices, the $\overline{IRF}$ outputs of all devices will be HIGH. Thus, only the row master receives a LOW on the $\overline{IES}$ input during initialization.

Whenever $\overline{MR}$ and $\overline{IES}$ are LOW, the master latch is set. Whenever $\overline{TTS}$ goes LOW the request initialization flip-flop will be set. If the master latch is HIGH, the input register will be immediately initialized and the request initialization flip-flop reset. If the master latch is reset, the input register is not initialized until $\overline{IES}$ goes LOW. In array operation, activating the $\overline{TTS}$ initiates a ripple input register initialization from the row master to the last slave.

A similar operation takes place for the output register. Either a $\overline{TOS}$ or TOP input initiates a load-from-stack operation and sets the $\overline{ORE}$ request flip-flop. If the master latch is set, the last Output Register flip-flop is set and $\overline{ORE}$ goes HIGH. If the master latch is reset, the $\overline{ORE}$ output will be LOW until an $\overline{OES}$ input is received.

TABLE 1. Summary of master/slave status outputs.

OUTPUT CONDITION	INTERNAL STATE	
	Master Operation–IES Low when Initialized	Slave Operation OES HIGH when Initialized
IRF LOW	Input Register Full	Input Register Full and IES LOW
ORE HIGH	Output Register not Full	Output Register not Full and IES OES LOW

The circuit which, together with the OASIS routine, provides the handshaking between LSI-11 and Integrator/ADC is shown in Fig. 17. Output data from the LSI-11 is simply strobed into Integrator/ADC registers by REQB; the LSI-11 sets up the output data, sets CSR1, and checks for REQB before resetting CSR1.

When the FIFO buffer is full, the data acquisition cycle of the Integrator/ADC will be completed and RSTI will be reset. This causes REQA to be set and, thus, the interrupt service routine of OASIS to be entered. BUFSTRB, which clocks data out of the FIFO, simply follows CSR0. Since REQA is BUFSTRB, the software can check for REQA = 0 before resetting CSR0. When all 16 words have been transferred to the LSI-11, DTARCDN is sent. This sets BUFSTRB (thus resetting REQA), reinitializing the cycle for the next RSTI.

GROUNDING AND SHIELDING

Grounding and shielding are among the most important issues to be addressed in the design of any electronic system. In the case of a control system that resides in an extremely hostile noise environment and in which even small offsets contribute to significant degradation of system performance, particular attention must be paid to the system grounding scheme.

The design of the OAS electronics and, in particular, of the Integrator/ADC package, was based on the philosophies of single point grounding to eliminate noise-ridden ground loops, and shield-within-a-shield to combat radiated electrical and magnetic noise. As illustrated in Fig. 18, great care was exercised to maintain a single point ground configuration both inside the Integrator/ADC and in interfacing with other devices.

The single point ground was defined as a large surface area located on the Integrator/ADC board. The Common from each of the ±15 VDC and +5 VDC local power supplies were tied to this point as were the common signals from both the Analog Multiplexer and the 12-bit A/D Converter.

Each lateral-effect photodiode (four output signals each) serves as an input device to the Integrator/ADC; however, each of the input signals passes through a buffer amplifier and is subsequently connected to the Integrator/ADC, EACH INPUT WITH ITS OWN REFERENCE OR COMMON SIGNAL. Each signal then serves as an input to a unique integrator with its reference or common connected to the single point ground.

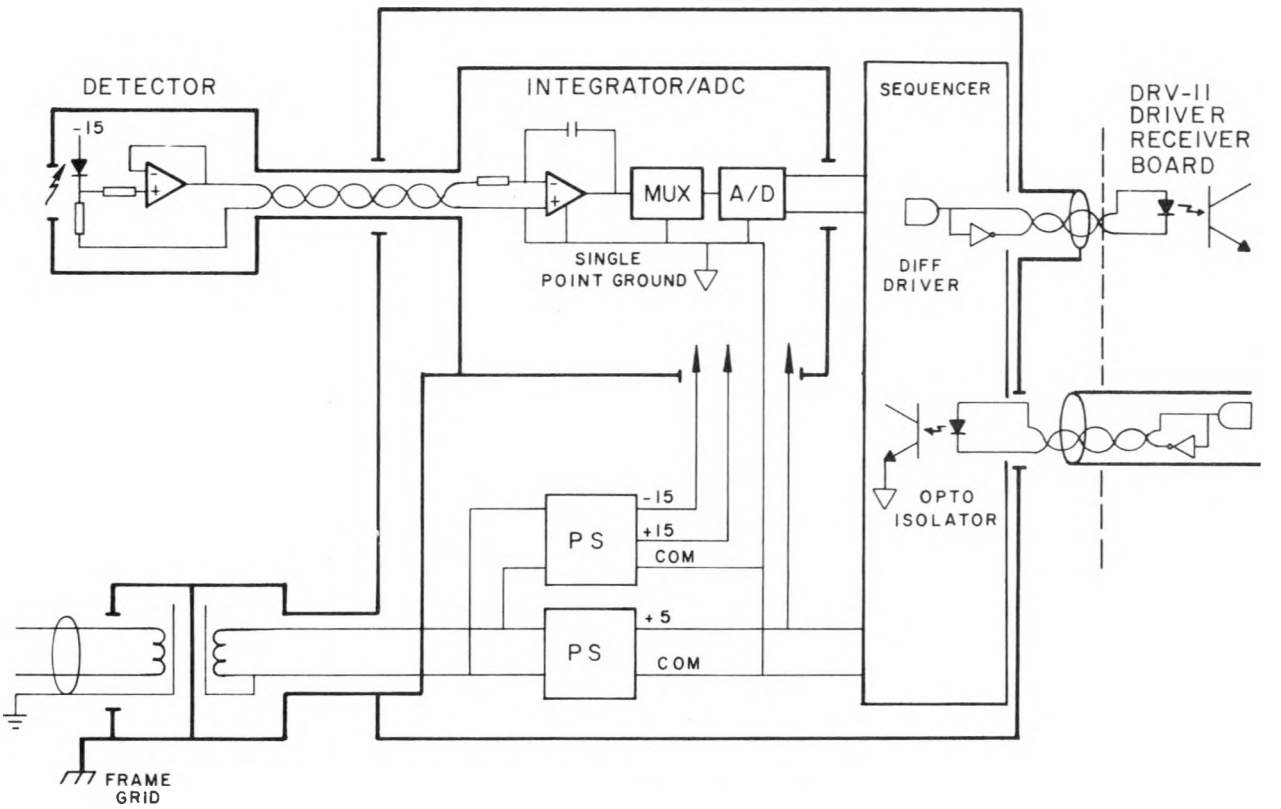


FIG. 18. Internal grounding and shielding—integrator/ADC.

There is a shielded, twisted-pair cable connection to each of the detectors. The shields are connected to ground at the Integrator/ADC end of the cable only, by way of a specially designed copper shield which serves as the inner shield of the shield-within-a-shield, completely enclosing the circuitry on the Integrator/ADC board. This shield is connected to the frame grid at one point only. Figure 18 shows Integrator/ADC package, illustrating, in particular, the shield enclosing the analog circuitry.

Special attention was also paid to the interface with the LSI-11. Since it was extremely undesirable to have any electrical noise from the LSI-11 couple into the data acquisition system proper, the DRV11 Driver-Receiver (LEA 77-1257) was designed to isolate the Integrator/ADC from the LSI-11. In fact, this card can be used to isolate any user device which communicates with an LSI-11 via the parallel interface DRV11 (as shown in Fig. 19). All signals to be input to the LSI-11 are differentially driven from the Integrator/ADC and input to a DRV11 via optical isolators on the Driver-Receiver card. Similarly, all signals being output from the DRV11 are differentially driven by the Driver-Receiver card and received by optical isolators in the Integrator/ADC. Thus, the data acquisition system is totally isolated from the LSI-11 and noise coupling between the two is minimized.

In addition to the grounding philosophy, there is a system consideration which has proven to be invaluable in terms of system diagnosis: the Integrator/ADC was designed with an internal test feature, software initiated, which ensures that this system component has executed its data acquisition functions properly. This feature has contributed to minimal system down time and increased reliability.

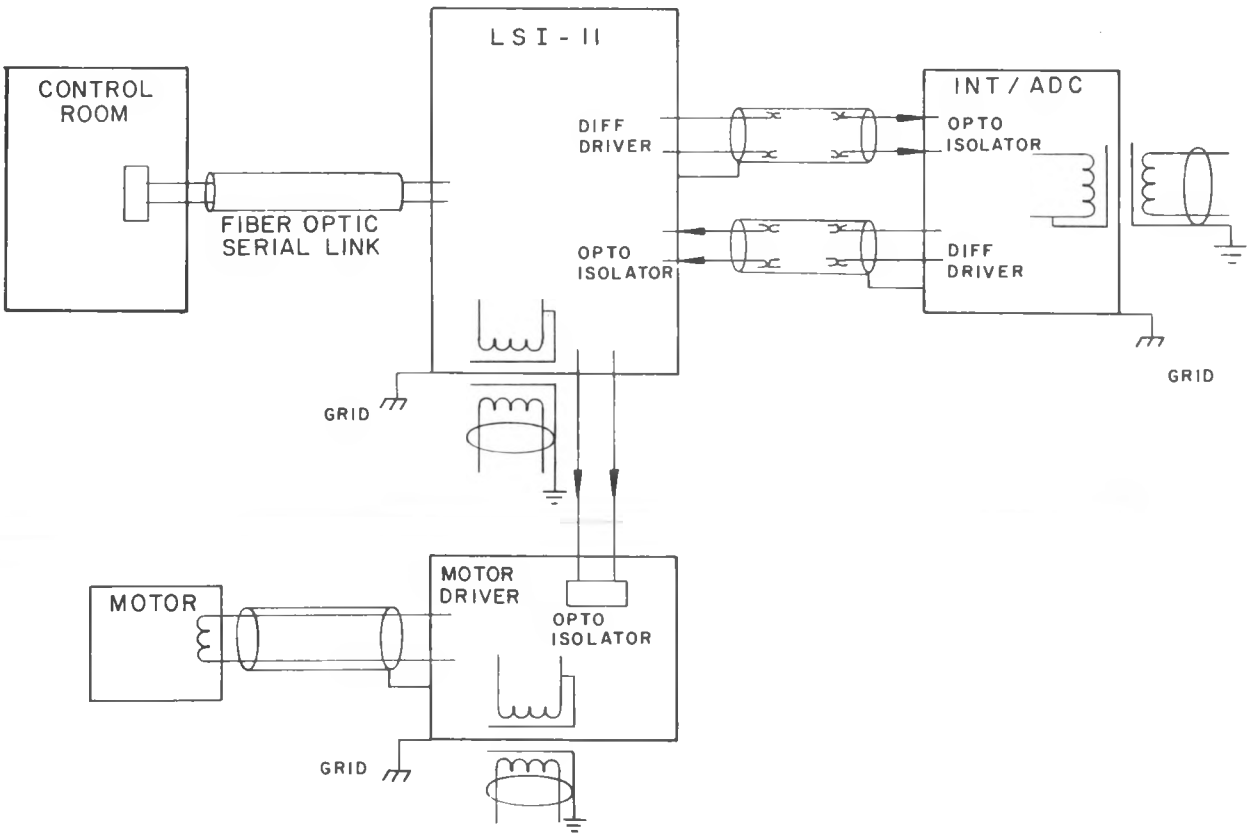


FIG. 19. Grounding and shielding configuration—external devices.

VI. COMPUTER INTERFACE AND CONTROL

The Oscillator Alignment System is based on an LSI-11 microprocessor which is used to monitor and control all phases of system operation. It is configured with 28 K of core memory, floppy disk, terminal, and several special purpose interface and control modules. These modules include:

- Stepping motor logic cards for interface between the LSI-11 and stepping motor drive circuitry.
- Motor clock and DRV11 parallel interface (also for motor interface).
- Control panel interface.
- KWV11-A programmable clock used for delaying trigger signals.
- Parallel interface (DRV11) and optical isolation card for interfacing the Integrator/ADC package.
- Serial interface DLV11 for network fiber optic link to the control room.
- A power-fail detect module used in auto-booting and (eventually) down-line loading operations.

The LSI-11 hardware configuration is shown in Fig. 20.

Extensive software was necessary to support this diversity of hardware. The control program for OAS is written in REBEL/BASIC organized into two sections ("Users"): the first section performs all calculations, interfacing, and display and control functions; the second section or "User" is dedicated to network communication with the control room via SHIVANET. In general, interface functions are performed via calls to assembly language routines while control calculations are done in BASIC. Routines for driving stepping motors (MOTORS) and the control panel (PANEL) are described in detail in the LLL Stepping Motor Processor Manual (UCID-18701). Interface with the Integrator/ADC package is through the routine OASIS, through which the integrator sampling width, trigger selections, and (test) mode may be specified. The routine DELAY allows the specification of a programmed delay, via the KWV11-A clock module, between the actual system trigger and the initiating trigger signal sent to the Integrator/ADC.

The following sections include a detailed description of the OASIS and DELAY modules, along with a complete, yet limited, description of the control program. Due to extensive effort in documentation, this program is best understood by reading the source itself.

INTEGRATOR/ADC INTERFACE—OASIS

The OAS LSI-11 communicates with the Integrator/ADC package via the assembly language routine OASIS. This permits the control program to specify system parameters such as integration time, test mode, and trigger selection as well as obtain alignment data which has been processed by the integrator package.

OASIS is organized into three sections: an initialization part in which the system parameters are set up before the Integrator/ADC is triggered; a subroutine (HANSHK) which handles the details of communication between LSI-11 and Integrator/ADC; and an Interrupt Service Routine which services an interrupt generated by the condition that the FIFO buffer in the Integrator/ADC is full (and, therefore, that the data is ready to be transferred to the LSI-11) by transferring the alignment data.

Operation

The address and vector for the Integrator/ADC interface DRV11 were chosen to be 167670 and 350, respectively. Only interrupt "A" is enabled.

The philosophy of the module OASIS is such that the parameters necessary to determine the state of the system are initialized before enabling interrupts. In addition, a reset pulse is sent to the Integrator/ADC to ensure that the hardware is properly initialized. Disabling interrupts while sending the initialization pulse solve the problem (inherent in DRV11's) of resetting REQ A while an interrupt is pending, which results in a bus error. Thus, the integration time, trigger selection, and (test) mode are specified whereupon the subroutine returns to the BASIC control program. When the Integrator/ADC has been triggered and its FIFO buffer

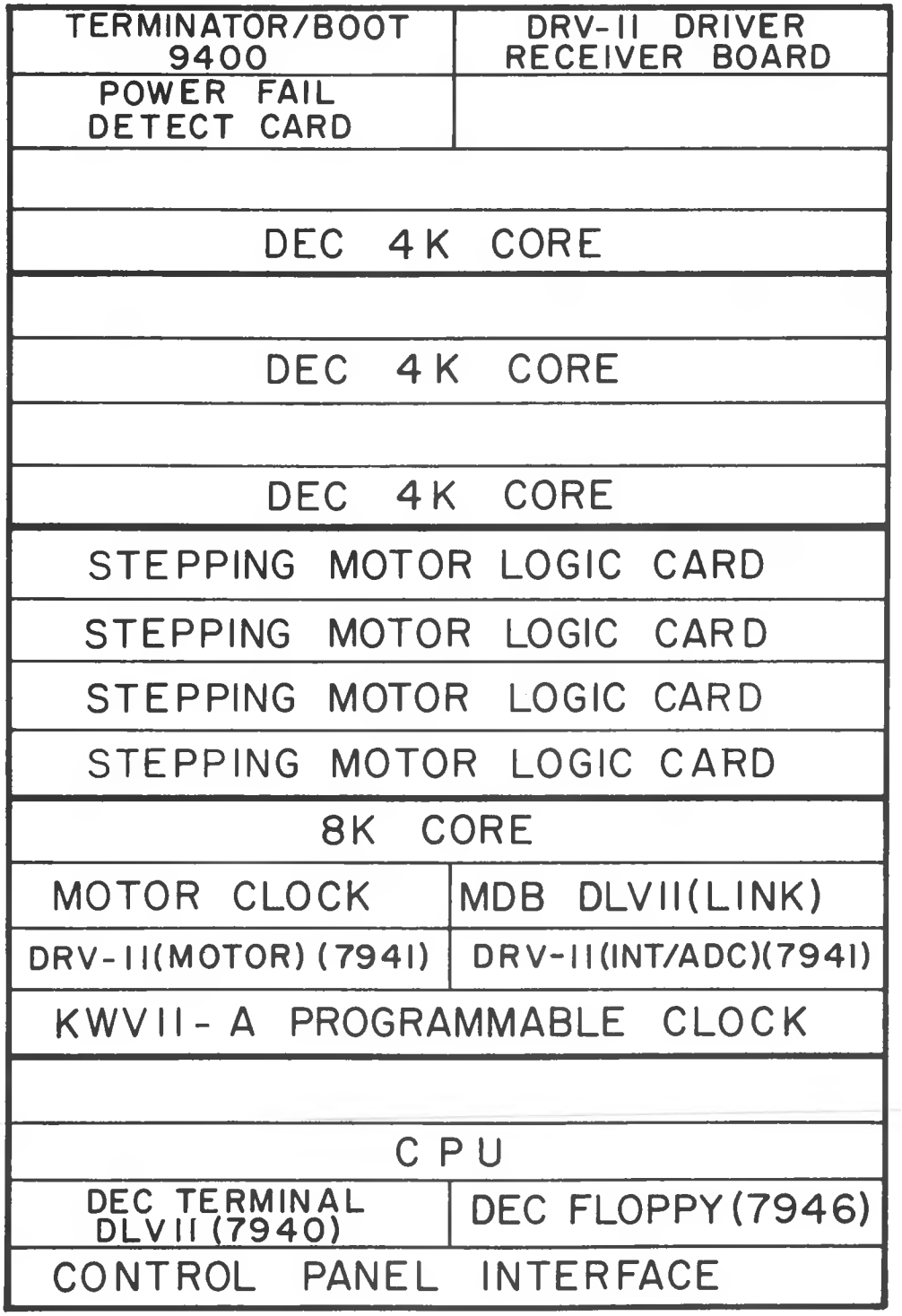


FIG. 20. Hardware configuration—LSI-11.

filled with data, an interrupt request is sent to the LSI-11. Service is via the Interrupt Service Routine in OASIS (entry point: OASINT) which transfers the data into memory.

The integration time is the amount of time for which the integrators are enabled and is specified by the first eight bits of the output word (LSB = 1 μs).

The OAS may be triggered by any of several (4) sources, whose selection is specified by bits 10 and 11 of the output word. These signals are physically connected to the Trigger Multiplexer located on the rear panel of the OAS LSI-11.

Alignment data for input to the LSI-11 is masked to 12 bits which is the extent of the A/D converter.

PROGRAMMABLE CLOCK INTERFACE—DELAY

The time delay (variable) needed to synchronize the gated integration of the laser signal with the signal itself is introduced in the triggering of the Integrator/ADC by a KWV11-A Programmable Clock, one of whose modes allows the time between input and output events to be varied under program control.

The main subroutine specifies the base clock to be used (1 MHz, in this case) along with the mode of operation, loads the Buffer Preset Register (BPR) with the two's complement of the desired delay, and returns to the calling program. Upon receiving an input signal, an interrupt is generated causing a counter to start and ultimately output a signal at the appropriate (specified delay) time.

The KWV11-A is an integral part of the OAS system; any trigger selected may be delayed for from 1 μs to 65 ms in increments of 1 μs. The signal path for any given trigger is illustrated in Fig. 21.

Trigger delays of 10 μs are avoided because of the settling time required in the integrator module analog switches.

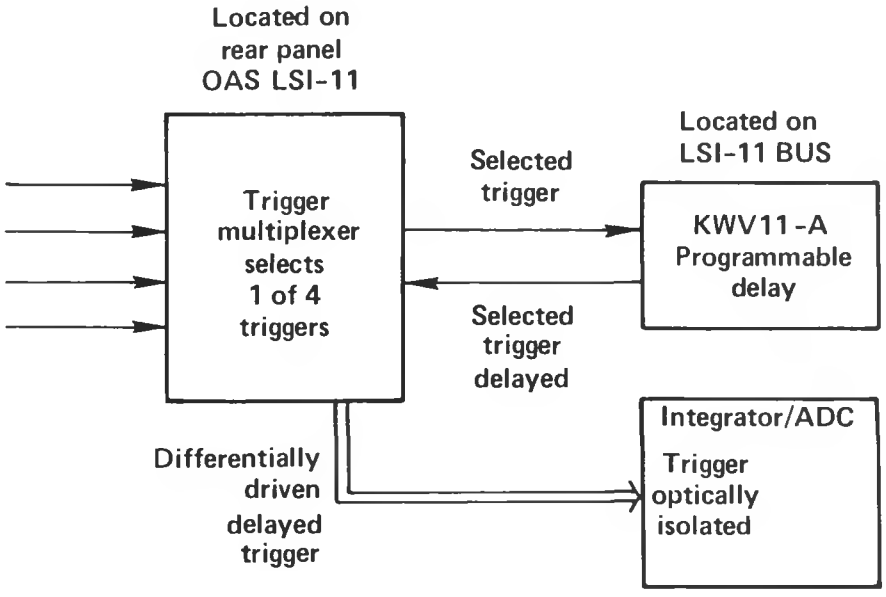


FIG. 21. OAS trigger signal path.

VII. OAS CONTROL PROGRAM

One of the most important parts of the Oscillator Alignment System is the control software, written in REBEL/BASIC, with its wide variety of functions, from performing calculations for closed-loop alignment to interfacing with an operator and displaying system status information. The combination of control program and special assembly language routines permits convenient access to stepping motors, control panel, Integrator/ADC, and, in fact, all other I/O devices in the system.

The control program is organized into three main sections: Initialization, Control, and Network Communication. The initialization section presets all variables and constants to be used in the remaining sections of the program. Operationally, it is loaded in from floppy disk, run, and deleted in order to allow as much space as possible for the remainder of the program.

The control program (also known as User #1) consists of several subroutines which perform specific system-operation functions including automatic control, manual control, test mode, sampling alignment data, adjustment of gain (attenuators), decoupling calculations, motor control, sampling integrator parameters (offsets), control panel servicing, and processing of commands from the control room.

The network communication section (User #2) handles all communication with the control room via calls to SHIVANET network routines. This section runs only when there is a network message and, even then, asynchronously from User #1. Requests for status or commands are treated separately. In the case of commands, Users 1 and 2 are interlocked to prevent the issuing of one command while another is being executed.

INITIALIZATION

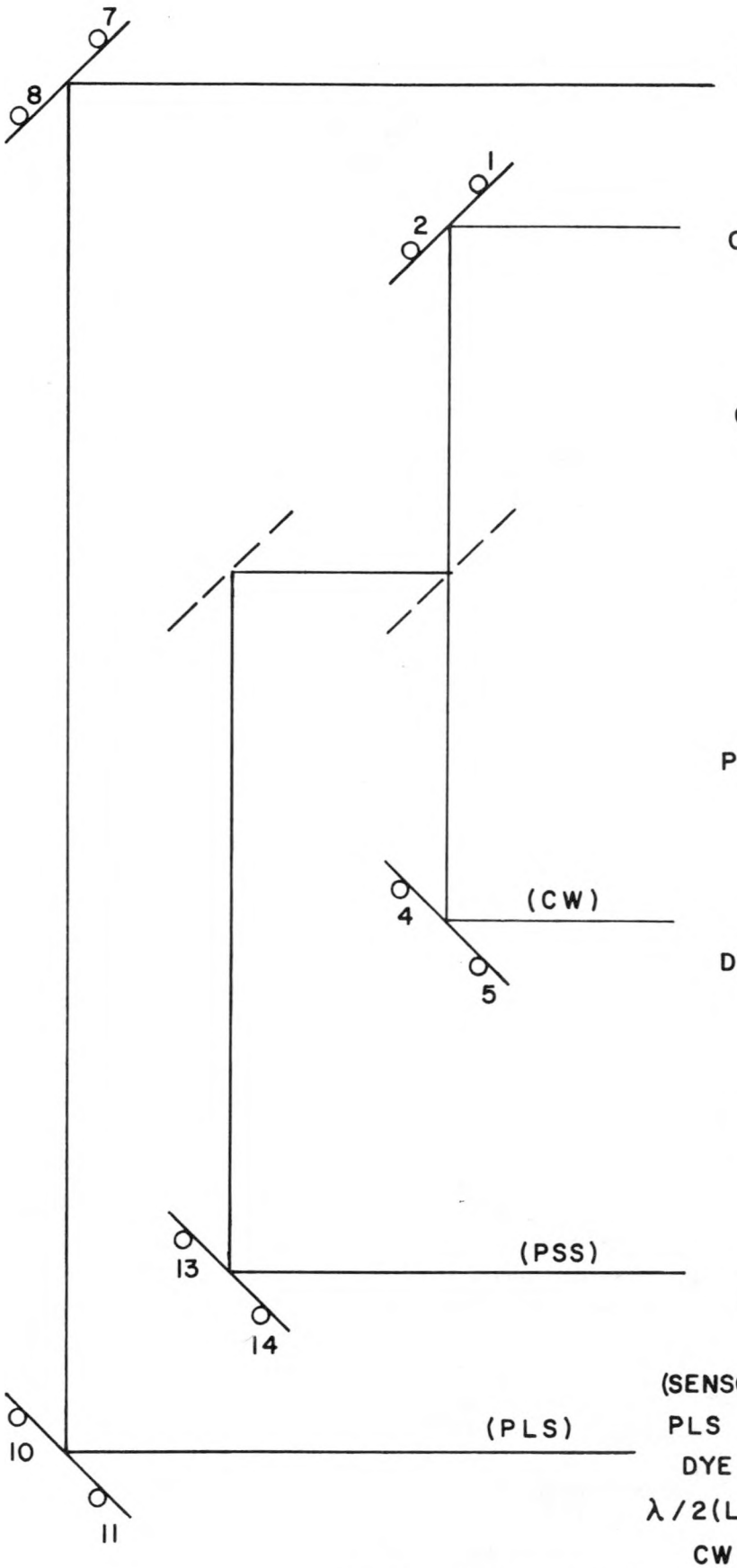
The first section of the control program initializes all the arrays, variables, and constants needed for proper system operation. Many of the arrays are indexed by a variable representing the alignment mode selected. The modes currently available are:

CW Long Path
CW Attenuator
CW Short Path
Pulse Oscillator
Pulse Attenuator
Dye Attenuator
Pulse Synchronization System

Most of the variables are described completely in the program source; waveplate offsets and motor assignments, however, warrant some additional explanation.

The motor position limit switches of the waveplates used in the OAS are not positioned (with respect to the crystal) at points corresponding to maximum and minimum attenuation. Furthermore, the relative position of the limit switches with respect to the crystal is different for each waveplate. As a solution to this problem, the variable WOFF represents the distance from a limit switch to the point of maximum (or minimum, depending on how the particular waveplate is used) transmission. Also, the CW attenuator is positioned such that its motor sense is opposite that of the other attenuators; that is, while all other attenuators operate through a range of positive motor positions, the CW attenuator operates through negative positions.

The stepping motor interface software permits the flexibility of assigning any logical motor number (MNUMB array) to any port on the Stepping Motor Drive Amp package. These assignments, done in the initialization section, reflect the motor-device correspondence shown in Fig. 22.



	MNUMB	MOTOR
J=0 CWLP	0	1
	1	2
	2	4
	3	5
J=1 CW ATN	4	7
	5	8
	6	10
	7	11
J=2 CW SP	8	1
	9	2
	10	4
	11	5
J=3 PLS	12	7
	13	8
	14	10
	15	11
J=4 PLS ATN	16	7
	17	8
	18	10
	19	11
J=5 DYE ATN	20	7
	21	8
	22	10
	23	11
J=6 AUX	24	7
	25	8
	26	10
	27	11
J=7 PSS	28	1
	29	2
	30	13
	31	14
(SENSOR) ATN	32	19
PLS ATN	33	20
DYE ATN	34	21
λ/2(LP/SP)	35	22
CW ATN	36	23

FIG. 22. OAS motor assignments.

The initialization is completed after all the system waveplates have been driven to a limit in order to preset their positions. Since there are no potentiometers or other position decoders on the motors, this action defines a position to which all subsequent motor movements are referenced.

CONTROL

The OAS control program allows direct system interaction via either the LSI-11 control panel or network messages from the control room. In addition, a variety of system status information can be displayed on the control panel readouts.

The main section of code consists of a very short loop which calls any of several subroutines depending on the present and desired state of the system. Since operator interaction was deemed a very important aspect of the system, the control panel service subroutine, which interfaces the LSI-11 control panel, is called whenever possible, to provide nearly continuous feedback to the operator.

A general flowchart for the control section (User #1) of the program is shown in Fig. 23. The state of the system is defined primarily by the alignment or attenuator mode selected, together with the control mode configuration, namely, whether the system is in RUN or STOP, AUTO or MANUAL, and NORMAL or TEST modes. The main control loop first performs a number of checks to determine the state of the system and then sets all gimbals and waveplates, based on that determination. If the system is in an attenuator mode, control is passed between the main loop and the gain setting subroutine only; all other operations are inhibited. When in an alignment mode, a second part of the main loop allows automatic or manual control of alignment. Except for important, subtle differences in calling order and subroutine entry points, both automatic and manual control involve sampling of alignment errors, applying a decoupling matrix, and moving gimbal motors to compensate for the sampled errors. An additional, and often invaluable, feature of the OAS is a test mode in which, under program control, the Integrator/ADC uses a voltage reference as an input test signal instead of signals from the alignment sensor. This provides a fast, easy check of an important system component, even when the system is on line.

There are four primary sections of code which are part of the main control loop: subroutine SELECT, AUTOMATIC CONTROL, MANUAL CONTROL, and TEST MODE (see Fig. 24 flowcharts). When the alignment or attenuator mode is changed, the SELECT subroutine closes the sensor shutter, sets status bytes to a "standby" value, and updates the local control panel by calling the front panel service routine.

Automatic control is performed by sampling alignment errors (via a call to the OASIS handler), decoupling centering and pointing errors, and giving appropriate commands to motors (via the motor subroutine) to correct the alignment errors. If the errors are found to be either out of range or within a "dead band," control is passed back to the main control loop without moving any motors.

Manual control includes a number of options regarding stepping motor control. Both GOTO and DECLARE ZERO functions are available under manual control; DECLARE ZERO specifies a current motor position to be a relative zero, while GOTO ZERO commands a motor to move to the position last declared as zero. These functions apply in both alignment and attenuator modes.

Control of the various OAS attenuators is also available under manual control. Alignment mode permits control of the sensor attenuator which controls the detector signal level. In attenuator modes, the sensor attenuator is closed, permitting control of the waveplate corresponding to the mode selected.

Finally, manual control permits the movement of gimbal motors, but not individually. For the control panel switches and the manual mode code, motor movements are decoupled error distances; that is, motors are moved in terms of pointing or centering errors rather than absolute motor positions. Control is eventually passed back to the top of the main control loop. In all manual motor operations, the speed at which the motors move may be selected (in terms of the number of steps moved) via the SLEW or STEP switches which specify 55 or 5 steps, respectively.

One of the most important features of the OAS is the test mode capability. Under automatic test, analog switches in the Integrator/ADC cause a voltage reference input to be provided for the integrators in

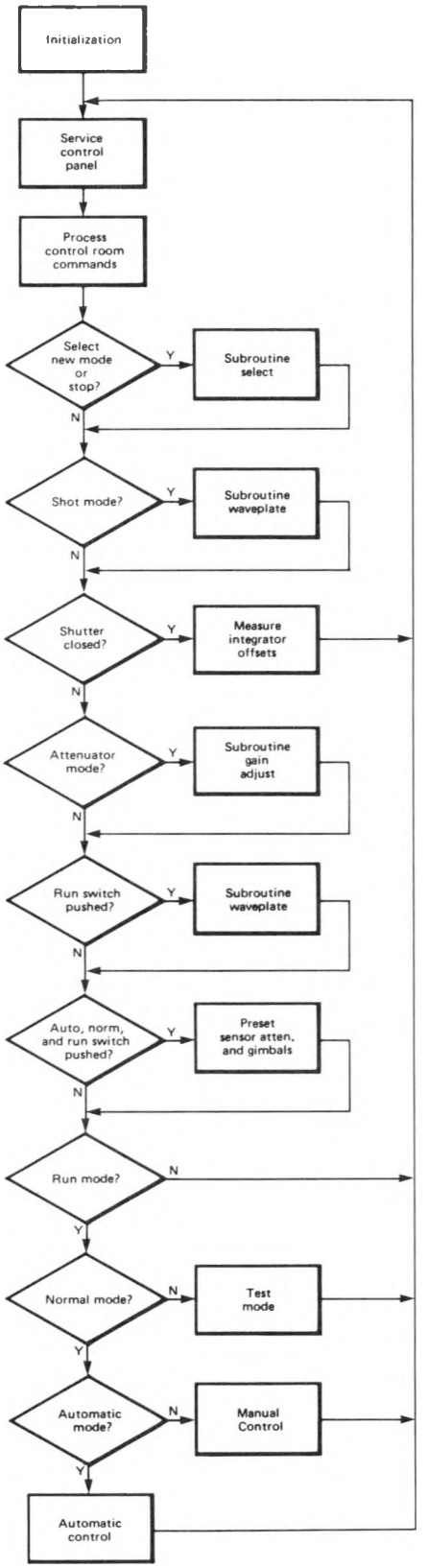


FIG. 23. Main control loop.

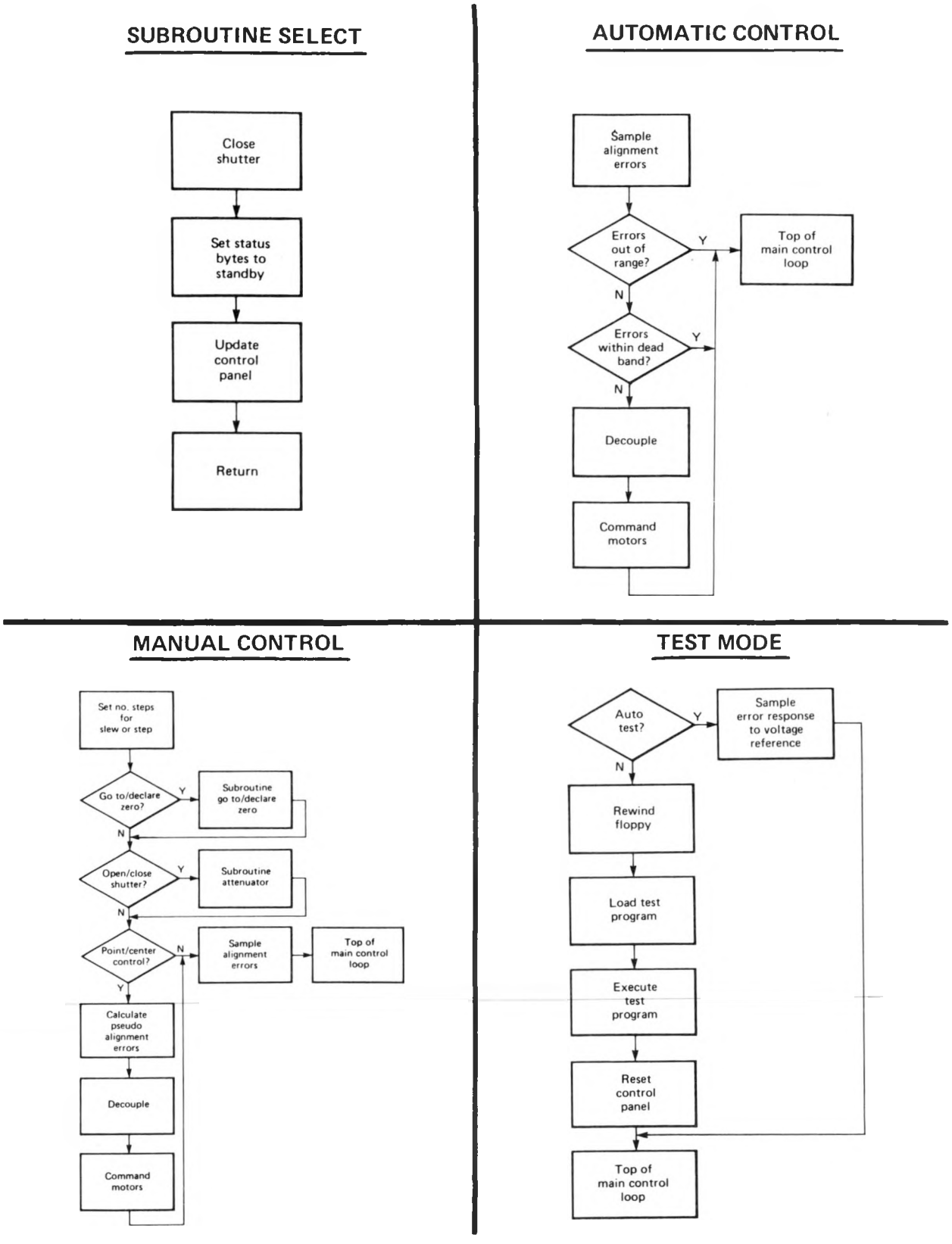


FIG. 24. Subdivisions of main control loop.

place of the normal detector signals. In this way, the response of the integrators can be checked for proper operation. In manual test mode, a test program stored on floppy disk is loaded into the LSI-11 and executed, providing the flexibility for testing new system developments.

Figure 25 describes the subroutine that samples alignment errors of the appropriate laser oscillator. In addition to retrieving data from the Integrator/ADC, the OASIS handler sets up parameters pertaining specifically to sampling time and trigger selection. By controlling the KWV11-A programmable clock, the DELAY routine provides a delay between the actual oscillator trigger and the triggering pulse used to start the Integrator/ADC operations.

Once the data is received from the Integrator/ADC, the noise (determined by means of the integrator offset subroutine which samples data while the shutter is closed) is subtracted, the signal intensity and alignment errors calculated, and the status array updated. If signal or alignment errors are out of range, a control panel light is turned on.

With the system in manual mode, the gain of any of several waveplate-polarizer pairs (attenuators) may be set from the local control panel or from the control room. As shown in Fig. 26, the subroutine determines which attenuator to control, calculates the distance to move the motor, and commands the motor to move. In these modes, if the DECLARE or GOTO ZERO switched are pushed, the attenuator either moves to the last declared gain position or stores the present gain position for future use.

After the motor is commanded to move, its status is checked. Motor power being on is taken to mean that the destination has yet to be reached, and the local control panel is updated. Eventually, the subroutines return to the top of the main control loop.

In any of the attenuator modes, all other functions are disabled and the control program operates exclusively between the main control loop and the gain adjust subroutine.

To move one or more of the stepping motors in OAS, the motor subroutine shown in Fig. 27 is called. All motors are indexed by the alignment or attenuator mode selected, so that each set of positions may be stored separately.

After the motor index is calculated, a check is made to determine whether the distance to move is extraordinarily large. If it is, the system response time is reduced by dividing the number of steps by a factor of five. This provides a means for preventing large alignment errors due to system malfunction.

Once the motors are commanded to move, a request for motor status is made, and then a check for motor power on. Power on is, again, taken to mean that the motor destination has yet to be reached, and the local control panel is updated.

When the motor reaches its destination, the relative motor positions and the status array are updated, and control is returned to the appropriate caller.

The OAS was designed to align any of several laser oscillators. To conveniently switch between them, waveplate-polarizer pairs (the waveplate being stepping-motor driven) were used to steer the beams. The waveplate subroutine illustrated in Fig. 28 moves each of the waveplates to the proper position, depending on the alignment mode selected.

The routine first gets the current status of the waveplate motors and then calculates the proper distance to move. In the case of SHOT MODE, which is defined as the PULSE ATTENUATOR MODE, the positions corresponding to the appropriate gain settings for the pulse and dye attenuators are calculated and the distance to move those motors determined.

After commanding the motors, the routine does the typical checking for motor power on, updating the local control panel and status array, and returning to the calling section.

To adjust the signal level for the OAS photodetectors, a waveplate-polarizer pair (attenuator) was used in the alignment sensor. This attenuator has the added feature of a shutter which closes completely when the attenuator is moved to its lower limit. To control the attenuator, the system must be in manual mode; the attenuator subroutine, which refers specifically to the sensor attenuator and is illustrated in Fig. 29, operates similarly to the waveplate subroutine previously mentioned.

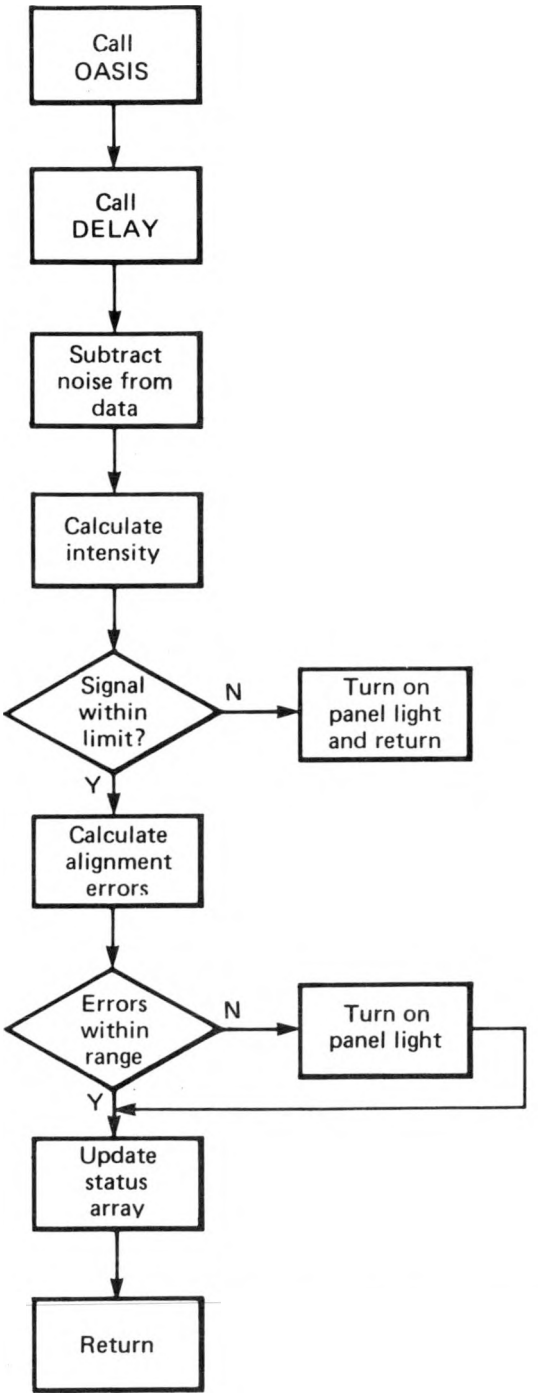


FIG. 25. Subroutine sample.

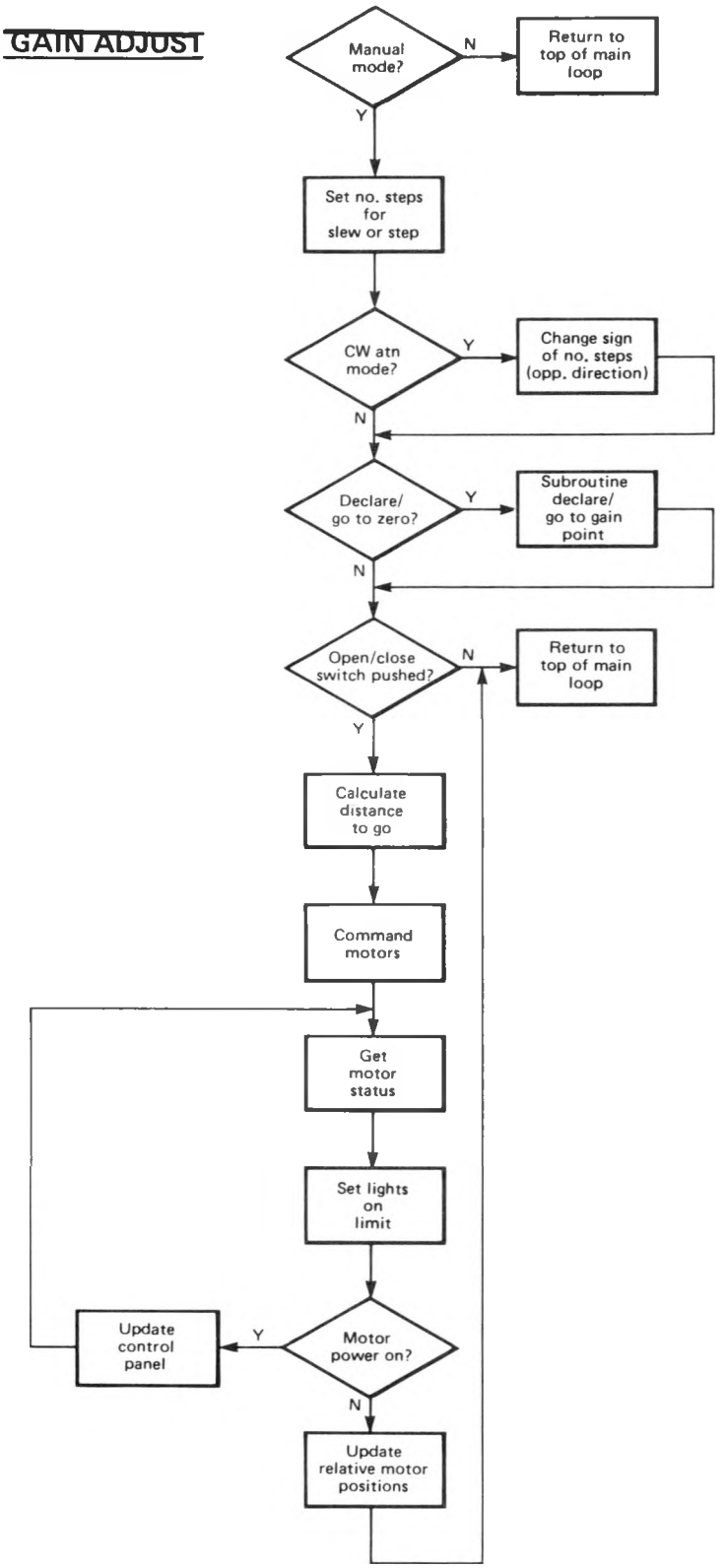


FIG. 26. Subroutine gain adjust.

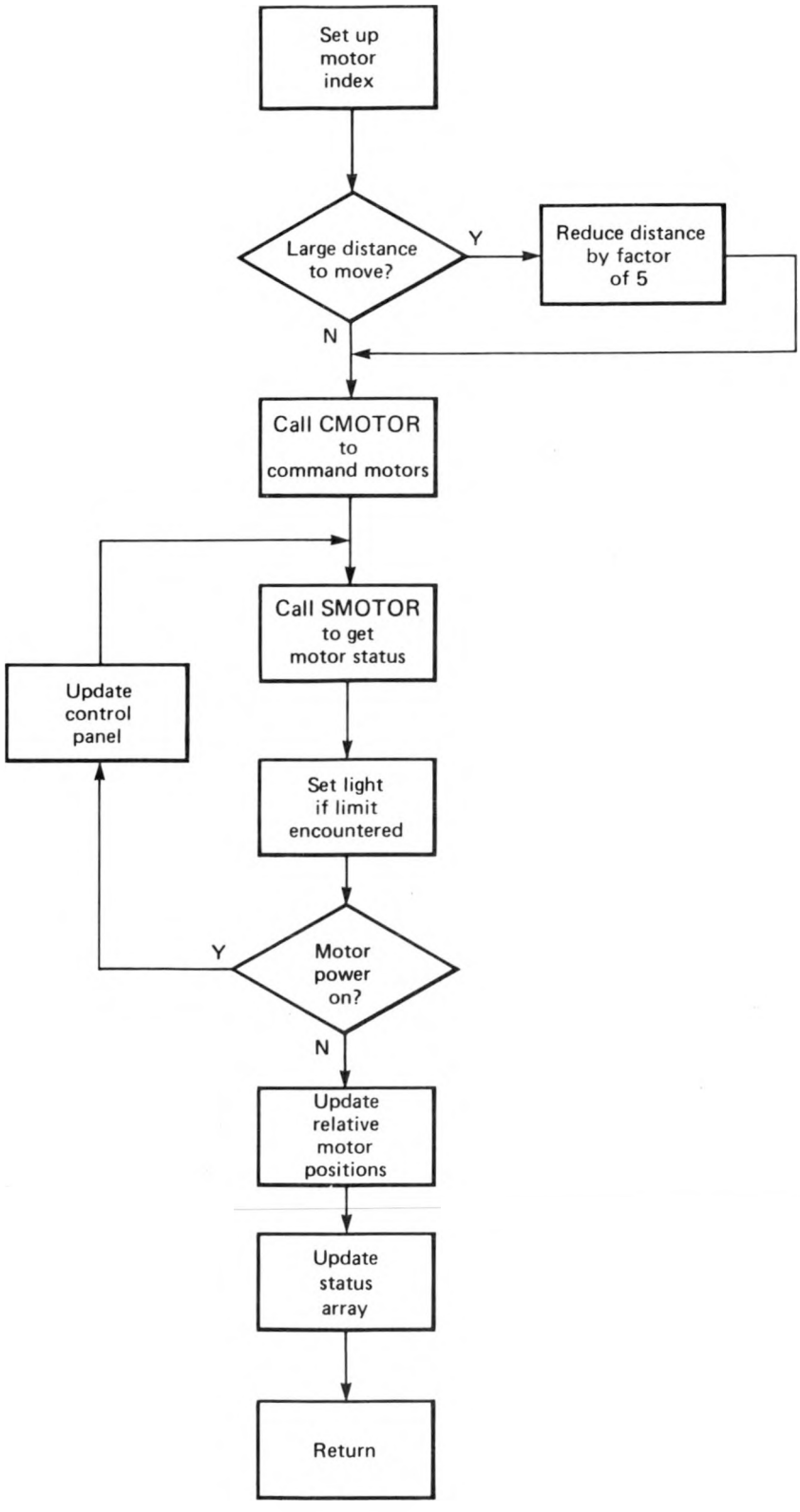


FIG. 27. Subroutine motors.

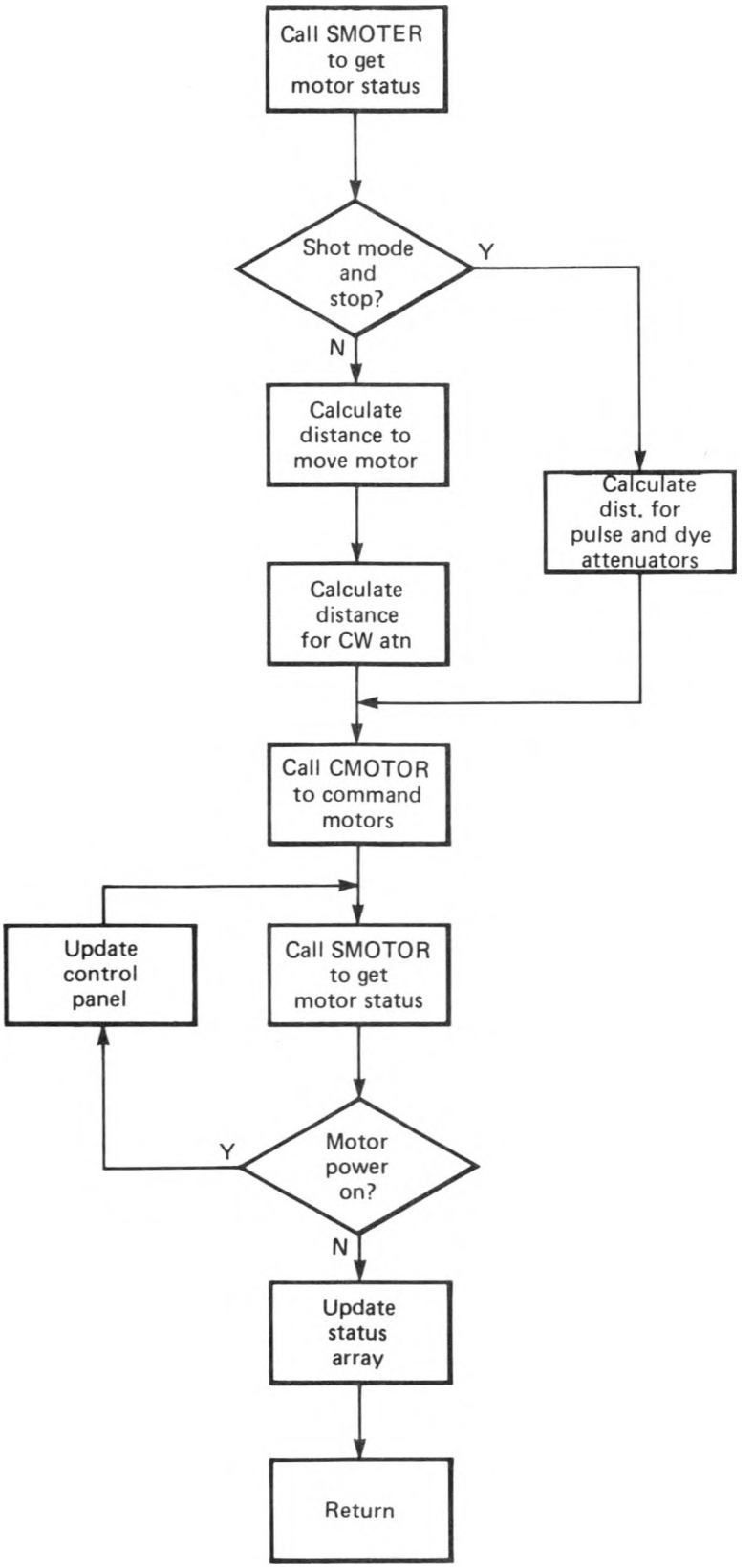


FIG. 28. Subroutine waveplate.

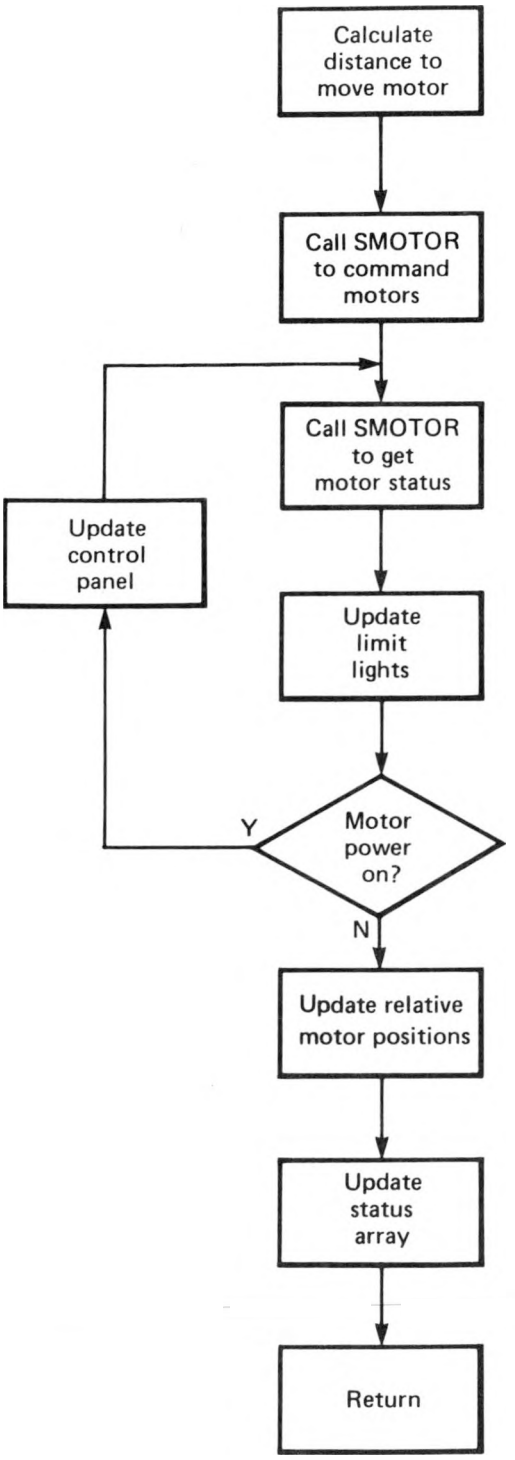


FIG. 29. Subroutine attenuator.

After calculating the distance to move the waveplate stepping motor, the command is sent via the assembly language MOTOR routine. As before, the motor status is requested and the local control panel updated while the motor power is still on. Subsequently, the relative motor positions and the system status array are updated, followed by a return to the calling routine.

As described in Section III, the OAS uses a two-gimbal arrangement in order to simultaneously correct pointing and centering alignment errors. The decoupling subroutine shown in Fig. 30 implements the equations derived in Section III.

After the alignment error data is read, the errors are adjusted by the proper magnification, which is determined by the system configuration and components. Decoupled motor distances are then calculated in order to correct the measured alignment errors.

To account for drift in the electronic components, AC coupling is essentially done in the software by subtracting off offsets from the alignment data. Integrator offsets are obtained via the integrator offsets subroutine shown in Fig. 30 by sampling while the alignment shutter is closed. A more detailed treatment of offsets and normalization is contained in the decoupling section.

Offsets are obtained by calling the routine OASIS while the shutter is closed. If the offsets are within prescribed limits, they are stored in an array; if not, an error code and status flag are set.

In many cases, it is desirable to have the capability of storing motor positions and moving to those positions automatically. The DECLARE/GOTO subroutines shown in Fig. 30 perform these tasks; one subroutine applies to gimbal and waveplate motors (including the sensor attenuator) in an alignment mode while the other applies to gain settings of attenuators in attenuator modes. Both subroutines operate similarly: after the motor status is obtained, a decision is made depending on whether the DECLARE or GOTO mode has been selected. In the former case, the present motor position is stored and the relative motor position is reset to zero. In the latter case, the distance from the desired position is calculated and calls made to move the appropriate motors.

Interfacing with an operator was perhaps one of the most important aspects of the OAS system design. It was necessary to provide means for manual or automatic operation, changing alignment and control modes, and monitoring and displaying many vital system parameters. Figure 31 shows the local control panel used in the OAS. A detailed description of its operation is given in the OAS User's Manual.

The actual hardware and software interface to the panel is by means of an LSI-11 Control Panel Interface Card (LEA 77-1252) together with the assembly language handler PANEL. However, the link between the control system and the panel interface is performed by the control panel service subroutine shown in Fig. 32. Since the update of the local control panel and thus the feedback to the operator was deemed to be of utmost importance, the control panel is serviced whenever possible. This includes situations where no other functions are being performed as well as those in which an event or function is waiting to complete.

Where the panel subroutine is entered depends upon whether the situation calls for reading the panel or merely updating, which is done when another operation is pending. After the initial panel update, the service routine checks whether there is a command from the control room (the network and communication interlock scheme is described in the next sections). If there is such a command, it is copied into the PBARRY, decoded, the interlock flag reset, and the network user restarted. Otherwise, the panel is read and the push button information stored in PBARRY. Thus, commands from the control room are essentially an emulation of the local control panel.

Next, the alignment or attenuator mode is calculated and the status array updated. If the system is in an attenuator mode, a significantly different procedure is followed than for alignment modes: only certain limited functions are permitted and the display functions are restricted to attenuator transmissions. In the case of an attenuator mode, the proper motor index is calculated, the limit and display control lights are reset, and the control mode variables (RUN, AUTO, etc.) are calculated. Should the system be in AUTO or STOP (in which case control of an attenuator will not be possible), the manual control lights will be reset, the appropriate data read from arrays, and the panel updated before returning to the caller. Otherwise, the SLEW/STEP value is calculated; the manual gimbal lights reset; the attenuator control variable calculated; the starting, present, and change of transmission calculated; and the panel updated before exiting.

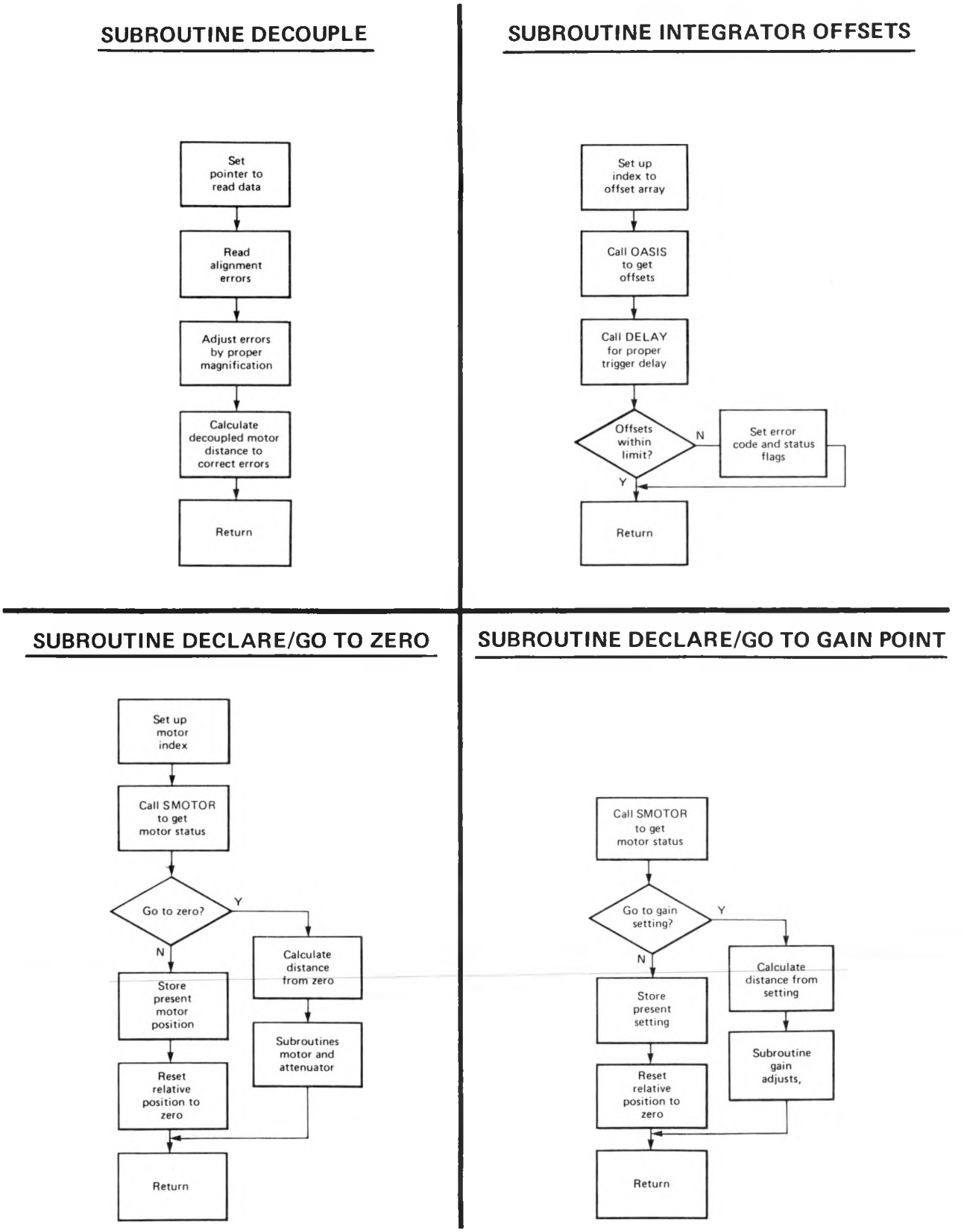


FIG. 30. Decoupling, integrator offsets, and DECLARE subroutines.

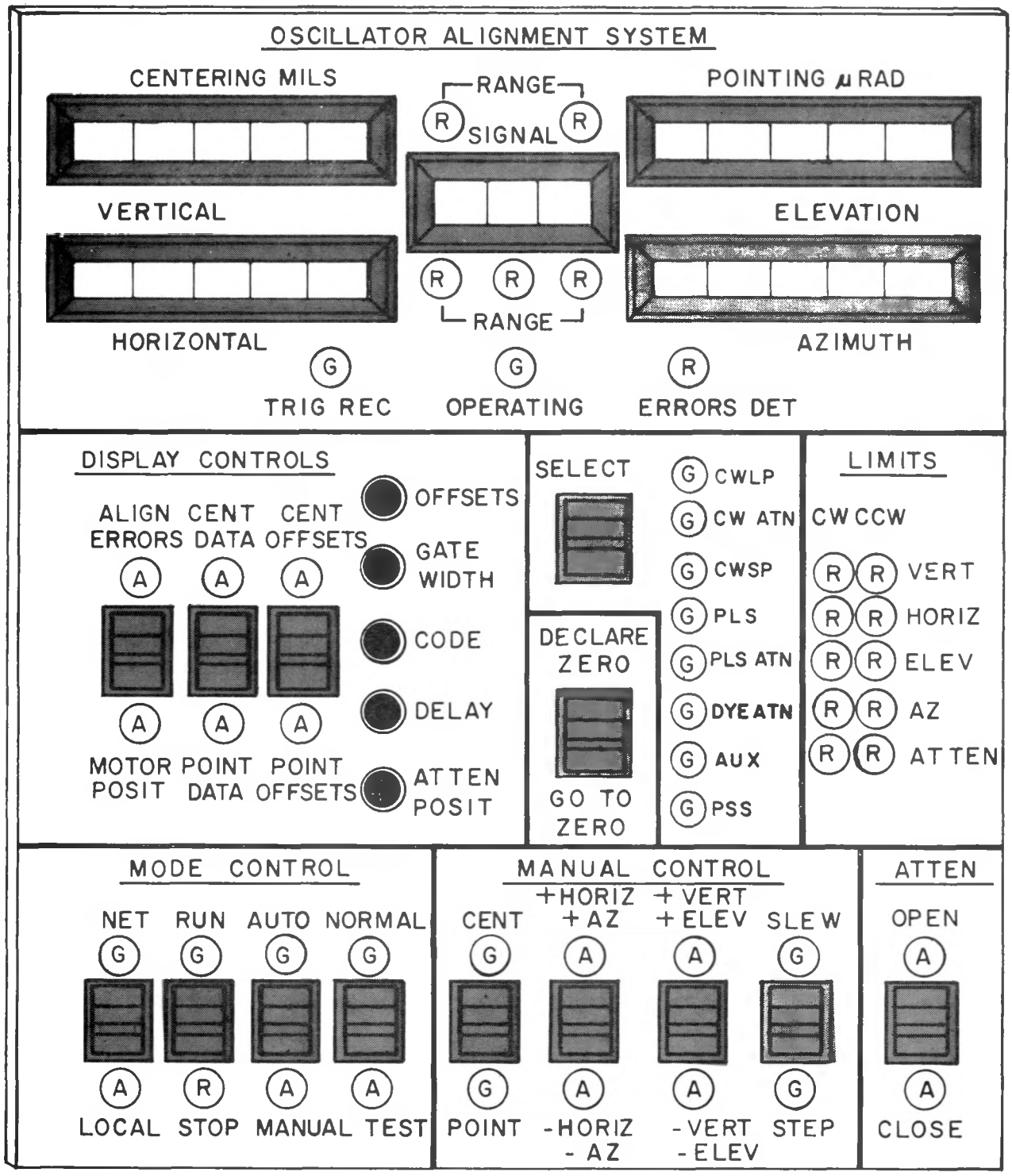


FIG. 31. OAS control panel.

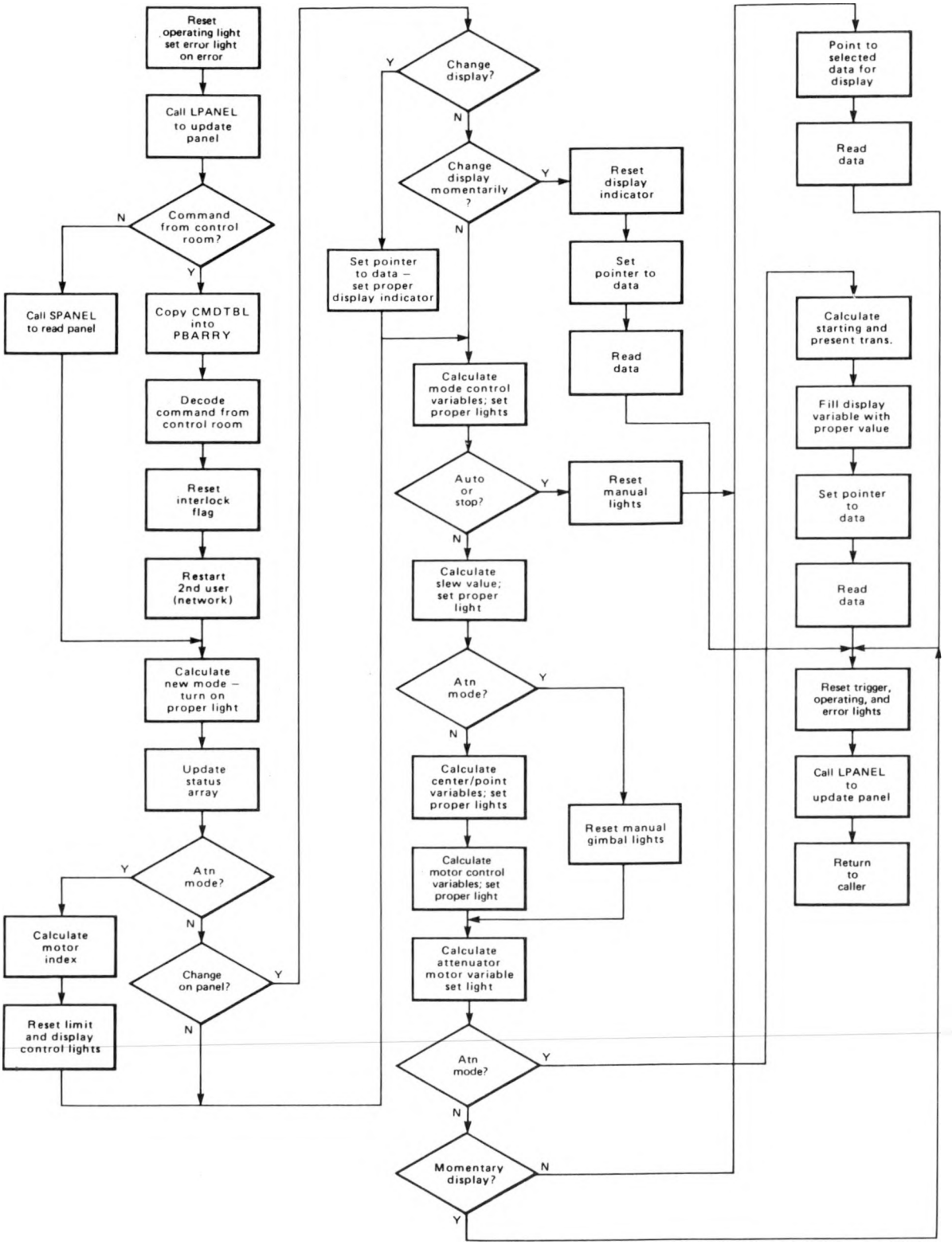


FIG. 32. Subroutine control panel.

In the case of an alignment mode, the panel is checked to see if changes have been made (i.e., if buttons have been pushed). Momentary displays are handled immediately by resetting the display indicator, reading the appropriate data, and updating the panel. The mode control variables are then calculated and the system checked to determine if it is in AUTO or STOP. If it is not, various manual control parameters are calculated (pointing/centering; slew/step; manual motor switches) before the data to be displayed is read and the panel updated. If the system is in AUTO or STOP, the manual section is bypassed; the data is simply read and the panel updated.

The OPERATING light on the panel is reset on entry and set on exit (with intervening calls to the panel) so that the apparent blinking is an indication of the panel service routine being executed.

NETWORK COMMUNICATIONS

The OAS can also be operated from the central control room via a plasma touch panel; the panel communicates with OAS over fiber optic links using the LLL-designed protocol SHIVANET. To handle network communications efficiently without degrading or inhibiting the performance of the control software, a scheme was developed in the OAS whereby a second User, running asychronously, takes care of all communication with the control room. Since both commands and status information must be exchanged between the control program and the network handler, the two are interlocked so that no new command may be initiated while another is yet pending. In addition, all parameters to be communicated to central control are passed in COMMON between the two Users.

Figure 33 illustrates the OAS communication interlock scheme. When there is no command from the control room, CMDFLG is zero. Thus the program receives command information from the local control panel via the SPANEL call.

Initially, since the OAS operates in master-slave mode, a read is posted for messages from the control room and the second (network) User suspended until a valid message is detected. After redefining the routing information so that the return message will be directed to the sender, the program checks whether the message is a command or status request. In the case of a status request, the appropriate data is copied into the sending array (SARRAY) and a call made to SLINK to return the information. The User then suspends until another message is received.

In the case of a command from the control room, the second User checks the value of CMDFLG. If CMDFLG is set, indicating that a previous command is being copied into the PBARRY (control program push button array), the second User suspends until that operation is complete and the control program resumes the network User. Note that the control program resets CMDFLG before resuming the second User and that the second User rechecks CMDFLG after resuming. This alleviates any problems of contact switching.

If the check of CMDFLG reveals that it is reset, the array received from the control room is copied into a command table for subsequent copying into PBARRY in the control User. The CMDFLG is then set, indicating that a new command has been received and loaded, and a response sent back to the control room before the network User again suspends.

This efficient scheme permits use of machine time for both control functions and network communications, since the network User is suspended unless there is a network message. As mentioned previously, this scheme also provides easy implementation of both local and central control software in that all network commands are performed by emulating the local control panel.

To avoid an unduly complex interface when commands to the OAS are implemented from central control, the data structure was modified. Although commands are, in general, performed by emulating the local control panel, some operations (such as changing the alignment mode) needed to be simplified. The PBARRY, which is the array of push buttons for the local panel, was extended to include a command ID from the control room, which specifically identified the command being sent, the new alignment mode, a word for the desired setting of any attenuator and several spare words. With these few additions, implementation of both central control and local software was greatly simplified.

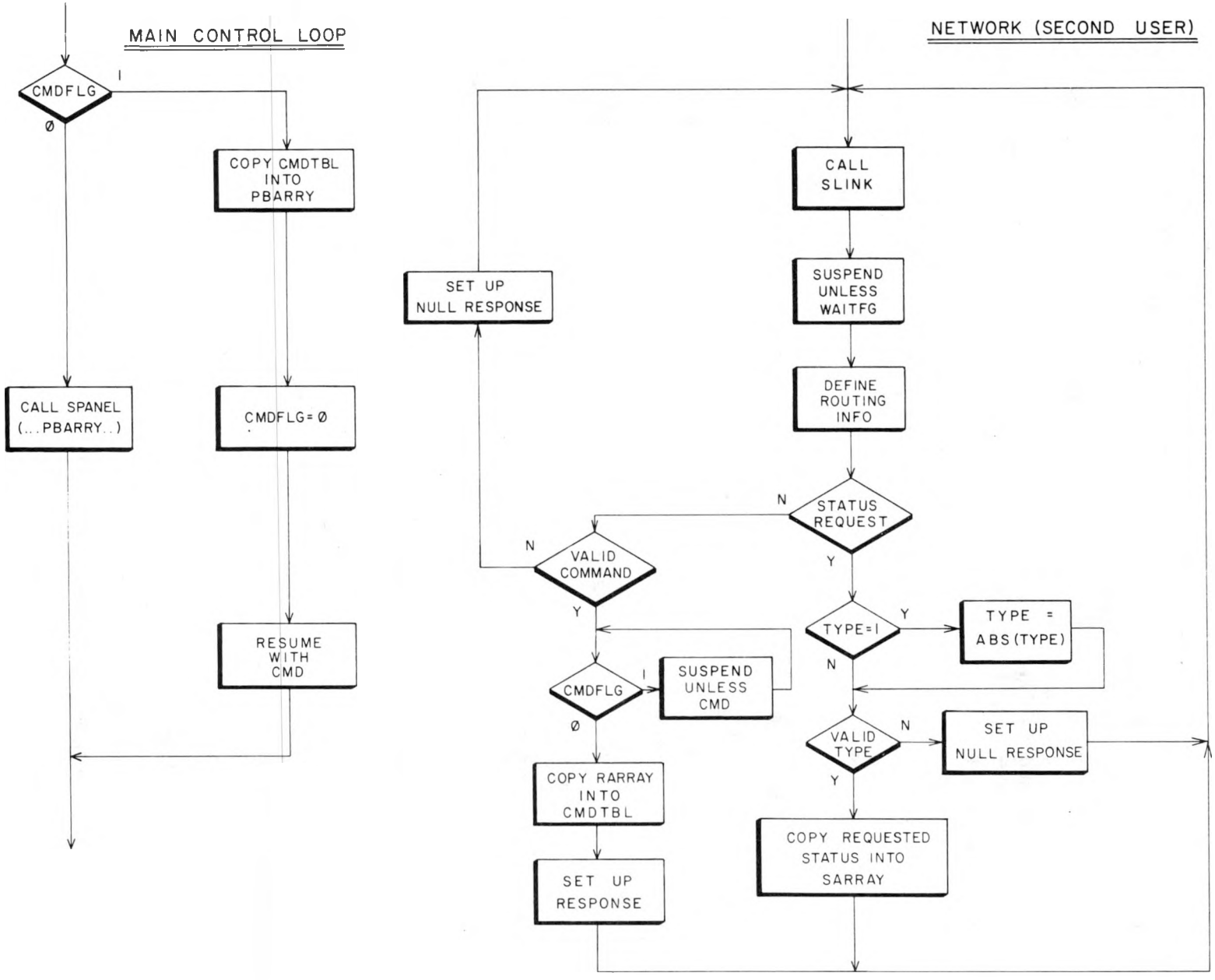


FIG. 33. Network communication interlock.

As shown in Fig. 34, the control command is decoded by the value of the command identification code which is sent as part of the PBARRY. At present, there are eight possible commands available via central control. Only the change oscillator (alignment mode) is substantially different when executed remotely; the new mode is selected immediately, contrasted with the "stepping-through" of different modes necessary with the local control panel. Once the alignment mode is changed, the system is placed in STOP, just as if a new mode had been selected locally. The value of "J" must also be updated since this important variable provides an index into virtually every array in the system.

Should an attenuator mode (PULSE, DYE, or CW) be selected, the attenuator can be set to any transmission value automatically. This is done by redefining the declared gain setting for that attenuator and commanding the motor to move to that point. The sensor attenuator is set in a similar manner, but can be set in any alignment mode. Conceivably, then, the shutter could be opened during a shot, an operation that must be done with care and with knowledge of the circumstances.

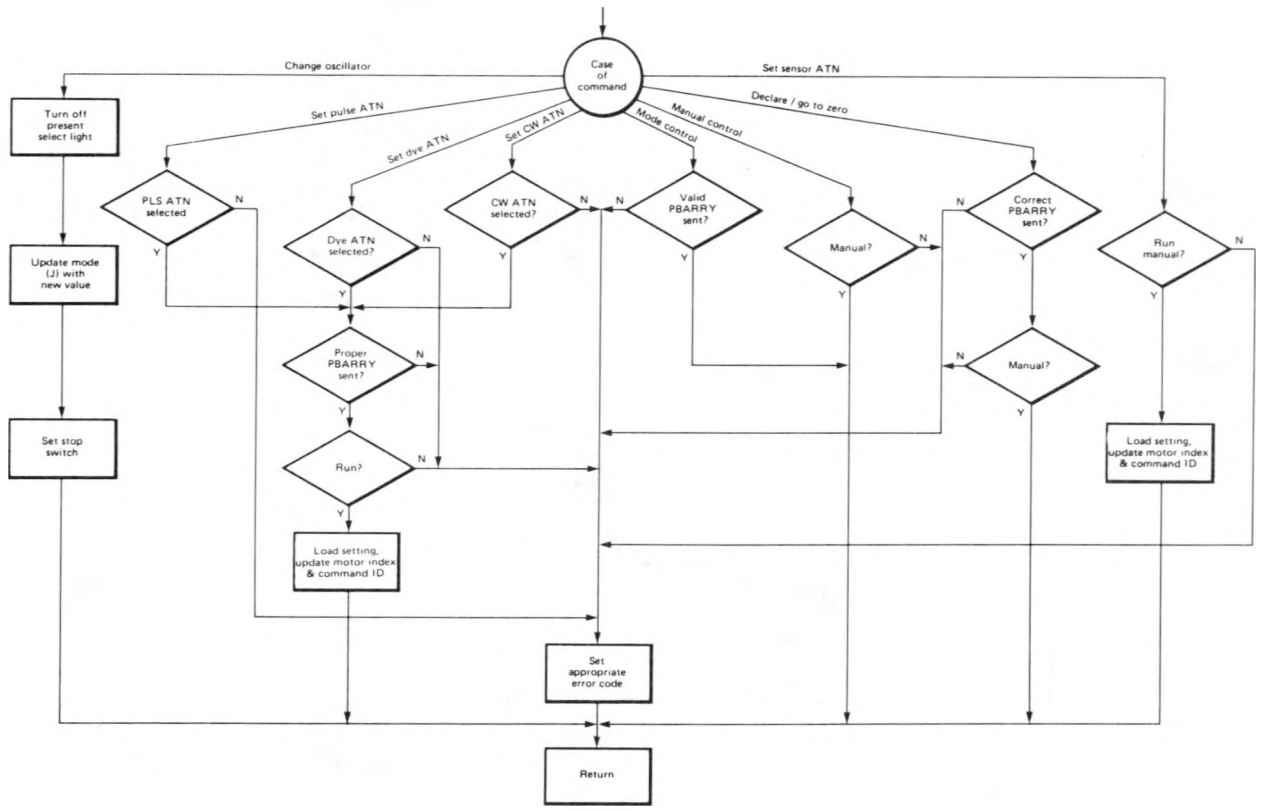


FIG. 34. Subroutine decode control room command.

The mode control function refers to the corresponding section of the local control panel. Any of the functions available locally can be enabled from central control. Thus, the system may be put into NETWORK or LOCAL, RUN or STOP, AUTO or MANUAL, and NORMAL or TEST modes. The DECLARE/GOTO ZERO command performs identically to that of the local panel. The manual control command allows direct control of OAS motors from the programmable switch panels located in the central console.

All commands except the change oscillator command require that a specifically formatted PBARRY be sent to the OAS. All commands except changing the oscillator and mode control require that the system be in or changing to MANUAL and RUN modes. Where appropriate, the received PBARRY is checked for validity (as well as MANUAL and RUN) and, if any discrepancy is discovered, an error code is set.

VIII. SYSTEM PERFORMANCE

The closed-loop response of the OAS to alignment errors of 400 μrad (pointing) and 4 mm (centering) is shown in Fig. 35. Errors in the alignment are reduced to the noise level in 10–20 seconds, a noteworthy response time particularly from the standpoint of aligning several different oscillators. Thus, coaligning alignment CW and pulsed oscillators typically takes a matter of minutes.

System resolution and stability are at least an order of magnitude better than any of the oscillators to be aligned. The system was biased in such a way that the nominally aligned signal on the detector produced a 5-V integrator output and thus an A/D output of approximately 2000 counts out of approximately 4000 (12-bit A/D; full scale is 4095). The system resolution is then given by

$$\text{Resolution} = \frac{R}{S} M \left(\frac{\pm 1 \text{ (LSB)}}{2000 + 2000} \right),$$

where R/S is the figure of merit for the detectors and M is the pointing or centering magnification. With R/S = 250 and the pointing and centering magnifications being 4.0 and 3.68, respectively, the system resolution is 0.25 μr for pointing and 0.23 mils for centering. That the system is stable to this degree has been verified by using a stable voltage reference as an input signal.

Actual system resolution is limited by inherent wander or drift in the laser oscillator being aligned. By running the system open-loop and monitoring alignment errors, it was determined that beam wander amounted to a maximum ±10 mils and ±10 μrad, centering and pointing alignment errors, respectively. For this reason, a dead band was incorporated into the system which inhibits closed-loop correction unless alignment errors exceed a given threshold value.

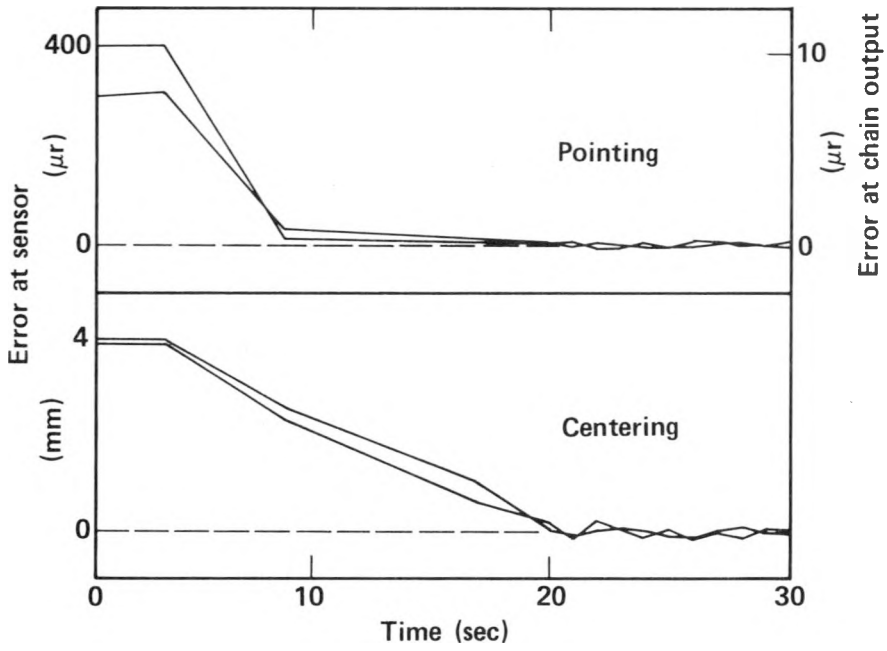


FIG. 35. Closed loop performance: acquisition range = 400 μr, 4 mm.

APPENDIX A. Schematics and Cabling

This appendix provides the minimal information necessary to build or debug the Oscillator Alignment System. The first part consists of several schematics to provide a reasonable understanding of the construction and hardware operation of the system. The second part is a set of pictorial instructions on how to build the myriad of cables used in the system. The system is shown diagrammatically with the LEA numbers of each major system component so that further information may be obtained from the particular drawing set. The cable ID numbers are identified in the table describing each and every cable in the system, the termination points, the connectors to be used at each termination, the nominal length, and the type cable used, including the appropriate LLL stock number. Cables that require special construction procedures are described on separate pages with the ID number to be used in crossreferencing back to the master cable table.

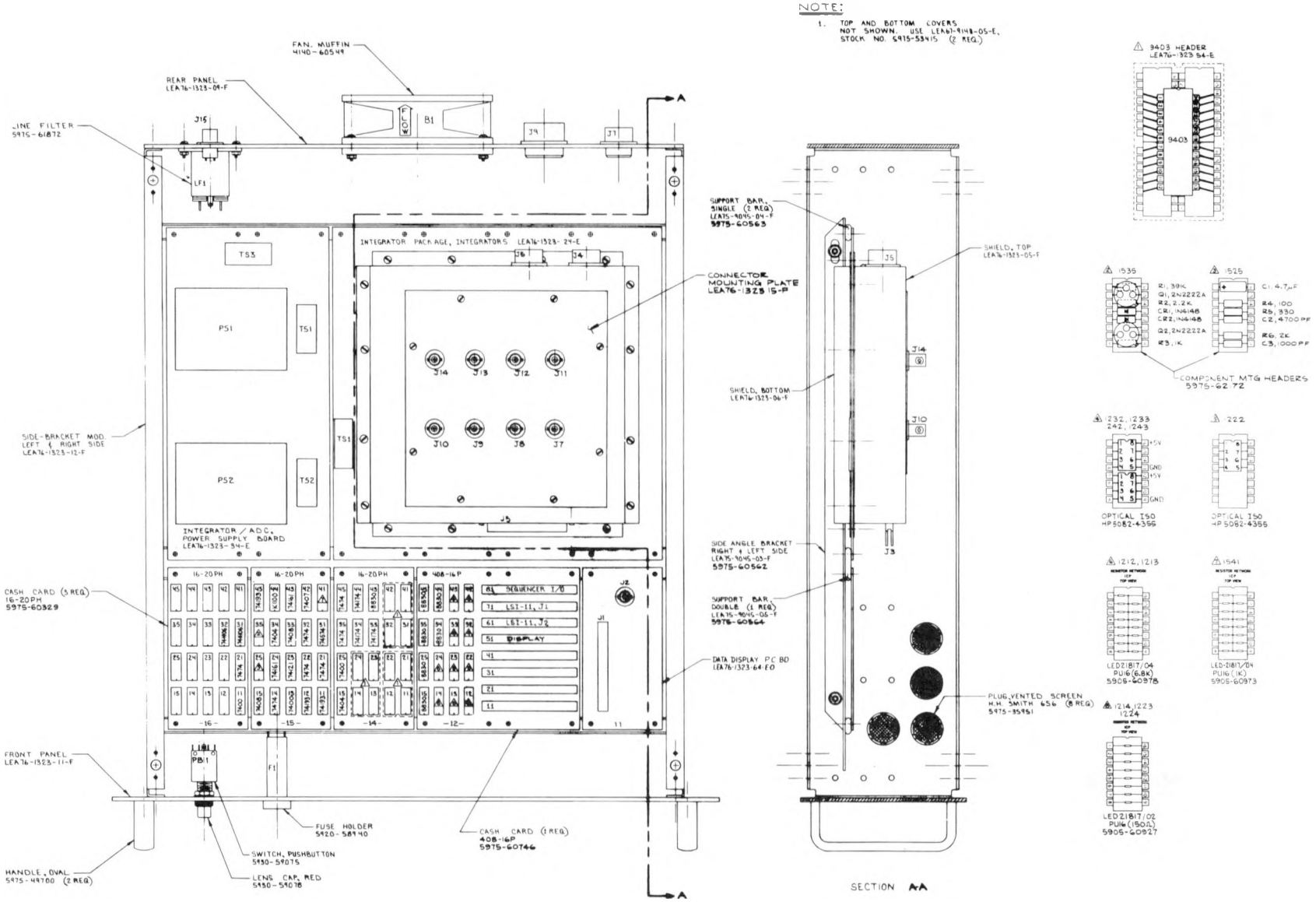


FIG. A-1. Integrator/ADC assembly.

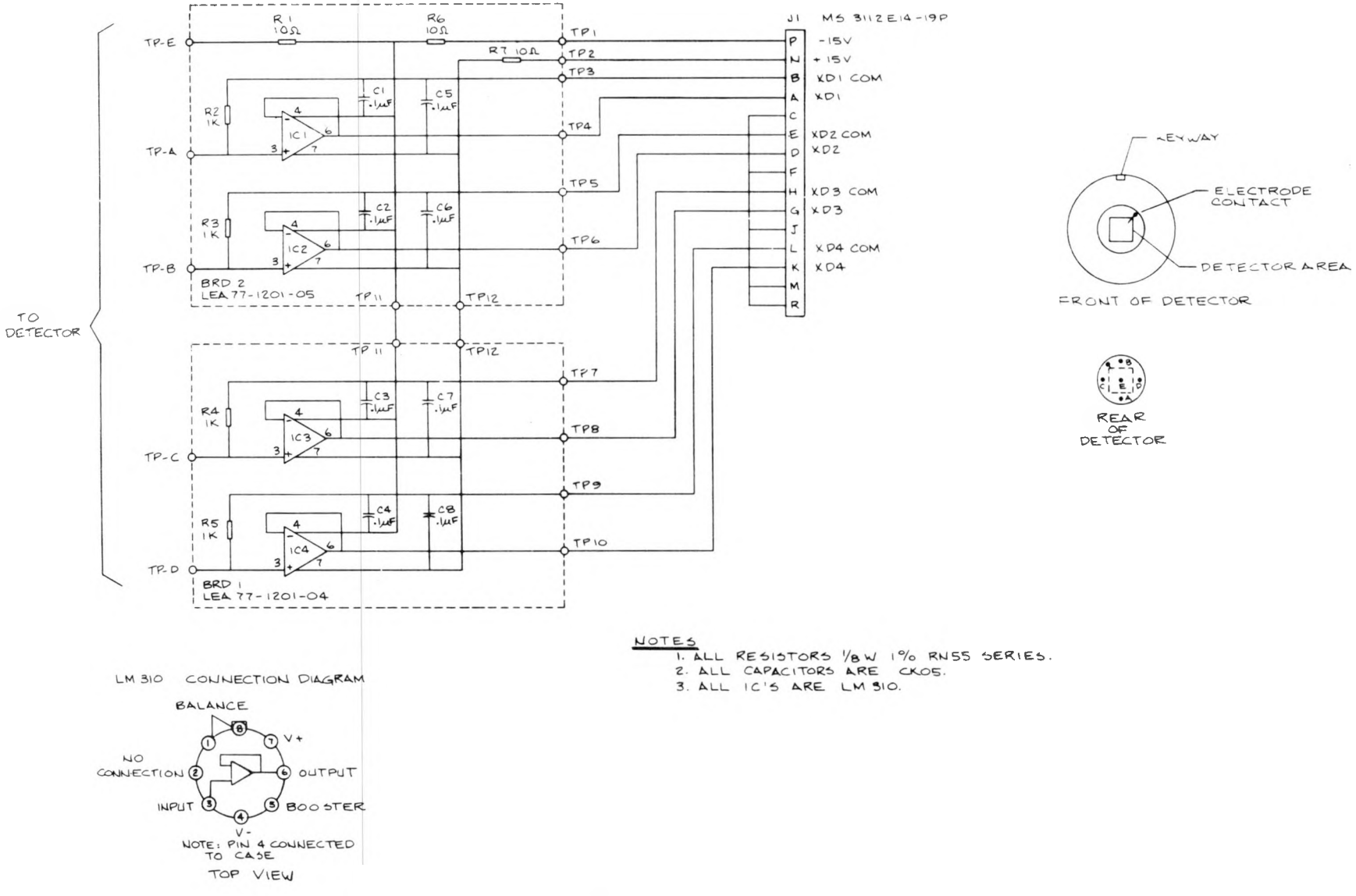


FIG. A-2. Schematic of oscillator alignment system, position detector 1 and 2.

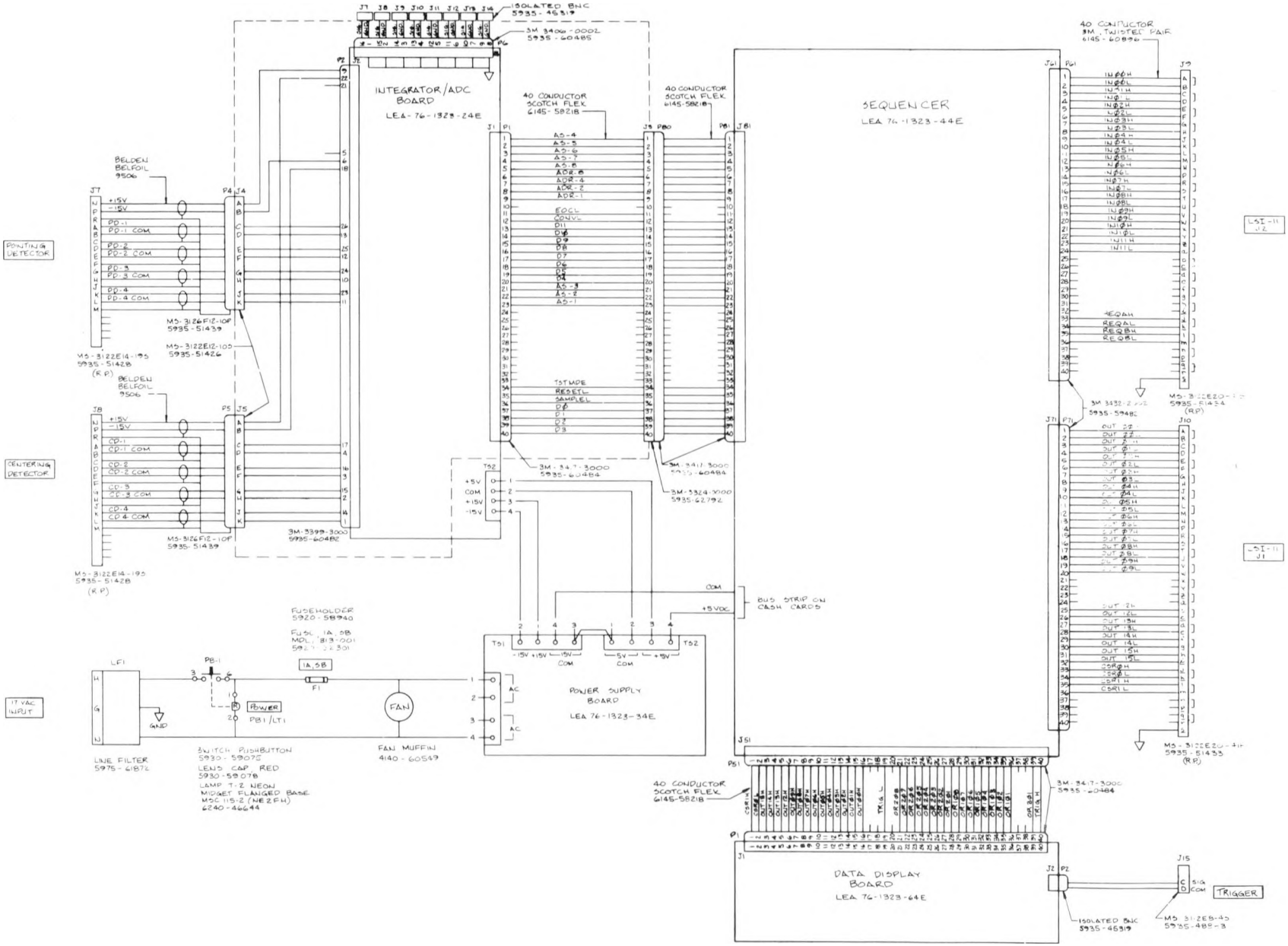


FIG. A-3. Integrator/ADC wiring diagram.

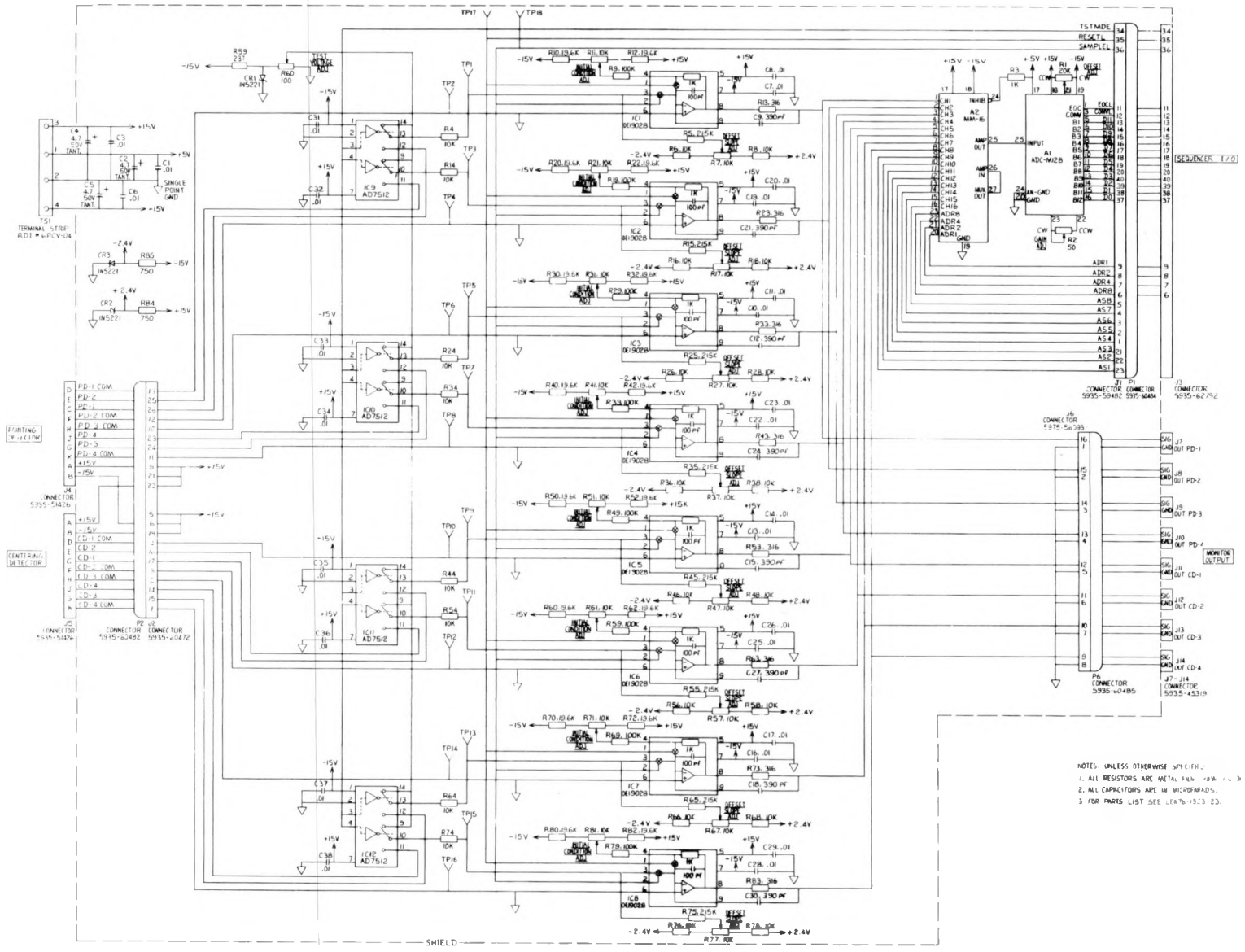


FIG. A-4. Integrator/ADC schematic.

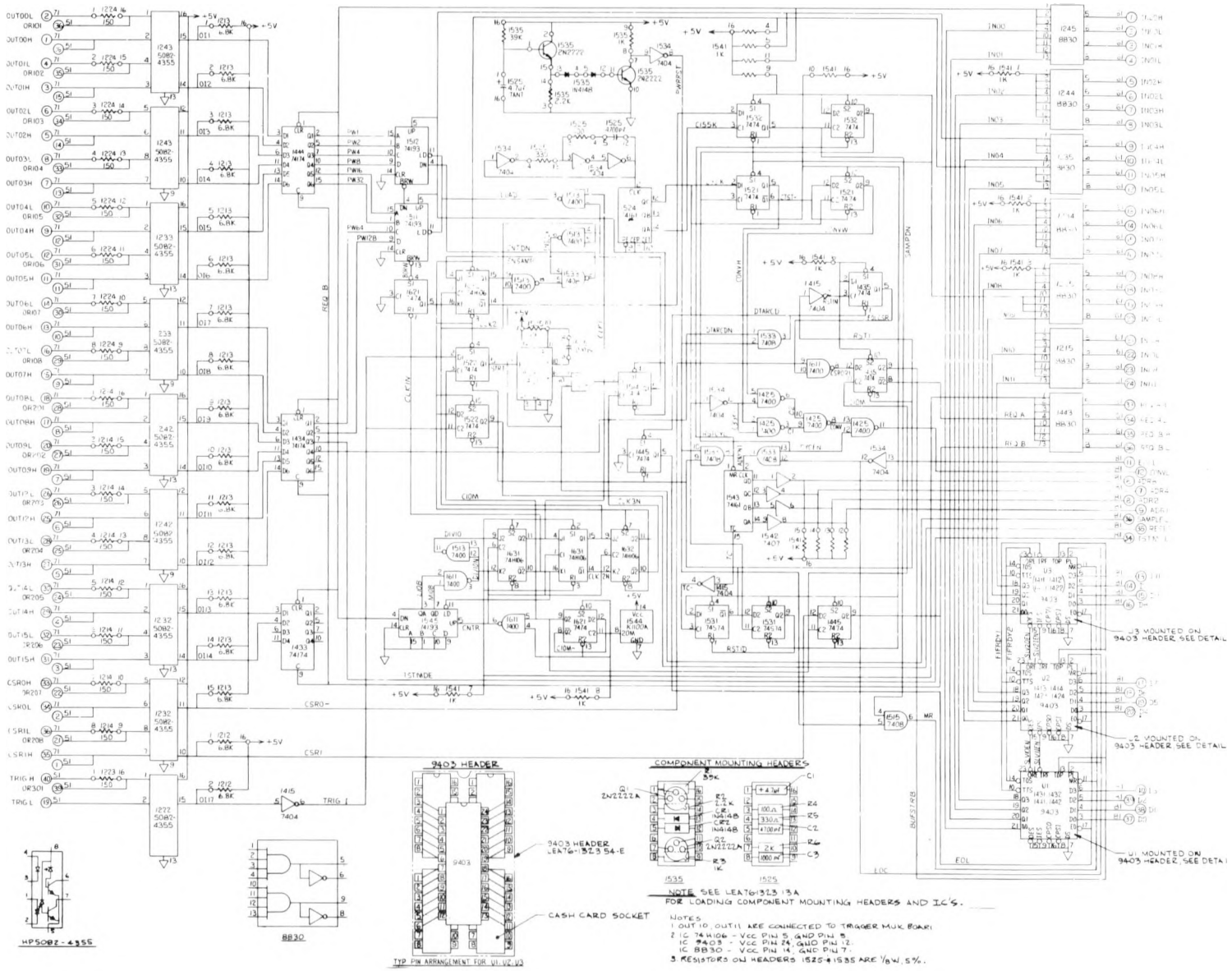
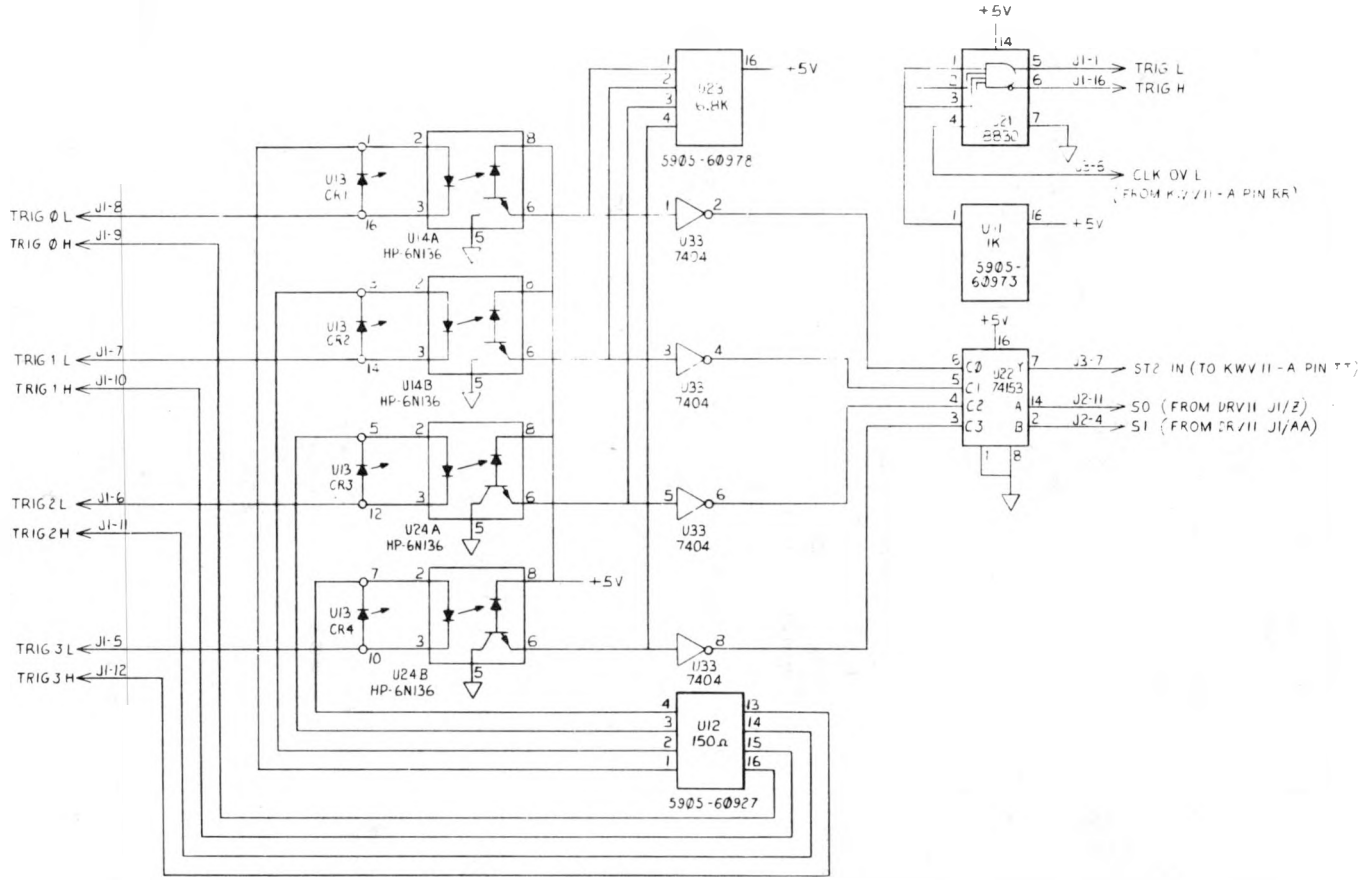


FIG. A-5. Integrator/ADC sequencer schematic.

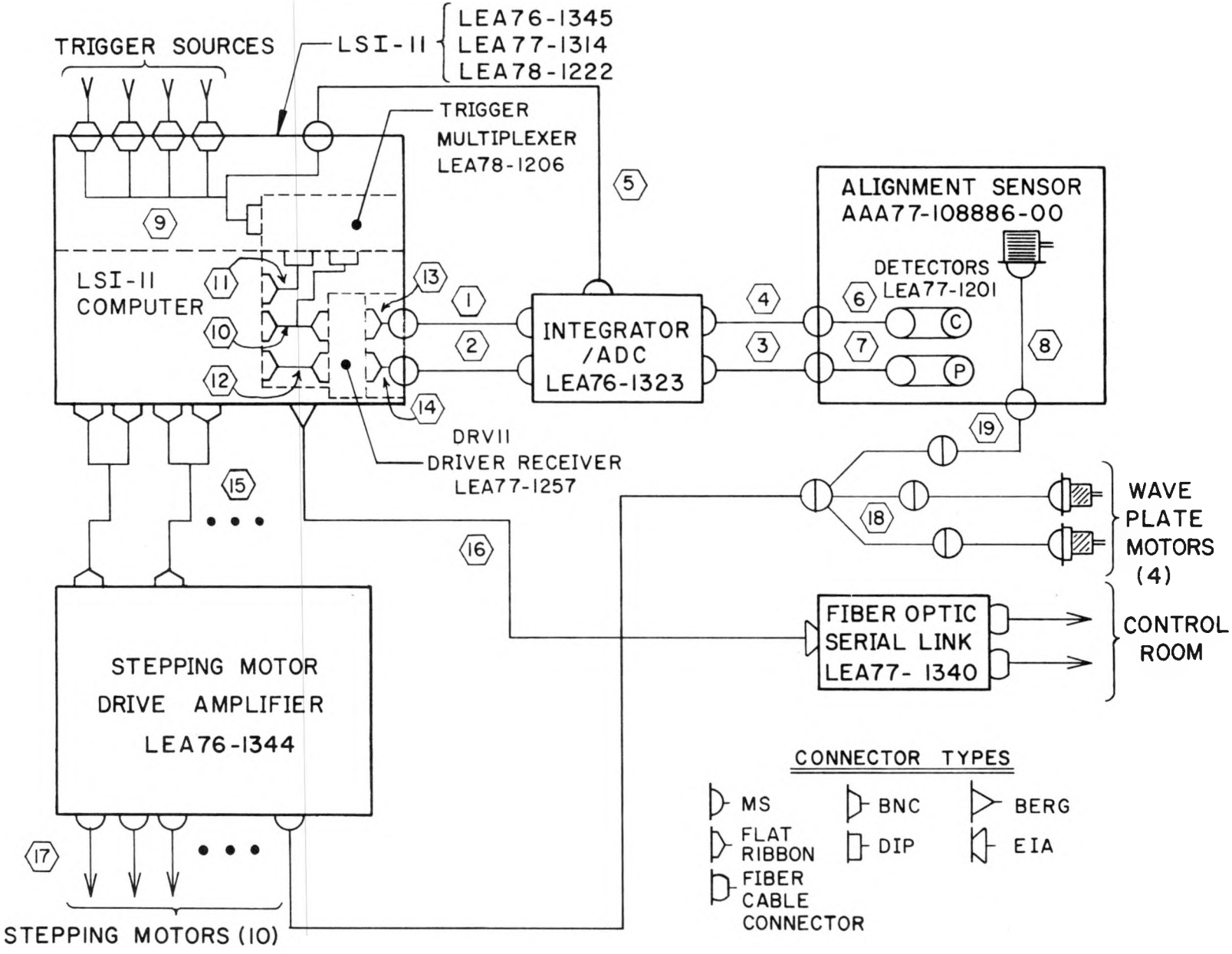


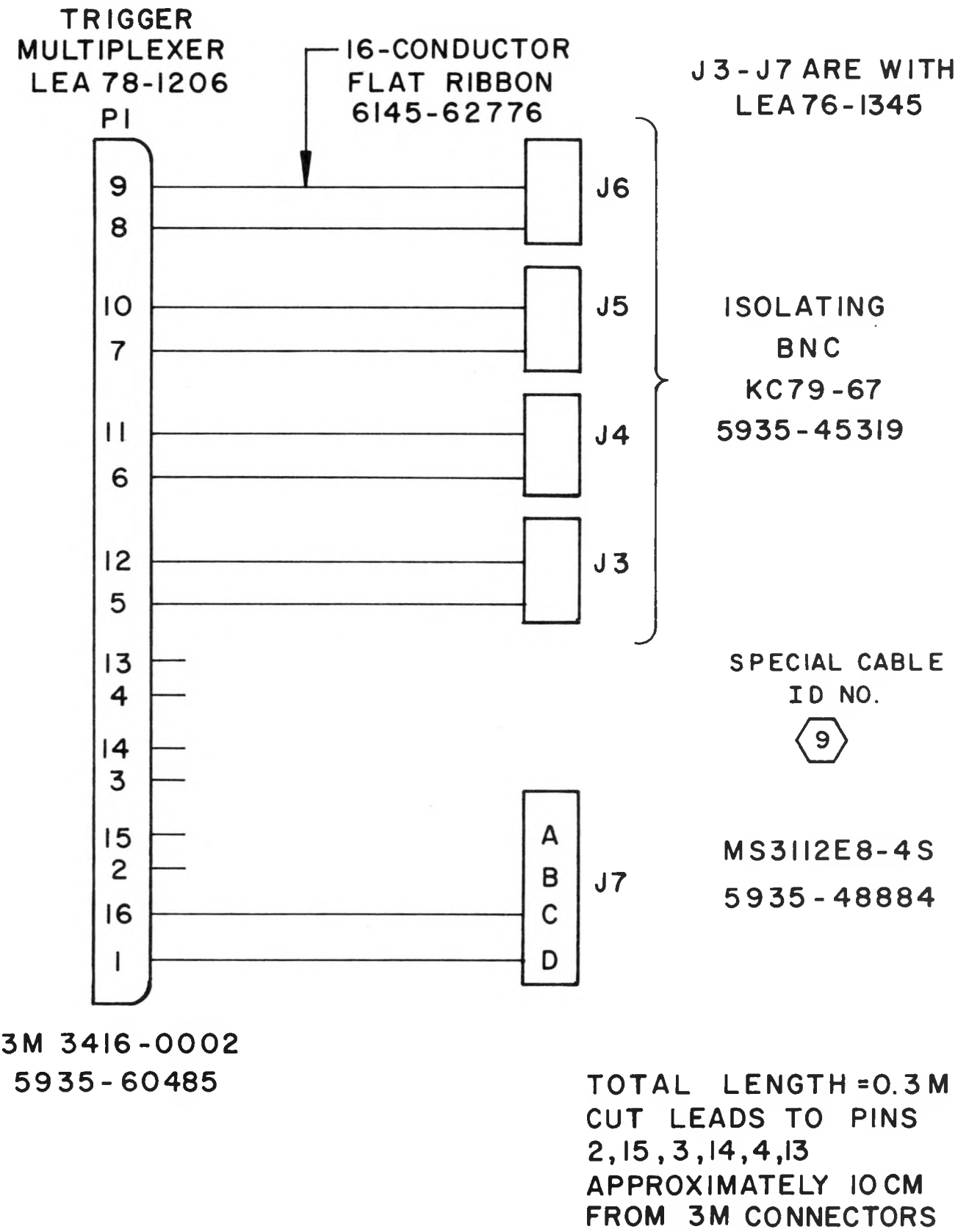
NOTES:

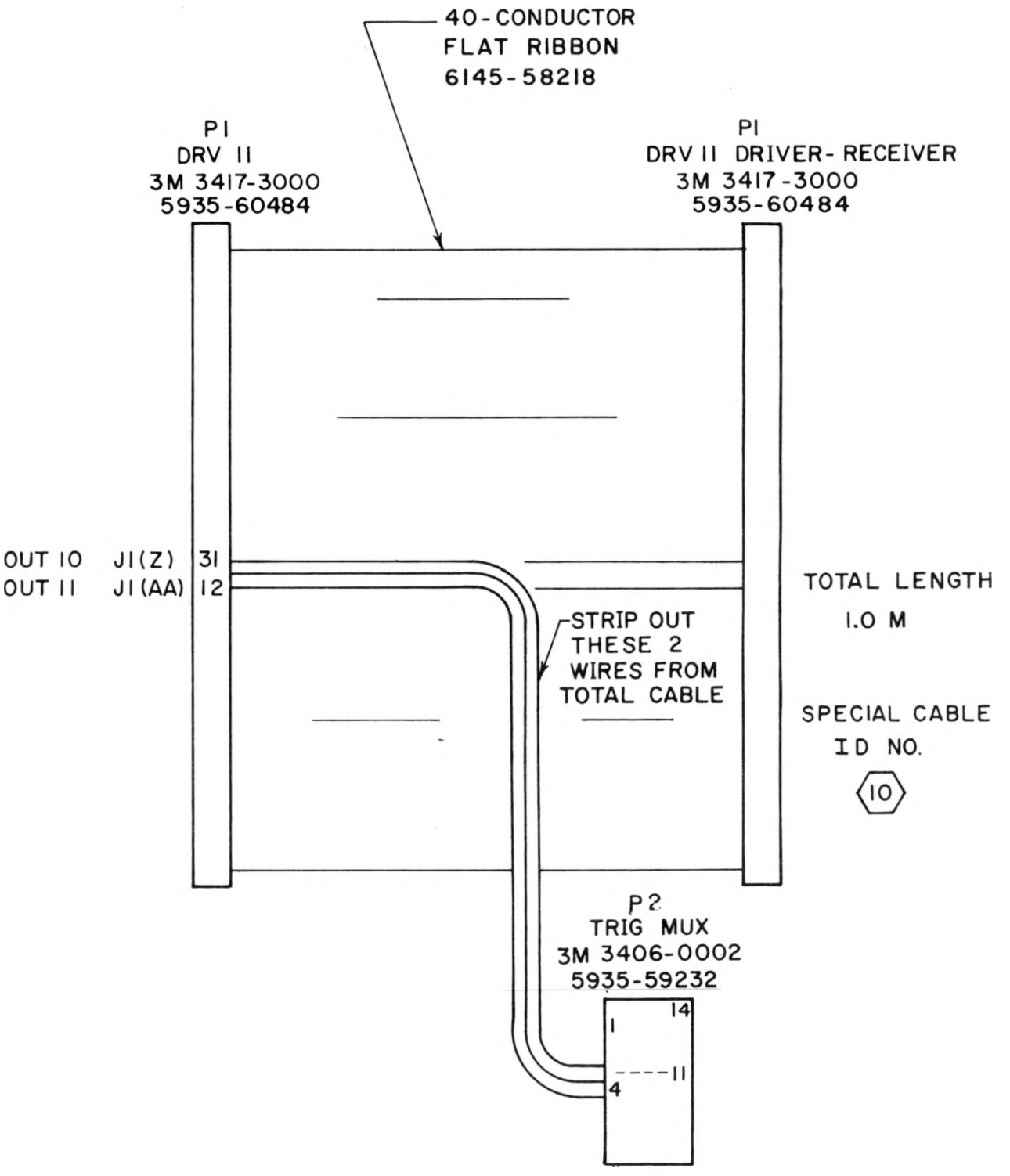
1. U33 PIN 4 IS V_{cc} U33 PIN 7 IS GND
2. J1 IS U34, J2 IS U32 J3 IS U43.
3. J1 J3 ARE 3M 3416-0002 J2 IS 3M 3406-0002.
4. LSI-11 +5V. AND COMMON CONNECTED DIRECTLY TO BOARD
5. CR1-CR4 ARE LED 21719 6240-59278

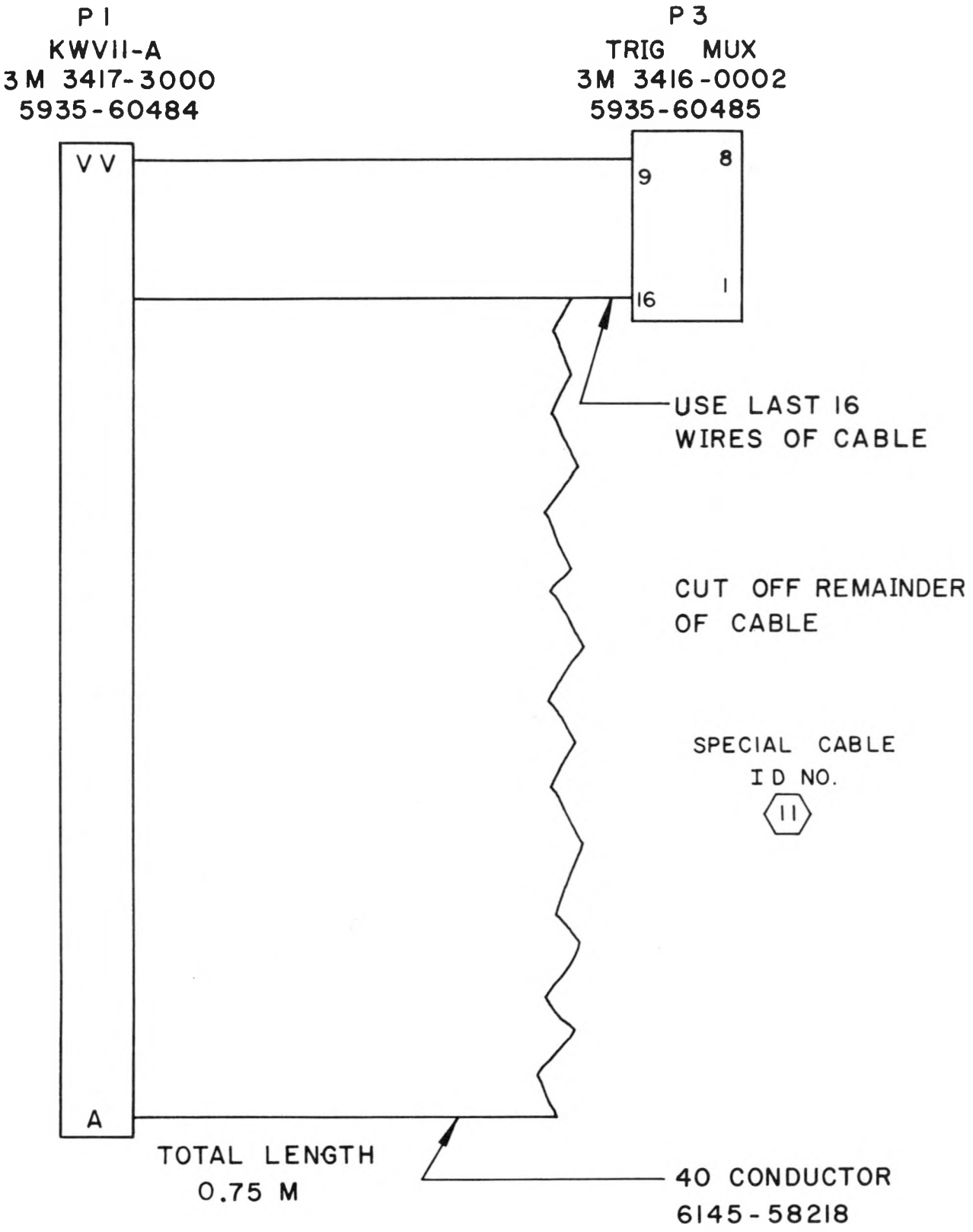
FIG. A-6. Trigger multiplexer schematic.

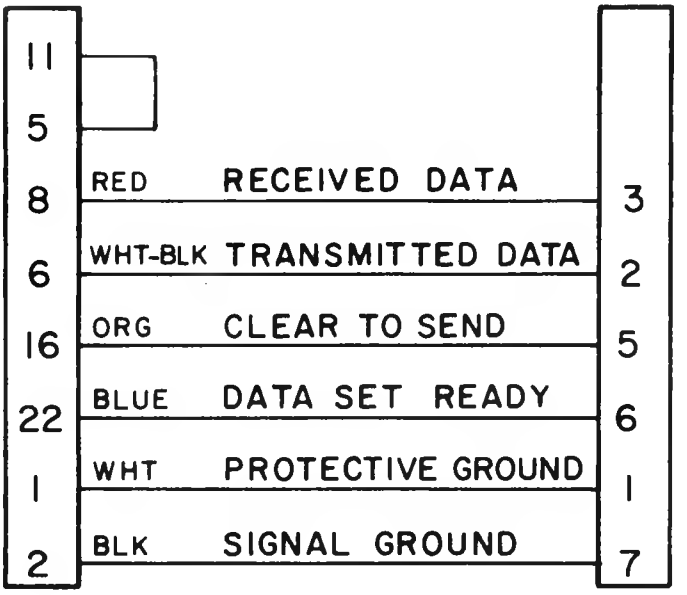
TERMINATIONS A	TERMINATIONS B	A	B	LENGTH	TYPE CABLE	ID NO.
P9(INT/ADC)	P2(LSI-11)	MS3126F20-41P 5935-51447	MS3126F20-41S 5935-51448	19M	BELDEN 8773 6145-62189	1
P10(INT/ADC)	P1(LSI-11)	MS3126F20-41S 5935-51448	MS3126F20-41P 5935-51447	19M	BELDEN 8773 6145-62189	2
P7(INT/ADC)	P3(SENSOR) (POINTING)	MS3126F14-19P 5935-51441	MS3126F14-19S 5935-51442	4M	BELDEN 9784	3
P8(INT/ADC)	P2(SENSOR) (CENTERING)	MS3126F14-19P 5935-51441	MS3126F14-19S 5935-51442	4M	BELDEN 9784	4
P11(INT/ADC)	P7(LSI-11) (TRIGGER)	MS3116F8-4S 5935-48868	MS3116F8-4P 5935-48867	19M	BELDEN 8723 6145-43545	5
J2(SENSOR)	P5(DETECTOR) (CENTERING)	MS3122E14-19P 5935-51427	MS3126F14-19S 5935-51442	0.75M	BELDEN 9784	6
J3(SENSOR)	P6(DETECTOR)	MS3122E14-19P 5935-51427	MS3126F14-19S 5935-51442	0.75M	BELDEN 9784	7
J1(SENSOR)	P4(SHUTTER MOTOR) (SHUTTER)	MS3122E16-26P 5935-51429	MS3126F16-26S 5935-51444	0.6M		8
P1(TRIG. MUX)	J3-J7(LSI-11)	3M 3416-0002 5935-60485	MS3112E8-4S 5935-48884 KC79-67 (BNC-4 REQ'D) 5935-45319	0.3M	MODIFIED 16-CONDUCTOR 6145-62776 REF LEA-78-1222-06	9
P2 (TRIG.MUX)	P1(DRV-11)	3M 3406-0002 5935-59232	3M 3417-3000 5935-60484	1M	MODIFIED 40-CONDUCTOR 6145-58218 REF. LEA-78-1222-07	10
P1(DRVII DR.-REC)		3M 3417-3000 5935-60484				
P1(KWVII-A)	P3(TRIG.MUX)	3M 3417-3000	3M 3416-0002	0.75M	MODIFIED 40 CONDUCTOR 6145-58218 REF.LEA-78-1222-08	11
P2(DRVII)	-P2(DRVII DR.REC)	3M 3417-3000 5935-60484	3M 3417-3000 5935-60484	0.3M	40 CONDUCTOR 6145-58218	12
J1(LSI-11)	-P3(DRVII DR. REC.)	MS3122E20-41S 5935-51433	3M 3417-3000 5935-60484	0.75M	40 CONDUCTOR 6145-58218	13
J2(LSI-11)	-P4(DRVII DR. REC.)	MS3122E20-41P 5935-51434	3M 3417-3000 5935-60484	0.75M	40 CONDUCTOR 6145-58218	14
P1(SMLC)	-P1(DRIVER CARD) P1(DRIVER CARD)	3M 3417-3000 5935-60484	3M 3399-3000 5935-60483 3M3399-3000 5935-60483	2 M	MODIFIED 40 CONDUCTOR 6145-58218 REF.LEA76-1344	15
P1(DLVII)	-EIA(FIBER OPTIC SERIAL LINK)	BERG 65043-017 5935-60990	AMPHENOL 17-20250-1 5935-34163	2 M	BELDEN 8427 6145-24475	16
STEPPING MOTOR CABLE REF. LEA76-1332-01 (8 REQUIRED)						17
SPECIAL 3-1 STEPPING MOTOR CABLE REF. LEA 76-1332-01 (2 REQUIRED)						18
SPECIAL SINGLE STEPPING MOTOR CABLE REF. LEA 76-1332-01 (6 REQUIRED)						19


TRIGGER SOURCES
LSI-11
LEA76-1345
LEA77-1314
LEA78-1222
TRIGGER
MULTIPLEXER
LEA78-1206
LSI-11
COMPUTER
INTEGRATOR
/ADC
LEA76-1323
ALIGNMENT SENSOR
AAA77-108886-00
DETECTORS
LEA77-1201
DRVII
DRIVER RECEIVER
LEA77-1257
WAVE
PLATE
MOTORS
(4)
FIBER OPTIC
SERIAL LINK
LEA77- 1340
CONTROL
ROOM
STEPPING MOTOR
DRIVE AMPLIFIER
LEA76-1344
STEPPING MOTORS (10)
CONNECTOR TYPES
MS
BNC
BERG
FLAT
RIBBON
DIP
EIA
FIBER
CABLE
CONNECTOR


TRIGGER
MULTIPLEXER
LEA 78-1206
P1
16-CONDUCTOR
FLAT RIBBON
6145-62776
J3-J7 ARE WITH
LEA 76-1345
9
8
J6
10
7
J5
ISOLATING
BNC
KC79-67
5935-45319
11
6
J4
12
5
J3
13
4
14
3
15
2
16
1
A
B
C
D
J7
SPECIAL CABLE
ID NO.
9
MS3112E8-4S
5935-48884
3M 3416-0002
5935-60485
TOTAL LENGTH=0.3M
CUT LEADS TO PINS
2,15,3,14,4,13
APPROXIMATELY 10CM
FROM 3M CONNECTORS


40-CONDUCTOR
FLAT RIBBON
6145-58218
P1
DRV 11
3M 3417-3000
5935-60484
P1
DRV 11 DRIVER-RECEIVER
3M 3417-3000
5935-60484
OUT 10 J1(Z) 31
OUT 11 J1(AA) 12
STRIP OUT
THESE 2
WIRES FROM
TOTAL CABLE
TOTAL LENGTH
1.0 M
SPECIAL CABLE
ID NO.
10
P2
TRIG MUX
3M 3406-0002
5935-59232
1
14
11
4


P 1
KWV11-A
3M 3417-3000
5935-60484
P 3
TRIG MUX
3M 3416-0002
5935-60485
V V
A
9
8
16
1
USE LAST 16
WIRES OF CABLE
CUT OFF REMAINDER
OF CABLE
SPECIAL CABLE
I D NO.
11
TOTAL LENGTH
0.75 M
40 CONDUCTOR
6145-58218



40 PIN
BERG CONNECTOR

25 PIN
AMPHENOL (MALE)
17-20250-1

EIA INTERFACE CABLE

⟨16⟩ FIBER OPTIC SERIAL LINK

APPENDIX B. Control Program

This appendix provides a complete listing of the OAS control program which runs in the LSI-11, as a complement to the descriptions given earlier in Section VII.

This program initializes the OAS LSI-11, performs all necessary control calculations, communicates with the central control room providing information about the system and executing received commands, oversees all control operations, and interacts with an operator via a specially designed control panel.

The program is written in and runs under the REBEL/BASIC system.

```
    2 ! OSCILLATOR ALIGNMENT SYSTEM--INITIALIZATION AND CONTROL PROGRAMS
    3 ! AUTHORS: JOHN PARKER AND MARK A. SUMMERS
!
!
! NOTE:
!               This program was modified by F. Holloway to allow a separate
!               'User' (OASCNTRH.BAS) to perform the control room I/O operations
!               in a 'standard' fashion.  OASCNTRM.BAS contains a complete
!               description of the standard features.
!
! Specific Modifications Were:
!
!               1. All references to SARRY1,SARRY2,SARRY3,RARRAY were deleted.
!
!               2. The Network Update Subroutine (10000 etc.) was deleted,
!                  and the functions it performed were put into OASCNTRM.
!
!               3. The variables CMD, CMDFLC, and the array CMDTBL was added
!                  to the common block.
!
!       4. The original arrays PBARRY and BITLIT were put in the common
!                  block.
!
!
!-------------------- INITIALIZATION --------------------
!
    4 ACTIVITY OAS

    5 CMD USE 2=KB0
    9 IMP INTEGER A-Z
   10 LET NWRDS=307,C=0,BERCOD=678
   20 CALL CORE(NWRDS,C,BERCOD)
   25 LET @046=C


   82 EQU J=@(C)!                                          OSCILLATOR MODE SELECTION
!                                                          J = 0 : CW Long Path
!                                                          J = 1 : CW Attenuator
!                                                          J = 2 : CW Short Path
!                                                          J = 3 : Pulse Oscillator
!                                                          J = 4 : Pulse Attenuator
!                                                          J = 5 : Dye Attenuator
!                                                          J = 6 : Auxiliary
!                                                          J = 7 : Pulse Synchronization


!                     The first nine arrays in the following common block have
!                     a similar structure in that all are indexed by the variable
!                     J (Oscillator Mode) defined above.

   83 INTEGER AZM(7)@(C+2)=0!                              AZIMUTH ALIGN. ERRORS,EACH OSC.
   84 INTEGER ELV(7)@(C+18)=0!                             ELEVATION ALIGN ERRORS, EACH OSC
   85 INTEGER HORZ(7)@(C+34)=0!                            HORIZ ALIGN ERRORS, EACH OSC
   86 INTEGER VERT(7)@(C+50)=0!                            VERTICAL ALIGN ERRORS, EACH OSC
   87 INTEGER DLY(7)@(C+66)=0!                             TRIGGER DELAY, EACH OSC
   90 INTEGER PTOL(7)@(C+82)=0!                            POINTING TOLERANCE
   92 INTEGER CTOL(7)@(C+98)=0!                            CENTERING TOLERANCE
   93 INTEGER PRANGE(7)@(C+114)=0!                         POINTING RANGE
   94 INTEGER CRANGE(7)@(C+130)=0!                         CENTERING RANGE
```

```
!                Motor positions since last declare are indexed via the
!              Oscillator Mode (J) variable also, however, the structure is

!              slightly different: for each mode there are four motors (two per
!              gimbal, two gimbals per mode)

  95 INTEGER MOTOR(36)@(C+146)=0!                                  MOTOR POSITIONS SINCE DECLARE
  96 EQU ERROR=@(C+220)!                                           ERROR CODE
  97 INTEGER INTOFF(63)@(C+222)=0!                                 INTEGRATOR OFFSETS,8 PER J
  98 INTEGER CPOSIT(40)@(C+350)=0!                                 MOTOR POSITIONS
  99 INTEGER IP(7)@(C+432)=0!                                      POINTING INTENSITY
 100 INTEGER IC(7)@(C+448)=0!                                      CENTERING INTENSITY

!                The following is a definition of the bytes in PBARRY
!              in terms of the push button functions they represent on the
!              LSI-11 control panel:

!              PBARRY(0)      Alignment Errors      PBARRY(20)     +Horz,+Az
!              PBARRY(1)      Motor Positions       PBARRY(21)     -Horz,-Az
!              PBARRY(2)      Centering Data        PBARRY(22)     +Vert,+El
!              PBARRY(3)      Pointing Data         PBARRY(23)     -Vert,-El
!              PBARRY(4)      Centering Offsets     PBARRY(24)     Slew
!              PBARRY(5)      Pointing Offsets      PBARRY(25)     Step
!              PBARRY(6)      Select Up             PBARRY(26)     Atten Open
!              PBARRY(7)      Select Down           PBARRY(27)     Atten Close
!              PBARRY(8)      Declare Zero          PBARRY(28)     Offsets
!              PBARRY(9)      Go To Zero            PBARRY(29)     Gate Width
!              PBARRY(10)     Network               PBARRY(30)     Error Code
!              PBARRY(11)     Local                 PBARRY(31)     Trigger Delay
!              PBARRY(12)     Run                   PBARRY(32)     Atten Position
!              PBARRY(13)     Stop                  PBARRY(33)     Command ID from Control Room
!              PBARRY(14)     Auto                  PBARRY(34)     Selecte Osc  from Control Room
!              PBARRY(15)     Manual                PBARRY(35)     Spare
!              PBARRY(16)     Normal                PBARRY(36)     1st byte of Waveplate Setting
!              PBARRY(17)     Test                  PBARRY(37)     2nd byte of Waveplate Setting
!              PBARRY(18)     Center                PBARRY(38)     Spare
!              PBARRY(19)     Point                 PBARRY(39)     Spare

 102 BYTE PBARRY(39)@(C+464)=0!                                    PANEL PUSH BUTTON ARRAY
!                                                                  33 P.B.'s; 7 Control Bytes
 104 EQU CMD=@(C+504)
 105 INTEGER CMDTBL(19)@(C+506)=0
 110 EQU CMDFLG=@(C+546)
 111 BYTE OASTAT(13)@(C+549)=0!                                    OAS Status Table defined below
 112 BYTE BITLIT(49)@(C+562)=0
 113 EQU WAITFG=@(C+612)
```

```
!                    The following is a definition of the BITLIT array
!                 as pertains to the Oscillator Alignment LSI-11 Control Panel:

!                 BITLIT(0)       Trig Received          BITLIT(24)      CW Horz
!                 BITLIT(1)       Vert Range             BITLIT(25)      CCW Horz
!                 BITLIT(2)       Elv Range              BITLIT(26)      CW Elv
!                 BITLIT(3)       Horz Range             BITLIT(27)      CCW Elv
!                 BITLIT(4)       Signal Level           BITLIT(28)      CW Azm
!                 BITLIT(5)       Azm Range              BITLIT(29)      CCW Azm
!                 BITLIT(6)       Operating              BITLIT(30)      CW Atten
!                 BITLIT(7)       Errors Detected        BITLIT(31)      CCW Atten
!                 BITLIT(8)       Align Errors           BITLIT(32)      Net
!                 BITLIT(9)       Motor Position         BITLIT(33)      Local
!                 BITLIT(10)      Center Data            BITLIT(34)      Run
!                 BITLIT(11)      Point Data             BITLIT(35)      Stop
!                 BITLIT(12)      Center Offsets         BITLIT(36)      Auto
!                 BITLIT(13)      Point Offsets          BITLIT(37)      Manual
!                 BITLIT(14)      CWLP                   BITLIT(38)      Normal
!                 BITLIT(15)      CW ATN                 BITLIT(39)      Test
!                 BITLIT(16)      CWSP                   BITLIT(40)      Center
!                 BITLIT(17)      PLS                    BITLIT(41)      Point
!                 BITLIT(18)      PLS ATN                BITLIT(42)      +Horz,+Az
!                 BITLIT(19)      DYE ATN                BITLIT(43)      -Horz,-Az
!                 BITLIT(20)      Auxiliary              BITLIT(44)      +Vert,+El
!                 BITLIT(21)      PSS                    BITLIT(45)      -Vert,-El
!                 BITLIT(22)      CW Vert                BITLIT(46)      Slew
!                 BITLIT(23)      CCW Vert               BITLIT(47)      Step
!                                                        BITLIT(48)      Atten Open
!                                                        BITLIT(49)      Atten Closed

   115 BYTE MNUMB,TRANS,CTRAN,MSTAT
   120 REAL N,M,L,HAZMAG,VELMAG,GIMPOS,ERR,STARTT,PREST
   122 REAL STRPLS,STRDYE,STRSWP,STRCW
   125 REAL PTRPLS,PTRDYE,PTRSWP,PTRCW

   130 DIM OLDSTA(13)!                                                    BACKUP STATUS ARRAY
   135 DIM TEMP(5)

!Status array entries for OAS are defined as follows:

!                 OASTAT(0) ---------- J (Beam selection)
!                 OASTAT(1) ---------- WP#1
!                 OASTAT(2) ---------- WP#2
!                 OASTAT(3) ---------- WP#3
!                 OASTAT(4) ---------- WP#4
!                 OASTAT(5) ---------- WP#5
!                 OASTAT(6) ---------- Gimbal #2 for pulse modes
!                 OASTAT(7) ---------- Gimbal #2 for CW or PSS modes
!                 OASTAT(8) ---------- Gimbal #1 for CW modes
!                 OASTAT(9) ---------- Gimbal #1 for PSS mode
!                 OASTAT(10) --------- Gimbal #1 for pulse modes
!                 OASTAT(11) --------- Pointing status
!                 OASTAT(12) --------- Centering status
!                 OASTAT(13) --------- Change flag (one of the status bytes has changed since last time)
```

```
140 DIM MNUMB(37),TRANS(40),CTRAN(40)!
150 DIM CDIST(40),RATEDV(40),DIST(40)!
160 DIM POSITN(40),MSTAT(37),NS(20)!

170 DIM CONFIG(5),DIGLIT(4),DCARRY(0)!

180 DIM AZOFF(10),ELOFF(10)!
190 DIM HOFF(10),VOFF(10)!

195 DIM SETC(40)!

200 DIM TRIG(7)!

210 DIM GATE(7)!

220 DIM DATA(72)!

230 DIM HAZMAG(7),VELMAG(7)!

250 DIM GIMPOS(15)!

260 DIM ROT(40)!

270 DIM ZERO(31)!

280 DIM WOFF(7)!

290 DIM ATTEN(7)!

300 DIM ERR(20)=0!

305 LET NPB=33,NB=50,M=33,K=0,NC=5
310 LET P=0,Q=0,TEST=0,ND=0
315 LET ERROR=2000,DEBUG1=0,SETGBL=0
320 LET COUNT=0,FH=250,N=3.66,M=4.0,NS(0)=15
325 LET C=0,EXT=1,TDC=0,MDC=0,ID=2
330 LET J=2,RUN=0,AUTO=0,CONT=0,COM2=0,COM3=0,SLEW=0
335 LET NUMB=0,LKERR=0,SFLG=0,NORM=1
340 LET PNT=0,CNT=0,ELCID=0
345 LET BLANK = -32768!
350 LET CMD=0,WAITFC=0
```

ARRAYS FOR MOTOR CALLS

ARRAYS FOR PANEL CALLS

OFFSET COMPENSATION FOR
APPARENT/ACTUAL "ZERO"

GAIN SETTING-WAVEPLATES

TRIGGER SELECTION

INTEGRATOR SAMPLE TIME

DATA FROM INTEGRATOR/ADC

OPTICAL MAGNIFICATIONS

GIMBAL SPACING PARAMETERS

WAVEPLATE ANGULAR POSITIONS

ARRAY OF DECLARED POSITIONS

WAVEPLATE OFFSETS

ATTENUATOR POSITIONS

ERROR VALUES FOR CALCULATION

CODE FOR BLANKING DIGITAL DISPLAY

```
!--------------------- GIMBAL POSITIONS IN MILS -------------------------

   402 DATA 494200,28000,0,0,66500,28000,516200,156800
   403 DATA 0,0,0,0,0,0,66500,147320
   404 FOR Z=0 TO 7
   406 READ GIMPOS(2*Z),GIMPOS(2*Z+1)
   408 NEXT Z
   409 RESTORE 412


!------------- INTEGRATOR GATE WIDTH IN UNITS OF MICROSECONDS ----------

   412 DATA 250,250,250,250,250,250,250,250
   414 FOR Z=0 TO 7
   416 READ GATE(Z)
   418 NEXT Z
   419 RESTORE 432


!---------------- ATTENUATOR POSITIONS IN UNITS OF STEPS ---------------

   432 DATA 320,0,320,750,0,0,0,0
   434 FOR Z=0 TO 7
   436 READ ATTEN(Z)
   438 NEXT Z
   439 RESTORE 442


!---------------- TRIGGER DELAYS IN UNITS OF MICROSECONDS -----------

   442 DATA 999,999,999,75,999,999,999,999
   444 FOR Z=0 TO 7
   446 READ DLY(Z)
   448 NEXT Z
   449 RESTORE 452


!--------------------- RATE DIVISORS ----------------------

   452 DATA 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
   453 DATA 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
   454 FOR Z=0 TO 40
   456 READ RATEDV(Z)
   458 NEXT Z
   459 RESTORE 462


!--------------------- OPTICAL CONSTANTS -------------------------

   462 DATA 2.0,-2.0,2,2,-2.2,2.2,2.0,-2.0
   463 DATA 2,2,2,2,2,2,-2.2,2.2
   464 FOR Z=0 TO 7
   466 READ HAZMAG(Z),VELMAG(Z)
   468 NEXT Z
   469 RESTORE 472
```

```
!------------------- POINTING RANGE ---------------------

   472 DATA 500,500,500,500,500,500,500,500
   474 FOR Z=0 TO 7
   476 READ PRANGE(Z)
   478 NEXT Z
   479 RESTORE 482

!------------------ CENTERING RANGE --------------------

   482 DATA 200,200,200,200,200,200,200,200
   484 FOR Z= 0 TO 7
   486 READ CRANGE(Z)
   488 NEXT Z
   489 RESTORE 502

!-------------------- ASSIGN TRIGGERS --------------------

   502 DATA 1,1,1,2,1,1,1,1
   504 FOR Z=0 TO 7
   506 READ TRIG(Z)
   508 NEXT Z
   509 RESTORE 512

 !------------------ POINTING TOLERANCE ----------------------

   512 DATA 10,10,10,10,10,10,10,10
   514 FOR Z = 0 TO 7
   516 READ PTOL(Z)
   518 NEXT Z
   519 RESTORE 522

!------------------ CENTERING TOLERANCE ------------------

   522 DATA 10,10,10,10,10,10,10,10
   524 FOR Z = 0 TO 7
   526 READ CTOL(Z)
   528 NEXT Z
   529 RESTORE 532


!----------------- -- WAVEPLATE POSITIONS -----------------

   532 DATA 0,0,45,0,0,0,0,45
   534 DATA 45,0,0,0,0,0,0,0
   536 DATA 45,45,0,0,45,45,45,0
   537 DATA 0,0,0,45,45,45,45,0
   538 FOR Z=0 TO 31
   540 READ ROT(Z)
   542 NEXT Z
   544 RESTORE 564

!-------------------- WAVE PLATE OFFSETS -----------------

   564 DATA 117,0,92,-60,0,0,0,0
   566 FOR Z=0 TO 7
   568 READ WOFF(Z)
   569 NEXT Z
   595
```

```
!-------------------- MOTOR ASSIGNMENTS -----------------

!                     Associate each element of the 'MNUMB' array (motor
!                table) with a particular port of the Motor Drive Amp Chassis.
!
   602 LET MNUMB(0)=1,MNUMB(1)=2,MNUMB(2)=4,MNUMB(3)=5
   603 LET MNUMB(4)=7,MNUMB(5)=8,MNUMB(6)=10,MNUMB(7)=11
   604 LET MNUMB(8)=1,MNUMB(9)=2,MNUMB(10)=4,MNUMB(11)=5
   605 LET MNUMB(12)=7,MNUMB(13)=8,MNUMB(14)=10,MNUMB(15)=11
   606 LET MNUMB(16)=7,MNUMB(17)=8,MNUMB(18)=10,MNUMB(19)=11
   607 LET MNUMB(20)=1,MNUMB(21)=2,MNUMB(22)=13,MNUMB(23)=14
   608 LET MNUMB(24)=7,MNUMB(25)=8,MNUMB(26)=10,MNUMB(27)=11
   609 LET MNUMB(28)=1,MNUMB(29)=2,MNUMB(30)=13,MNUMB(31)=14
   610 LET MNUMB(32)=19,MNUMB(33)=20,MNUMB(34)=21,MNUMB(35)=22
   611 LET MNUMB(36)=23,MNUMB(37)=24

   612 CALL IMOTOR(NS,MNUMB,POSITN,TRANS)!                          INITIALIZE MOTORS


!-------------------- INITIALIZE WAVEPLATES --------------------

   615 LET NS(0)=4
   620 FOR Z=0 TO NS(0)
   622 LET DIST(32+Z)=-2000
   625 NEXT Z
   630 CALL CMOTOR(NS,MNUMB(32),RATEDV(32),DIST(32))
   640 CALL SMOTOR(NS,MNUMB(32),CDIST(32),CPOSIT(32),CTRAN(32),MSTAT(32))
   645 IF NS(5)<>0 GOTO 640!                                        TEST MOTOR POWER STILL ON
   650 FOR Z=32 TO 31+NS(0)
   655 IF MSTAT(Z)<>-1 ENTER "WAVEPLATE ERROR"
   660 NEXT Z


   663 FOR Z=0 TO NS(0):LET CDIST(32+Z)=0:NEXT Z!                   INITIALIZE
   665 CALL IMOTOR(NS,MNUMB(32),POSITN(32),TRANS(32))!              ATTENUATOR

               GET STATUS OF INITIALIZED MOTORS

   666 CALL SMOTOR(NS,MNUMB(32),CDIST(32),CPOSIT(32),CTRAN(32).MSTAT(32))
   667 PRINT "INITIALIZATION COMPLETE"


   671 DEL 1-670


   675 CMD USE 4=DX0,5=DX0
   677 MORE/CN!                                                     LOAD CONTROL PROGRAM
   678 PRINT "CONTROL PROGRAM - USER #1 - LOADED"

   679 GOTO 760
RUN

   700 CMD LOCON 2=DX0
   705 SUSPEND

   706 PRINT "BEGIN CONTROL OPERATIONS"
   707 CALL ILINK(1,ID)
```

```
!------------------ CROSS COUPLING FUNCTIONS ----------------------

   710 DEF FNC1(P,Q,R,S,T)=INT((-P*(1+R/S)+Q*1.E6/S)/(T*1.45))
   712 DEF FNC2(P,Q,R,S,T)=INT((P*R/S-Q*1.E6/S)/(T*1.45))

!--------------- LOGICAL FUNCTION FOR PANEL SWITCHES ----------------

   715 DEF FNC3(P,Q,R)=(P88#Q)\\(R88#P88#Q)

!------------------ DATA NORMALIZATION FUNCTION ------------------

   717 DEF FNC4(P,Q,R,S)=INT(P*Q*(R-S)/(R*S))


!------------------ TRANSMISSION FUNCTION ------------------------

   720 DEF FNC5(P,Q) = COS(((Q - P)*3.14)/2700)*COS(((Q - P)*3.14)/2700)



!*******************CONTROL PROGRAM*****************************


!--------------------- INITIALIZE PANEL ---------------------

   726 LET RTN=727:GOSUB 5032

!-------------- SET WAVEPLATES; READ INTEGRATOR OFFSETS -------------------

   727 GOSUB 2310
   728 GOSUB 3001

!--------------------- MAIN LOOP ----------------------

   730 LET RTN=751:GOSUB 5000!                                                  SERVICE THE FRONT PANEL

   740 GOSUB 15000!                                                             PROCESS ANY C.R. COMMANDS

   751 IF PBARRY(6)\\PBARRY(7)\\PBARRY(13) GOSUB 761
   752 IF (RUN=0)&&(J=4) GOSUB 2310!                                            SHOT MODE
   753 IF (RUN=0)&&(METAT(32)=-1) GOSUB 3001: GOTO 730!                         MEASURE INT. OFFSETS
   754 IF (J=1)\\(J=4)\\(J=5)  GOSUB 1800!                                      GAIN ADJUST
   755 IF PBARRY(12) LET RUN=1: GOSUB 2310!                                     POSITION BEAM STEERING WAVEPLATES
   756 IF PBARRY(12)&&(AUTO=1)&&(NORM=1) GOSUB 2500:LET SETCBL=1:GOSUB 3700!    SHUTTER,GIMBALS
   757 IF RUN=0 GOTO 730!                                                       STOP
   758 IF NORM=0 GOTO 920!                                                      TEST
   759 IF AUTO=0 GOTO 830!                                                      MANUAL CONTROL
   760 GOTO 770!                                                                AUTO CONTROL
```

```
!------ ------------- SUBROUTINE SELECT --------------------

   761 LET RUN=0:GOSUB 2505



   762 FOR Z=1 TO 10
   763 LET OLDSTA(Z)=OASTAT(Z),OASTAT(Z)=5!                                  STATUS SET TO 'STAND BY'
   764 IF (OASTAT(Z)-OLDSTA(Z)<>0) LET OASTAT(13)=1!                         SET FLAG ON CHANGE
   765 NEXT Z



   768 LET RTN=769:GOSUB 6000
   769 RETURN

!------------------- AUTOMATIC CONTROL --------------------

   770 GOSUB 1000!                                                           SAMPLE ALIGNMENT ERRORS
   780 IF BITLIT(1)+BITLIT(2)+BITLIT(3)+BITLIT(4)+BITLIT(5)+ERROR>0 GOTO 730
   782 LET DBAND=12-7*(J=3)
   785 IF(ABS(AZM(J))<DBAND)&&(ABS(ELV(J))<DBAND)&&(ABS(HORZ(J))<DBAND)&&(ABS(VERT(J))<DBAND) GOTO 810
   790 GOSUB 1940!                                                           DECOUPLE
   800 GOSUB 2110!                                                           COMMAND MOTORS
   810 GOTO 730!                                                             RETURN AND SERVICE THE FRONT PANEL

!--------------------- MANUAL CONTROL ----------------------

   820 LET CONMAN=5*SLEW*50
   832 IF (PBARRY(8)\\PBARRY(9))&&SFLG GOSUB 3700!                           DECLARE ZERO/GOTO ZERO
   835 IF COM3<>0 LET DIST(32)=COM3*(CONMAN),RTN=925:GOSUB 2508
   840 IF (COM1<>0)\\(COM2<>0)<1 GOTO 865
   850 LET ERR(0)=COM1*PNT*(CONMAN),ERR(1)=COM2*PNT*(CONMAN)
   860 LET ERR(2)=COM1*CNT*(CONMAN),ERR(3)=COM2*CNT*(CONMAN)
   870 GOSUB 1980!                                                           DECOUPLE
   880 GOSUB 2110!                                                           COMMAND MOTORS
   885 GOSUB 1000!                                                           SAMPLE ALIGNMENT ERRORS
   890 GOTO 730!                                                             RETURN AND SERVICE THE FRONT PANEL

!--------------------- TEST MODE ----------------------

   920 IF AUTO=1 GOTO 960!                                                   INTERNAL TEST OF INT/ADC
   925 CALL REWIND(4,FLC,0)!                                                 REWIND EXTERNAL TEST FLOPPY
   930 IF FLC=0 LET ERROR=2400,RTN=935:GOSUB 5000
   935 IF FLC<>1 GOTO 960
   940 MORE
   945 GOTO 20000!                                                           EXECUTE EXT. TEST PROGRAM
   950 DEL 20000-29999
   955 FOR Z=0 TO 49:LET BITLIT(Z)=0:NEXT Z
   956 LET RUN=0,NORM=1,TEST=6,RTN=730:GOSUB 6990
   960 LET TEST=#NORM:GOSUB 1000!                                            EXECUTE INT. TEST PROGRAM
   965 LET TEST=0:GOTO 730!                                                  RESET AND RETURN TO SERVICE THE FRONT PANEL
```

```
!--------------------  SUBROUTINE SAMPLE ---------------------

  1000 LET COUNT=0,SAMDEX=8*J
  1001 CALL OASIS(GATE(J),DATA(8*J),FLG,TEST,TRIG(J))!                    INTEGRATOR/ADC HANDLER
  1020 CALL DELAY(DLY(J),EXT,DFLG)!                                       KWV11-A HANDLER - TRIGGER DELAY
  1025 IF DFLG=1 LET BITLIT(0)=1:GOTO 1040
  1026 IF DFLG>1 LET ERROR=2510:RETURN
  1030 LET COUNT=COUNT+1:IF COUNT>50 LET ERROR=2500:RETURN
  1035 GOTO 1025
  1040 IF FLG<>1 LET ERROR=2520:RETURN

  1110 FOR Z=0 TO 7

  1115 LET DATA(SAMDEX+Z)=DATA(SAMDEX+Z)-INTOFF(SAMDEX+Z)!                SUBTRACT OFF NOISE
  1120 NEXT Z

  1130 LET IP(J)=DATA(SAMDEX)+DATA(SAMDEX+1)+DATA(SAMDEX+2)+DATA(SAMDEX+3)!   CALCULATE POINTING INTENSITY
  1135 LET IC(J)=DATA(SAMDEX+4)+DATA(SAMDEX+5)+DATA(SAMDEX+6)+DATA(SAMDEX+7)!   CALCULATE CENTERING INTENSITY

  1200 FOR Z=1 TO 5
  1205 LET BITLIT(Z)=0!                                                   RESET ALIGNMENT LIMIT LIGHTS
  1210 NEXT Z

!                       CHECK IF SIGNAL LEVEL IS WITHIN LIMITS

  1215 IF (IP(J)<200)\\(IP(J)>12000)\\(IC(J)<200)\\(IC(J)>12000) LET BITLIT(4)=1
  1250 IF BITLIT(4)>0 RETURN

!                       CALCULATE ALIGNMENT ERRORS FROM DATA

  1300 LET HORZ(J)=FNC4(N,FM,DATA(SAMDEX+7),DATA(SAMDEX+6))
  1320 LET VERT(J)=FNC4(N,FM,DATA(SAMDEX+4),DATA(SAMDEX+5))
  1340 LET AZM(J)=FNC4(M,FM,DATA(SAMDEX+2),DATA(SAMDEX+3))
  1360 LET ELV(J)=FNC4(M,FM,DATA(SAMDEX),DATA(SAMDEX+1))

!                       CHECK THAT ERRORS ARE WITHIN CONVERGENCE RANGE

  1620 IF ABS(AZM(J))>PRANGE LET BITLIT(5)=1
  1625 IF ABS(ELV(J))>PRANGE LET BITLIT(2)=1
  1630 IF ABS(HORZ(J))>CRANGE LET BITLIT(3)=1
  1635 IF ABS(VERT(J))>CRANGE LET BITLIT(1)=1
```

```
!              UPDATE STATUS ARRAY -- POINTING AND CENTERING PARAMETERS

 1637 LET OLDSTA(11)=OASTAT(11),OLDSTA(12)=OASTAT(12)!                    SAVE VALUES FOR COMPARISON
 1638 LET OASTAT(11)=3,OASTAT(12)=3
 1640 IF POFSTA=1 LET OASTAT(11)=1
 1642 IF COFSTA=1 LET OASTAT(12)=1!                                       CHECK FOR OFFSET LIMITS


 1647 IF (IP(J)<200)\\(IP(J)>12000) LET OASTAT(11)=1
 1650 IF (IC(J)<200)\\(IC(J)>12000) LET OASTAT(12)=1!                     CHECK FOR SIGNAL LIMITS


 1655 IF (ABS(AZM(J))\\ABS(ELV(J)))>PRANGE(J) LET OASTAT(11)=1
 1657 IF (ABS(HORZ(J))\\ABS(VERT(J)))>CRANGE(J) LET OASTAT(12)=1!         CHECK FOR MAXIMUM ALIGNMENT LIMIT

 1660 IF ((OASTAT(11)=1)\\(OASTAT(12)=1)) GOTO 1667

 1662 IF (ABS(AZM(J))\\ABS(ELV(J)))>PTOL(J) LET OASTAT(11)=2
 1665 IF (ABS(HORZ(J))\\ABS(VERT(J)))>CTOL(J) LET OASTAT(12)=2!           CHECK FOR SECONDARY ALIGNMENT LIMIT

 1667 IF (OASTAT(11)-OLDSTA(11)<>0) LET OASTAT(13) = 1
 1670 IF (OASTAT(12)-OLDSTA(12)<>0) LET OASTAT(13) = 1!                   SET CHANGE FLAG ON STATUS CHANGE


 1700 RETURN


!------------------ SUBROUTINE GAIN ADJUST ---------------------

 1800 IF AUTO<>0 RETURN 730!                                              MUST BE IN MANUAL MODE
 1815 LET CONMAN=-(5+SLEW*50)!                                            SLEW OR STEP
 1817 IF (J=1) LET CONMAN=5+SLEW*50!                                      CW ATN IS BACKWARDS


 1820 IF (PBARRY(8)\\PBARRY(9))&3(SFLG\\((ELCID>=288ELCID(=4))) GOSUB 4000!   DECLARE/GOTO ZERO
 1825 IF COM3<>0 LET DIST(CANDEX)=COM3*CONMAN:GOSUB 1875!                 MOVE WP ON COMMAND
 1830 RETURN 730


 1875 LET NS(0)=1
 1880 CALL CMOTOR(NS,MNUMB(CANDEX),RATEDV(CANDEX),DIST(CANDEX))
 1882 CALL SMOTOR(NS,MNUMB(CANDEX),CDIST(CANDEX),CPOSIT(CANDEX),CTRAN(CANDEX),MSTAT(CANDEX))
 1885 LET BITLIT(1)=(MSTAT(CANDEX)=1\\MSTAT(CANDEX)=-2)
 1887 LET BITLIT(2)=(MSTAT(CANDEX)=-1\\MSTAT(CANDEX)=-2)
 1890 IF NS(5)<>0 LET RTN=1882:GOSUB 6000
 1892 LET MOTOR(CANDEX)=MOTOR(CANDEX) + DIST(CANDEX)!                     UPDATE RELATIVE MOTOR POSITIONS
 1895 RETURN
```

```
!-------------------- SUBROUTINE DECOUPLE ---------------------

!                    This subroutine calculates the proper distances to move gimbal
!               motors in order to correct alignment errors while correcting for
!               cross-coupling between pointing and centering motions.

  1940 RESTORE 1950!                                                    PREPARE TO READ CURRENT ALIGNMENT ERRORS
  1950 DATA AZM(J),ELV(J),HORZ(J),VERT(J)!                              CURRENT ALIGNMENT ERRORS

  1960 FOR Z=0 TO 3
  1965 READ ERR(Z)!                                                     READ ALIGNMENT ERRORS
  1970 NEXT Z

  1980 LET CPLDEX=4*J,HAFDEX=CPLDEX/2!                                  SET UP INDICES

!                Adjust alignment errors by the proper magnification.

  1982 LET ERR(0)=ERR(0)*HAZMAG(J),ERR(1)=ERR(1)*VELMAG(J)
  1985 LET ERR(2)=ERR(2)/HAZMAG(J),ERR(3)=ERR(3)/VELMAG(J)

!                Calculate motor distances to correct alignment errors.

  2000 LET DIST(CPLDEX)=FNC1(ERR(0),ERR(2),GIMPOS(HAFDEX),GIMPOS(HAFDEX+1),2)
  2020 LET DIST(CPLDEX+1)=FNC1(ERR(1),ERR(3),GIMPOS(HAFDEX),GIMPOS(HAFDEX+1),-1.414)
  2040 LET DIST(CPLDEX+2)=FNC2(-ERR(0),-ERR(2),GIMPOS(HAFDEX),GIMPOS(HAFDEX+1),2)
  2060 LET DIST(CPLDEX+3)=FNC2(ERR(1),ERR(3),GIMPOS(HAFDEX),GIMPOS(HAFDEX+1),-1.414)

  2080 RETURN

!-------------------- SUBROUTINE MOTORS -----------------------

!                    This subroutine performs the necessary calls to the assembly
!               language handler MOTORS for moving and obtaining the status of OAS
!               gimbal motors: panel light variables are set when limit switches are
!               encountered and the "motor position since last declare" is updated.

  2110 LET NS(0)=4,MOTDEX=4*J!                                          USE FOUR MOTORS; SET INDEX
! ******* IF DISTANCE TO GO IS LARGE, REDUCE SYSTEM RESPONSE BY DIVIDING NUMBER OF STEPS BY 5. ******** (12/15/78 JP)

  2112 IF DIST(MOTDEX)>4000 LET DIST(MOTDEX)=DIST(MOTDEX)/5
  2113 IF DIST(MOTDEX+1)>1000 LET DIST(MOTDEX+1)=DIST(MOTDEX+1)/5
  2114 IF DIST(MOTDEX+2)>4000 LET DIST(MOTDEX+2)=DIST(MOTDEX+2)/5
  2115 IF DIST(MOTDEX+3)>1000 LET DIST(MOTDEX+3)=DIST(MOTDEX+3)/5

  2120 CALL CMOTOR(NS,MNUMB(MOTDEX),RATEDV(MOTDEX),DIST(MOTDEX))!       MOVE MOTORS
  2140 CALL SMOTOR(NS,MNUMB(MOTDEX),CDIST(MOTDEX),CPOSIT(MOTDEX),CTRAN(MOTDEX),MSTAT(MOTDEX))!     GET STATUS OF MOTORS

  2240 FOR Z=0 TO 3
  2242 LET BITLIT(2*Z+22)=(MSTAT(MOTDEX+Z)=1)\\(MSTAT(MOTDEX+Z)=-2)!    SET PROPER LIGHT BIT
  2244 LET BITLIT(2*Z+23)=(MSTAT(MOTDEX+Z)=-1)\\(MSTAT(MOTDEX+Z)=-2)!   IF LIMIT ENCOUNTERED
  2248 NEXT Z

  2249 IF NS(5)<>0 LET RTN=2140:GOSUB 6000!                             IF MOTOR POWER STILL ON, UPDATE PANEL

  2250 FOR Z=22 TO 29:IF BITLIT(Z)=1 LET ERROR=2700!                    SET ERROR CODE IF LIMIT ENCOUNTERED
  2255 FOR Z=0 TO 3
  2260 LET MOTOR(MOTDEX+Z)=MOTOR(MOTDEX+Z)+DIST(MOTDEX+Z)!              UPDATE RELATIVE MOTOR POSITIONS
  2265 NEXT Z
```

```
!------ UPDATE STATUS ARRAY (GIMBALS) ------

  2267 FOR Z = 0 TO 3
  2270 LET TEMP(Z) = 5 - (ABS(MSTAT(MOTDEX+Z))+1)
  2272 NEXT Z

  2273 LET TGDAT1 = TEMP(0)&TEMP(1)
  2274 LET TGDAT2 = TEMP(2)&TEMP(3)

  2275 IF (J=1)\\(J=4)\\(J=5) GOTO 2306!                          SKIP GIMBAL UPDATE -- ATN MODE
  2276 IF (J=3) GOTO 2280   !                                    PULSE MODES
  2277 IF (J=0)\\(J=2)\\(J=7) GOTO 2287!                         CW OR PSS MODE
  2278 GOTO 2306

  2280 LET OLDSTA(6) = OASTAT(6),OLDSTA(10)=OASTAT(10)!          SAVE VALUES FOR COMPARE
  2282 LET OASTAT(6)=TGDAT1,OASTAT(10)=TGDAT2!                   STATUS IS MIN OF MOTOR STATS
  2284 IF(OASTAT(6)-OLDSTA(6)<>0)\\(OASTAT(10)-OLDSTA(10)<>0) LET OASTAT(13)=1
  2285 GOTO 2306


  2287 LET OLDSTA(7)=OASTAT(7)
  2289 LET OASTAT(7)=TGDAT1
  2290 IF(OASTAT(7)-OLDSTA(7)<>0) LET OASTAT(13)=1!              FIRST GIMBAL

  2292 IF (J=7) GOTO 2301!                                       PSS MODE

  2294 LET OLDSTA(8)=OASTAT(8)
  2296 LET OASTAT(8)=TGDAT2
  2298 IF(OASTAT(8)-OLDSTA(8)<>0) LET OASTAT(13)=1!              2ND GIMBAL IF CW

  2300 GOTO 2306

  2301 LET OLDSTA(9)=OASTAT(9)
  2302 LET OASTAT(9)=TGDAT2
  2304 IF(OASTAT(9)-OLDSTA(9)<>0) LET OASTAT(13)=1!              2ND GIMBAL IF PSS

  2306 RETURN



!------------------- SUBROUTINE WAVEPLATE ----------------------

  2310 LET NS(0)=4
  2320 CALL SMOTOR(NS,MNUMB(33),CDIST(33),CPOSIT(33),CTRAN(33),MSTAT(33))
  2325 IF (J=4)\\(J=5)&&(RUN=0) GOTO 2355
  2330 FOR Z=0 TO 3
  2335 LET DIST(33+Z)=30*ROT(8*Z+J)+WOFF(Z)-CPOSIT(33+Z)
  2340 NEXT Z
  2345 LET DIST(36)=SETG(36)-CPOSIT(36)
  2350 GOTO 2360

  2355 LET DIST(33)=SETG(33)-CPOSIT(33)
  2357 LET DIST(34)=SETG(34)-CPOSIT(34)
  2360 CALL CMOTOR(NS,MNUMB(33),RATEDV(33),DIST(33))
  2370 CALL SMOTOR(NS,MNUMB(33),CDIST(33),CPOSIT(33),CTRAN(33),MSTAT(33))
  2390 IF NS(5)<>0 LET RTN=2370:GOSUB 6000
```

```
!------ UPDATE STATUS ARRAY ---

  2395 FOR Z=1 TO 4
  2400 LET OLDSTA(Z) = OASTAT(Z)
  2405 LET OASTAT(Z) = MSTAT(37-Z) + 3
  2410 IF (OASTAT(Z) - OLDSTA(Z) <> 0) LET OASTAT(13) = 1
  2415 NEXT Z

  2430 RETURN

!-------------------- SUBROUTINE ATTENUATOR --------------------

!                    This subroutine allows control of the OAS shutter, located
!                in the alignment sensor.  There are three entry points depending
!                upon whether the current operating mode is automatic, manual, or
!                changing.  The shutter is automatically closed whenever the OAS
!                operating mode (beam line or gain adjust) is changed.
!                    An update is done of the status array whenever a change
!                is detected in the status of the attenuator motor.

  2500 LET NS(0)=1,DIST(32)=ATTEN(J)-CPOSIT(32):GOTO 2510
  2505 LET NS(0)=1,DIST(32)=-2000:GOTO 2510
  2508 LET NS(0)=1 : GOTO 2510 ! FOR MANUAL CONTROL
  2509 LET NS(0)=1,DIST(32)=SETC(32)-CPOSIT(32) : GOTO 2510
  2510 CALL CMOTOR(NS,MNUMB(32),RATEDV(32),DIST(32))
  2520 CALL SMOTOR(NS,MNUMB(32),CDIST(32),CPOSIT(32),CTRAN(32),MSTAT(32))
  2527 LET BITLIT(30)=(MSTAT(32)=1)\\(MSTAT(32)=-2)
  2528 LET BITLIT(31)=(MSTAT(32)=-1)\\(MSTAT(32)=-2)
  2530 IF NS(5)<>0 LET RTN=2520:GOSUB 6000
  2532 LET MOTOR(32) = MOTOR(32) + DIST(32)!                          UPDATE RELATIVE MOTOR POSITION

!------ UPDATE STATUS ARRAY ------

  2535 LET OLDSTA(5) = OASTAT(5)
  2540 LET OASTAT(5) = MSTAT(32) + 3
  2545 IF (OASTAT(5) - OLDSTA(5) <> 0) GOTO 2555
  2550 LET OASTAT(13) = 1


  2555 RETURN

!-------------------- SUBROUTINE INT. OFFSETS ----------------------

!                    This subroutine samples the offsets in the Integrator/ADC
!                by performing a call to the handler OASIS while the shutter in the
!                alignment sensor is fully closed, i.e., the signal level is zero.

  3001 LET OFFDEX=8*J!                                                SET UP INDEX
  3005 CALL OASIS(GATE(J),INTOFF(OFFDEX),FLG,0,TRIG(J))!              GET OFFSETS
  3020 CALL DELAY(DLY(J),0,DFLG)!                                     WITH PROPER TRIGGER DELAY
  3030 IF (J=1)\\(J=4) GOTO 3090!                                     CHECK IF IN ATTENUATOR MODE
  3050 IF FLG<>1 LET ERROR=10:RETURN!                                 CHECK FOR TIMEOUT IN INTEGRATOR/ADC
  3090 FOR Z=0 TO 3:IF (INTOFF(OFFDEX+Z)<1)\\(INTOFF(OFFDEX+Z)>75) LET ERROR=20:NEXT Z
  3100 FOR Z=4 TO 7:IF(INTOFF(OFFDEX+Z)<1)\\(INTOFF(OFFDEX+Z)>75) LET ERROR=25:NEXT Z
  3105 LET POFSTA=0,COFSTA=0
  3110 IF ERROR=20 LET POFSTA=1!                                      SET FLAG IF EXCESSIVE POINTING OFFSETS
  3115 IF ERROR=25 LET COFSTA=1!                                      SET FLAG IF EXCESSIVE CENTERING OFFSETS
  3120 RETURN
```

```
!---------------- SUBROUTINE DECLARE ZERO/GOTO ZERO --------------------

  3700 LET NS(0)=4,ZRODEX=4*J
  3705 CALL SMOTOR(NS,MNUMB(ZRODEX),CDIST(ZRODEX),CPOSIT(ZRODEX),CTRAN(ZRODEX),MSTAT(ZRODEX))
  3715 IF (PBARRY(9)\\SETGBL) GOTO 3725
  3720 FOR Z=0 TO 3:LET ZERO(ZRODEX+Z)=CPOSIT(ZRODEX+Z),MOTOR(ZRODEX+Z)=0:NEXT Z
  3722 LET ATTEN(J)=CPOSIT(32), MOTOR(32) = 0
  3723 GOTO 3730
  3725 FOR Z=0 TO 3:LET DIST(ZRODEX+Z)=ZERO(ZRODEX+Z)-CPOSIT(ZRODEX+Z):NEXT Z:GOSUB 2110
  3726 GOSUB 2500!                                                    RESET SENSOR ATTENUATOR
  3727 LET SETGBL = 0
  3730 RETURN

 !----------------- SUBROUTINE -- DECLARE/GOTO CAIN POINT ----------------

  4000 LET NS(0)=1
  4010 CALL SMOTOR(NS,MNUMB(CANDEX),CDIST(CANDEX),CPOSIT(CANDEX),CTRAN(CANDEX),MSTAT(CANDEX))
  4020 IF PBARRY(9) GOTO 4050
  4030 LET SETG(CANDEX)=CPOSIT(CANDEX),MOTOR(CANDEX)=0
  4040 GOTO 4070
  4050 LET DIST(CANDEX)=SETG(CANDEX)-CPOSIT(CANDEX),RTN=4060:GOSUB 1875
  4070 RETURN

!---------------- SUBROUTINE FRONT PANEL SERVICE --------------------
                                  80
!                  This subroutine is used for updating the OIS LSI-11 control
!               panel and includes code to interlock with the second user (CONTROL
!               ROOM OPERATIONS), copy command switches into PBARRY from the CMDTBL, determine
!               logical combinations of control panel switches for logical variables,
!               and calculate pointers to data for proper display selection.  Whenever
!               possible, this routine is called to update the control panel so
!               that operator interaction is effected regardless of other control operations
!               in progress.
  5000 LET BITLIT(6)=0,BITLIT(7)=(ERROR>2000)!                        RESET OPERATING LIGHT; SET ERROR LIGHT IF ER
  5003 CALL LPANEL(BITLIT,NB,0,DIGLIT,NC,0,0)!                       UPDATE THE PANEL LIGHTS

!       If no command from the control room, get switches from the control panel.

  5005 IF CMDFLG = 0 GOTO 5020

!       Otherwise, copy commands into the push button array.

  5007 CALL ARCOPY(40,CMDTBL,PBARRY)
  5008 GOSUB 10000!                                                   DECODE COMMAND FROM CONTROL ROOM
  5010 LET CMDFLG = 0!                                                RESET INTERLOCKING FLAG
  5012 RESUME WITH CMD!                                               RESTART SECOND USER
  5015 GOTO 5032!                                                     PROCEED WITH LOGICAL CALCULATIONS

  5020 CALL SPANEL(SFLG,CONFIG,PBARRY,NPB,DCARRY,NC)!                 GET INFO FROM CONTROL PANEL
!       Calculate the oscillator alignment mode selected.

  5032 LET BITLIT(J + 14) = 0!                                        TURN OFF PRESENT SELECT LIGHT

  5035 LET J = J - PBARRY(6) + PBARRY(7)!                             ADJUST J TO NEW VALUE
  5040 LET J = J + (J<0)*8 - (J>7)*8!                                 MAKE J MODULO 8
  5045 LET BITLIT(J+14) = 1!                                          TURN ON NEW SELECT LIGHT

  5050 LET OASTAT(0)=J!                                               UPDATE STATUS ARRAY

  5060 IF (J=1)\(J=4) GOTO 5070!                                      CHECK IF EITHER ATN MODE HAS BEEN SELECTED
  5062 IF (J=5) GOTO 5072!                                            DYE ATN MODE
  5065 GOTO 5100!                                                     IF NOT, CONTINUE
```

```
5070 LET GANDEX = 37 - J!                                   INDEX FOR ATN (EXCEPT DYE) MOTORS
5071 GOTO 5075
5072 LET CANDEX = 34!                                       INDEX FOR DYE ATN MOTOR
5075 FOR Z = 8 TO 13!                                       \
5080 LET BITLIT(Z) = 0,BITLIT(Z-8) = 0!                     CLEAR LIMIT LIGHTS, DISPLAY CONTROL LIGHTS
5085 NEXT Z!                                                /
5095 GOTO 6000!                                             DO LOGICAL CALCULATIONS; SET PROPER LIGHTS

5100 IF (SFLG=0) GOTO 6000!                                 IF NO CHANGE ON PANEL, CHECK MOMENTARY SWITC

!              If any of the Display Select push buttons have been pushed, go
!       light the correct light (above/below the push button).

5140 IF PBARRY(0) GOTO 5300!                                ALIGNMENT ERRORS
5145 IF PBARRY(1) GOTO 5300!                                MOTOR POSITIONS
5150 IF PBARRY(2) GOTO 5300!                                CENTERING DATA (INT/ADC
5155 IF PBARRY(3) GOTO 5300!                                POINTING DATA (INT/ADC)
5160 IF PBARRY(4) GOTO 5300!                                CENTERING OFFSETS (INT/ADC)
5165 IF PBARRY(5) GOTO 5300!                                POINTING OFFSETS (INT/ADC)

!              If any of the momentary buttons have been pushed, set up the
!       pointer to read the correct data for the digital display; otherwise,
!       continue with calculating logical variables.

5180 IF PBARRY(28) GOTO 5215!                               DETECTOR OFFSETS-N/A AT PRESENT
5185 IF PBARRY(29) GOTO 5215!                               INTEGRATOR GATE WIDTH
5190 IF PBARRY(30) GOTO 5215!                               ERROR CODE
5195 IF PBARRY(31) GOTO 5215!                               TRIGGER DELAY
5200 IF PBARRY(32) GOTO 5215!                               ATTENUATOR POSITION

5210 GOTO 6000

!              One of the momentary push buttons has been pushed; turn off
!       the current display indicator, find out which momentary button has
!       been pushed, set up the pointer to the correct data, read the data
!       and exit.

5215 LET BITLIT(TDC+8)=0!                                   TURN OFF DISPLAY INDICATOR
5220 FOR Z=28 TO 32
5225 IF PBARRY(Z) LET MDC = Z, Z = 32!                      MDC IS THE MOMEN. BUTTON PUSHED
5230 NEXT Z

5240 RESTORE 8000+MDC-28!                                   SET POINTER TO DATA
5250 FOR Z=4*(MDC<>28) TO 4!                                \
5260 READ DIGLIT(Z)!                                        "READ THE DATA
5270 NEXT Z!                                                /
5280 GOTO 9000!                                             RESET DISPLAY INDICATOR LIGHTS

5300 LET BITLIT(TDC+8) = 0!                                 TURN OFF DISPLAY INDICATOR
5310 FOR Z=0 TO 5
5320 IF PBARRY(Z) LET TDC = Z,BITLIT(Z + 8) = 1,Z = 5
5330 NEXT Z!                                                SET POINTER TO DATA, SET PROPER INDICATOR LI
```

```
!                       TURN ON NETWORK OR LOCAL LIGHTS

!               The following section calculates a number of logical variables
!       used in the main control loop.  Most variables are functions of push button
!       values and the present value of the variable.  Light values are mostly a
!       function of push button and variable values.  FNC3 is a logical function
!       defined in line 717.

  6000 LET RUN=FNC3(PBARRY(12),PBARRY(13),RUN)
  6050 LET BITLIT(34) = RUN, BITLIT(35) = #RUN!                    TURN ON THE PROPER 'RUN' LIGHT
  6100 LET NET=FNC3(PBARRY(10),PBARRY(11),NET)


  6150 LET BITLIT(32) = NET, BITLIT(33) = #NET!                    TURN ON NETWORK OR LOCAL LIGHTS

  6200 LET AUTO=FNC3(PBARRY(14),PBARRY(15),AUTO)
  6250 LET BITLIT(36) = AUTO, BITLIT(37) = #AUTO!                  TURN ON AUTO OR MANUAL LIGHT

  6300 LET NORM=FNC3(PBARRY(16),PBARRY(17),NORM
  6350 LET BITLIT(38) = NORM, BITLIT(39) = #NORM!                  TURN ON NORMAL OR TEST MODE LIGHT

  6400 IF (AUTO=1)\\(RUN=0) GOTO 6420!                             IF AUTO OR STOP, TURN OFF MANUAL SECTION LIG
  6410 GOTO 6500!                                                  OTHERWISE, CONTINUE WITH CALCULATIONS
  6420 FOR Z = 40 TO 49!                                           \
  6440 LET BITLIT(Z) = 0!                                          "TURN OFF MANUAL LIGHTS
  6450 NEXT Z!                                                     /
  6460 GOTO 8500!                                                  READ DISPLAY DATA AND EXIT

  6500 LET SLEW=FNC3(PBARRY(24),PBARRY(25),SLEW)
  6520 LET BITLIT(46) = SLEW, BITLIT(47) = #SLEW!                  TURN ON SLEW OR STEP LIGHT

!               If one of 'ATN' modes has been selected, turn off the lights for
!       'CENTERING/POINTING', '+/- VERTICAL/ELEVATION', '+/- HORIZONTAL/AZIMUTH'.

  6600 IF (J=1)\\(J=4)\\(J=5) GOTO 6620
  6610 GOTO 6700
  6620 FOR Z = 40 TO 45
  6630 LET BITLIT(Z) = 0
  6640 NEXT Z
  6650 GOTO 6810

  6700 LET CNT=FNC3(PBARRY(18),PBARRY(19),CNT)
  6710 LET BITLIT(40) = CNT, BITLIT(41) = #CNT, PNT = #CNT!        SET 'CENT/POINT' LIGHTS AND VARIABLES
```

```
!              In the following, COM1,COM2, AND COM3 are variables which indicate
!       which of the motor control buttons in the manual control section have been
!       pushed.  These refer to '+/- HORIZ/AZM', '+/- VERT/ELV', and 'OPEN/CLOSE'
!       (ATTEN), RESPECTIVELY.

  6750 LET COM1=PBARRY(20)-PBARRY(21)
  6770 LET BITLIT(42) = PBARRY(20), BITLIT(43) = PBARRY(21)
  6790 LET COM2=PBARRY(22)-PBARRY(23)
  6800 LET BITLIT(44) = PBARRY(22), BITLIT(45) = PBARRY(23)

  6810 LET COM3=PBARRY(26)-PBARRY(27)
  6820 LET BITLIT(48) = PBARRY(26), BITLIT(49) = PBARRY(27)
  6825 IF (J=1)\\(J=4)\\(J=5) GOTO 8600!                          IF ATN MODE CALCULATE TRANSMISSION
!              If any of the momentary buttons are being held, update the panel
!       and exit.

  6830 IF PBARRY(28)\\PBARRY(29)\\PBARRY(30)\\PBARRY (31)\\PBARRY(32) GOTO 9000

!       Otherwise, read the digital display data, update panel, and exit.

  6835 GOTO 8500

!***************DATA FOR DIGITAL DISPLAYS*****************************

!---------------DATA FOR CONTINUOUS DISPLAY (TOGGLE SWITCHES)--------


  7000 DATA VERT(J),HORZ(J),ELV(J),AZM(J),INT((IP(J)+IC(J))/160)!   ALIGNMENT ERRORS, SIGNAL LEVEL
  7001 DATA MOTOR(MPLDEX),MOTOR(MPLDEX+1),MOTOR(MPLDEX+2),MOTOR(MPLDEX+3),CPOSIT(32)!   MOTOR POS. SINCE DECLR.,ATTEN. POS.
  7002 DATA DATA(PNLDEX+4),DATA(PNLDEX+5),DATA(PNLDEX+6),DATA(PNLDEX+7),INT(IC(J)/80)!   CENTERING DATA,CENTERING INTENSITY
  7003 DATA DATA(PNLDEX),DATA(PNLDEX+1),DATA(PNLDEX+2),DATA(PNLDEX+3),INT(IP(J)/80)!     POINTING DATA, POINTING INTENSITY
  7004 DATA INTOFF(PNLDEX+4),INTOFF(PNLDEX+5),INTOFF(PNLDEX+6),INTOFF(PNLDEX+7),TRIG(J)!INT. OFFSETS (CENTERING),TRIG SELECTED
  7005 DATA INTOFF(PNLDEX),INTOFF(PNLDEX+1),INTOFF(PNLDEX+2),INTOFF(PNLDEX+3),TRIG(J)!   INTEGRATOR OFFSETS, TRIGGER SELECTED


!-------------------------------------------------------------------

!---------------DATA FOR MOMENTARY DISPLAY (PUSH BUTTONS)------------

  8000 DATA VOFF(J),HOFF(J),ELOFF(J),AZOFF(J),50!                 OFFSETS FOR ABSOLUTE POSITION,NOMINAL INTENS
  8001 DATA GATE(J)!                                              INTEGRATOR SAMPLE WIDTH
  8002 DATA ERROR!                                                ERROR CODE
  8003 DATA DLY(J)!                                               TRIGGER DELAY
  8004 DATA 100*(SIN(CPOSIT(32)*3.14/2700+.01))*(SIN(CPOSIT(32)*3.14/2700+.01))!   ATTENUATOR POSITION
  8010 DATA 100*STARTT,BLANK,100*PREST,BLANK,((PREST-STARTT)/STARTT)*100!   START TRANSMISSION,PRESENT TRANS.,% CHANGE
  8015 DATA ERROR!                                                ERROR CODE

!-------------------------------------------------------------------

!*******************************************************************


  8500 RESTORE 7000+TDC:LET BITLIT(TDC+8)=1,PNLDEX=8*J,MPLDEX=PNLDEX/2!   POINT TO DATA SELECTED FOR DISPLAY
  8520 FOR Z=0 TO 4
  8523 READ DIGLIT(Z)                                             READ THE CHOSEN DATA
  8525 NEXT Z!
  8530 GOTO 9000!                                                 RESET TRIG LIGHT,ERROR CODE;
!                                                                 SET OP LIGHT AND EXIT
```

```
!       CALCULATE THE STARTING AND PRESENT TRANSMISSION FOR ATN MODES

 8600 LET STRPLS = FNC5(WOFF(0),SETC(33)), PTRPLS = FNC5(WOFF(0),CPOSIT(33))!          START AND PRESENT TRANS. (PLS)
 8601 LET STRDYE = FNC5(WOFF(1),SETC(34)), PTRDYE = FNC5(WOFF(1),CPOSIT(34))!          START AND PRESENT TRANS. (DYE)
 8602 LET STRCW = FNC5(WOFF(3),SETC(36)), PTRCW = FNC5(WOFF(3),CPOSIT(36))!            START AND PRESENT TRANS. (CW)
 8603 LET STRSWP = FNC5(WOFF(2),SETC(35)), PTRSWP = FNC5(WOFF(2),CPOSIT(35))!          START AND PRESENT TRANS. (STEERING WP)

 8605 IF (J=1) LET STARTT = STRCW, PREST = PTRCW
 8606 IF (J=4) LET STARTT = STRPLS, PREST = PTRPLS
 8607 IF (J=5) LET STARTT = STRDYE, PREST = PTRDYE


 8609 RESTORE 8010!                                                                    POINT TO ATN MODE DATA


 8610 FOR Z = 0 TO 4
 8615 READ DIGLIT(Z):  NEXT Z!                                                         READ ATN MODE DATA

 8620 IF PBARRY(30) READ DIGLIT(4)!                                                    READ AATTENUATOR POS. IF REQUESTED

!       RESET TRIGGER LIGHT, ERROR CODE; SET OPERATING LIGHT; THEN EXIT

 9000 LET BITLIT(0)=0,ERROR=0,BITLIT(6)=1
 9008 CALL LPANEL(BITLIT,NB,0,DIGLIT,NG,0,0)
 9010 RETURN RTN


!               The following subroutine decodes a command sent from the control room
!       to the OAS LSI-11.  Commands such as changing the oscillator under control,
!       RUNning the system (same as pushing RUN on the control panel), or setting any
!       of the system attenuator positions are considered.  In some cases, a flag
!       (ELCID - the 'Command ID' (heh-heh)) is used so that the operation may be
!       initiated from the main control loop rather than from the front panel service
!       routine, thus preserving the control program philosophy. (2/16/79)


!-------------------------------------------------------------------------------

 10000 GOTO (11900 + 100 * PBARRY(33))!                                               PROCESS COMMAND

!-------------------------------------------------------------------------------

 12000 LET BITLIT(J + 14) = 0!                                                        TURN OFF SELECT LIGHT
 12002 LET J = PBARRY(34)!                                                            NEW OSCILLATOR SELECTED
 12005 LET PBARRY(13) = 1!                                                            STOP SYSTEM-DOUBLE CHECKS C.R.
 12007 IF (DEBUG1) ENTER
 12010 RETURN

!-------------------------------------------------------------------------------
```

```
12100 IF (J<>4) LET ERROR = 2200 : RETURN!                          TRIED TO COMMAND PLS ATN
!                                                                   W/O SELECTING THAT MODE
12102 IF (#PBARRY(9)\\#PBARRY(15)) LET ERROR = 2205 : RETURN!       TRIED TO COMMAND PLS ATN BUT
!                                                                   C.R. SENT INCORRECT PBARRY
12103 IF #RUN LET ERROR = 2207 : RETURN!                            MUST BE IN RUN STATE
12105 LET SETG(33) = CMDTBL(18)!                                    WORD CONTAINING CALCULATED DIST.
12107 LET GANDEX = 33, AUTO = 0!                                !   SET MOTOR INDEX AND MANUAL MODE
12110 LET ELCID = 2!                                                COMMAND ID FOR USE IN MAIN LOOP
12112 IF (DEBUG1) ENTER
12115 RETURN

!-------------------------------------------------------------------------------

12200 IF (J<>5) LET ERROR = 2210 : RETURN!                          TRIED TO COMMAND DYE ATN
!                                                                   W/O SELECTING THAT MODE
12202 IF (#PBARRY(9)\\#PBARRY(15)) LET ERROR = 2215 : RETURN!       TRIED TO COMMAND DYE ATN BUT
!                                                                   C.R. SENT INCORRECT PBARRY
12203 IF #RUN LET ERROR = 2217 : RETURN!                            MUST BE IN RUN STATE
12205 LET SETG(34) = CMDTBL(18)!                                    WORD CONTAINING CALCULATED DIST.
12207 LET GANDEX = 34, AUTO = 0!                                    SET MOTOR INDEX AND MANUAL MODE
12210 LET ELCID = 3!                                                COMMAND ID FOR USE IN MAIN LOOP
12212 IF (DEBUG1) ENTER
12215 RETURN

!-------------------------------------------------------------------------------

12300 IF (J<>1) LET ERROR = 2220 : RETURN!                          TRIED TO COMMAND CW ATN
!                                                                   W/O SELECTING THAT MODE
12302 IF (#PBARRY(9)\\#PBARRY(15)) LET ERROR = 2225 : RETURN!       TRIED TO COMMAND CW ATN BUT
!                                                                   C.R. SENT INCORRECT PBARRY
12303 IF #RUN LET ERROR = 2227 : RETURN!                            MUST BE IN RUN STATE
12305 LET SETG(36) = CMDTBL(18)!                                    WORD CONTAINING CALCULATED DIST.
12307 LET GANDEX = 36, AUTO = 0!                                    SET MOTOR INDEX AND MANUAL MODE
12310 LET ELCID = 4!                                                COMMAND ID FOR USE IN MAIN LOOP
12312 IF (DEBUG1) ENTER
12315 RETURN

!-------------------------------------------------------------------------------

12400 FOR Z = 0 TO 7
12403 IF PBARRY(10 + Z) GOTO 12410!                                 CHECK AT LEAST 1 PB IS ON
12405 NEXT Z
12407 LET ERROR = 2230 : RETURN!                                    MODCTL FROM CR BUT WRONG PBARRY

12410 IF (DEBUG1) ENTER
12415 RETURN

!-------------------------------------------------------------------------------
```

```
12500 IF (#PBARRY(15)&&AUTO) LET ERROR = 2235 : RETURN!                    MUST BE IN OR CHANGING TO MANUAL MODE

12510 IF (DEBUG1) ENTER
12550 RETURN

!-------------------------------------------------------------------------

12600 IF (#PBARRY(8)&&#PBARRY(9))\\AUTO LET ERROR = 2240 : RETURN!         CHECK FOR MANUAL AND CORRECT PB'S

12605 IF (DEBUG1) ENTER
12620 RETURN

!-------------------------------------------------------------------------

12700 IF (#PBARRY(15)&&AUTO) LET ERROR = 2245: RETURN
12703 IF #RUN LET ERROR = 2247 : RETURN!                                   MUST BE IN RUN AND MANUAL

12705 LET SETC(32) = CMDTBL(18)!                                           WORD CONTAINING CALC. DIST
12707 LET AUTO = 0, ELCID = 5!                                             COMMAND ID FOR USE IN MAIN LOOP

12710 IF (DEBUG1) ENTER

12750 RETURN

!-------------------------------------------------------------------------


!*************************************************************************
!           Subroutine to process control room commands: called from main loop
!       this subroutine carries out commands sent from the control room by inter-
!       preting the command ID (ELCID).

15000 IF ((ELCID >= 2) && (ELCID <= 4)) GOSUB 1800!                        ATN MODES ('GO TO POSITION')

15010 IF ELCID = 5 GOSUB 2509!                                             SET SENSOR ATTENUATOR FROM CR
15999 RETURN

!*************************************************************************

EOT
```

```
! OAS SYSTEM CONTROL ROOM COMMUNICATION ACTIVITY
!       Calculate the oscillator alignment mode selected.

! This program runs concurrently with the Oscillator Alignment System
! (OAS) control loop program in the OAS 'A' processor LSI-11, and
! provides communications to the control room PDP-11/34 for both
! status and command type messages.
!
! A separate program (ie. USER) is employed for this operation for the
! following reasons:
!
!       1.      This program can be run on the control room 'A' processor
!               in a test version without the normal OAS control loop
!               to check out the PDP-11/34 software, without disturbing the
!               operating OAS system.
!
!       2.      Insures flexibility to change 'packed' contents of the
!               communication arrays to suit the PDP-11/34 software
!               without modifying the main control programs.
!
!       3.      Nearly asynchronous operations for status requests.
!
!       4.      Double buffering of commands is provided.  Processor
!               will continue to respond to status requests from the
!               control room even though a command previously sent has
!               not been completely processed by the control loop.
!
!       5       Capability of providing combined 'command-status'
!               requests  in one message.
!
!       6       Flexibility to do (future) unrequested updates of status
!               changes to the control room via 'master-master' mode of
!               Shivanet.
!
!
! To interlock the operations of this program and the OAS control loop,
! the following additions will be made to the OAS control loop program
! (see block diagram):
!
!       1.      At a reasonable point where remote commands can be
!               processed, the variable "CMDFLC" (which is set by the
!               control room communication program when a request is ready)
!               will be tested.
!               a)      If zero, no further action is required.
!               b)      IF not zero, the value is taken as a type of
!                       request code.  The parameters defined for that
!                       particular type of request are then transfered
!                       into the proper control system parameters from
!                       the command table "CMDTBL".
!                       The CMDFLC is then set to zero, and a
!                       RESUME WITH CMD is issued to resume the commun-
!                       ication program.
!       2.      If there are places within the control loop where context
!               switching to the communication activity might cause the
!               picking up of erroneous status information, those sections
!               of code should be interlock with a LOCK STAT and UNLOCK STAT
!               pair.  This interlock interval should be minimized to allow
!               as rapid average 11/34 access as possible.
!
```

```
!
!
! The command and response messages processed by this program are formatted
! according to the definitions in the Parameter Description File (OAS.PDE).
!
!
!
CMD USE 4=DX0,5=DX0
OLD/CN
CMD USE 2=KB0

    10 RESUME "OAS"

   11 ACTIVITY CNTLRM
!
! Set up common, equates, and initial conditions
!
   20 GOSUB 9000
!
!****************** PDP-11 COMMUNICATION LOOP **********************
!
!
! Send response, post read, and wait for next request message
!
  300 CALL SLINK (UNIT, NUMS, NUMR, SARRAY, RARRAY, LKSTAT, WAITFG)
  330 SUSPEND UNLESS WAITFG
!
! A message has been recieved from the control room, determine type
!

   400 FOR Z = 0 TO 1
   402 LET SARRAY(Z) = RARRAY(Z+2)
   405 LET SARRAY(Z+2) = RARRAY(Z)
   407 NEXT Z


  410 LET TYPE=RARRAY(4)
  415 LET SARRAY(4) = RARRAY(4)!                    ECHO RESPONSE REQUEST
  416 LET SARRAY(5) = 077!                          PAD SEND ARRAY FOR UNPACKING LATER
  417 IF (DEBUG2) ENTER
  420 IF (TYPE-1) GOTO 2000
!
!--------------------- PROCESS A STATUS REQUEST ----------------------
!
 1000 IF TYPE = 1 LET OASTAT(13) = 1
 1010 LET TYPE=ABS(TYPE)
!
! Check for valid type code, return null response if not valid
!
 1020 IF ((TYPE>MAXSTY)\\(TYPE=0))  LET NUMS=1, SARRAY(4)=0 : GOTO 1095
!
!
! Temporary lock out access by control loop program to status variables
! and go to subroutine to process this type
!
 1070 GOSUB (1010 + TYPE * 100)

 1095 GOTO 300
```

```
!
! Subroutine to process TYPE -1 status request

!
! Pass back status variables, motor positions, and alignment errors.
 1100 !
 1110 LET NBTC=14
 1122 CALL ARCOPY (NBTC, OASTAT, SARRAY(6))
 1125 LET NBTC=82
 1127 CALL ARCOPY (NBTC, CPOSIT, SARRAY(20))
 1129 LET NBTC=16
 1133 CALL ARCOPY (NBTC, AZM, SARRAY(102))
 1135 CALL ARCOPY (NBTC, ELV, SARRAY(118))
 1137 CALL ARCOPY (NBTC, HORZ,SARRAY(134))
 1140 CALL ARCOPY (NBTC, VERT,SARRAY(150))
 1143 LET NBTC=6
 1145 CALL ARCOPY (NBTC, BITLIT(34), SARRAY(166))!                    RUN,STOP,AUTO,MANUAL,NORMAL,TEST LIGHTS
 1147 LET NBTC=16
 1150 CALL ARCOPY(NBTC, IP, SARRAY(172))
 1153 CALL ARCOPY(NBTC, IC, SARRAY(188))
 1157 LET NUMS = 204!                                                 NUMBER OF BYTES TO SEND BACK


 1160 LET OASTAT(13) = 0
 1190 RETURN
!
! Subroutine to process TYPE -2 status request
!
 1200 !
 1210 LET NBTC=82
 1215 CALL ARCOPY (NBTC, CPOSIT, SARRAY(6))
 1230 LET NUMS = 88!                                                  NUMBER OF BYTES TO SEND BACK
 1290 RETURN
```

```
!
! Subroutine to process TYPE -3 status request:
!
 1300 !
 1310 LET NBTC=74
 1320 CALL ARCOPY (NBTC,MOTOR,SARRAY(6))
 1330 LET NUMS = 80!                                    NUMBER OF BYTES TO SEND BACK
 1390 RETURN
!
! Subroutine to process TYPE -4 status request:
!
 1400 !
 1410 LET NBTC=128
 1420 CALL ARCOPY (NBTC,INTOFF,SARRAY(6))
 1430 LET NUMS = 134!                                   NUMBER OF BYTES TO SEND BACK
 1490 RETURN
!
! Subroutine to process TYPE -5 status request:
!
 1500 !
 1510 LET NBTC=51
 1520 CALL ARCOPY (NBTC,BITLIT,SARRAY(6))
 1530 LET NUMS = 57!                                    NUMBER OF BYTES TO SEND BACK
 1590 RETURN
!
! Subroutine to process TYPE -6 status request:
!
 1600 !
 1610 LET NBTC=16
 1615 CALL ARCOPY(NBTC,AZM,SARRAY(6))
 1620 CALL ARCOPY(NBTC,ELV,SARRAY(22))
 1625 CALL ARCOPY(NBTC,HORZ,SARRAY(38))
 1630 CALL ARCOPY(NBTC,VERT,SARRAY(54))
 1650 LET NUMS = 70!                                    NUMBER OF BYTES TO SEND BACK
 1690 RETURN
!
!
```

```
!
!----------------PROCESS A COMMAND REQUEST----------------------
!
!
! Check for valid command type
!
 2000 IF TYPE > MAXCTY  LET NUMS=5, SARRAY(4)=077 : GOTO 2090
!
! Make sure previous command has been processed.  If not, suspend myself
! until it has been.
!
  2030 IF CMDFLC = 0 GOTO 2060
  2040 SUSPEND UNLESS CMD
  2050 GOTO 2030

!
! Call subroutine to unpack command for each particular type
!
 2060 GOSUB (2010 + TYPE * 100)
!
! Set command flag for control loop so he will get it when he can
!
 2070 LET CMDFLG=TYPE
 2090 LET NUMS=5
 2092 LET SARRAY(4)=TYPE
 2099 GOTO 300

 Command request TYPE 1 is initialization -- send back status TYPE -1


! Subroutine to process command request TYPE 2
!
 2200 !
 2210 LET NUMP=40
 2220 CALL ARCOPY (NUMP,RARRAY(5),CMDTBL)
 2222 LET CMDFLG = 1
 2225 IF (DEBUG2) ENTER
 2290 RETURN
!
!
```

```
!-------------------------DECLARATIONS ---------------------------------
!
! All type and array declarations, equates, and initial conditions are
! set up in this subroutine which is call once on initial program
! execution.  This subroutine then deletes itself to save space in the
! LSI-11.
!
! This subroutine assumes that another program has been run and has:
!       a)      Established the common block via a core request, and
!               placed the address of the common block in location 046
!       b)      Initialized the necessary SHIVANET links via a call
!               to ILINK.
!
 9000 IMPLICIT INTEGER A-Z
 9100 ! DEFINITION OF COMMON
 9110 INTEGER C : LET C=@046
 9182 EQU J=@(C)
 9183 INTEGER AZM(7)@(C+2)
 9184 INTEGER ELV(7)@(C+18)
 9185 INTEGER HORZ(7)@(C+34)
 9186 INTEGER VERT(7)@(C+50)
 9187 INTEGER DLY(7)@(C+66)
 9190 INTEGER PTOL(7)@(C+82)
 9192 INTEGER CTOL(7)@(C+98)
 9193 INTEGER PRANGE(7)@(C+114)
 9194 INTEGER CRANGE(7)@(C+130)
 9195 INTEGER MOTOR(36)@(C+146)
 9196 EQU ERROR=@(C+220)
 9199 INTEGER INTOFF(63)@(C+222)
 9200 INTEGER CPOSIT(40)@(C+350)
 9202 INTEGER IP(7)@(C+432)
 9204 INTEGER IC(7)@(C+448)
 9205 BYTE PBARRY(39)@(C+464)
 9215 EQU CMD=@(C+504)
 9220 INTEGER CMDTBL(19)@(C+506)
 9225 EQU CMDFLG=@(C+546)
 9226 BYTE OASTAT(13)@(C+548)
 9227 BYTE BITLIT(49)@(C+562)
 9228 EQU WAITFG=@(C+612)
!
!---------END OF COMMON BLOCK--------------
!
 9320 BYTE TYPE, SARRAY(256), RARRAY(100)
```

```
!
! Initial conditions:
!
 9500 !
 9514 LET UNIT=2  ! UNIT NO. FOR CONTROL ROOM LINK
 9520 LET MAXSTY=10 ! MAX NO. OF STATUS REQUEST TYPES
 9530 LET MAXCTY=6 ! MAX NO. OF COMMAND REQUEST TYPES
 9540 LET CMDFLG=0,DEBUG2=0
!
! Initial message
!

 9600 !
 9610 PRINT "CNTLRM HAS BEEN LOADED"
!
! Initialize the control room link as a slave, and set up to post
! a read with no write
!
 9800 !
 9810 CALL ISLAV (UNIT)
 9820 LET NUMS=0
!
! Delete all declarations, equates, and initial condition statements
! to save room
!
 9900 !
 9910 DELETE 9000-9900
 9920 RETURN
RUN
```

APPENDIX C. Photographs of System Components

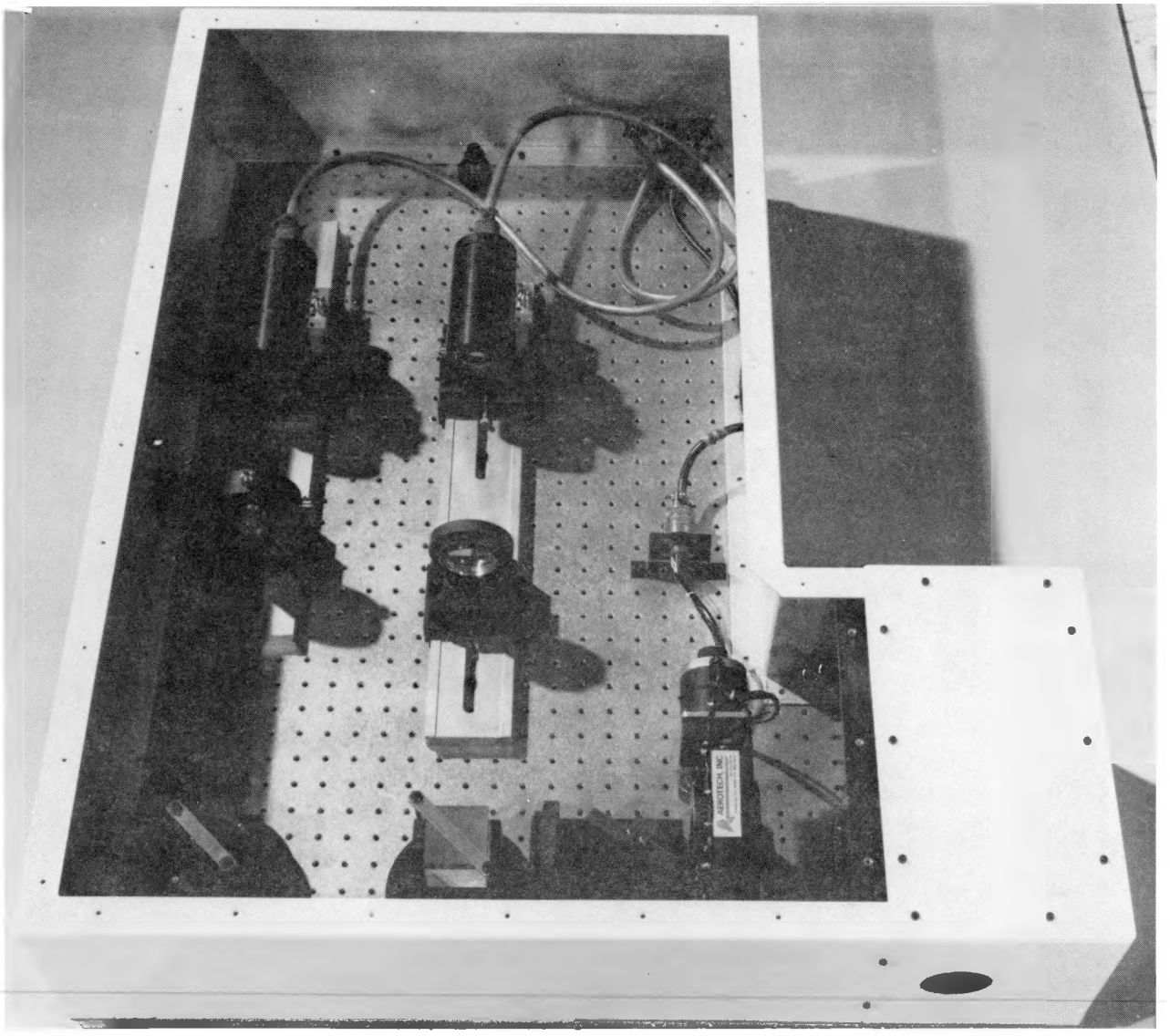


FIG. C-1. OAS sensor.

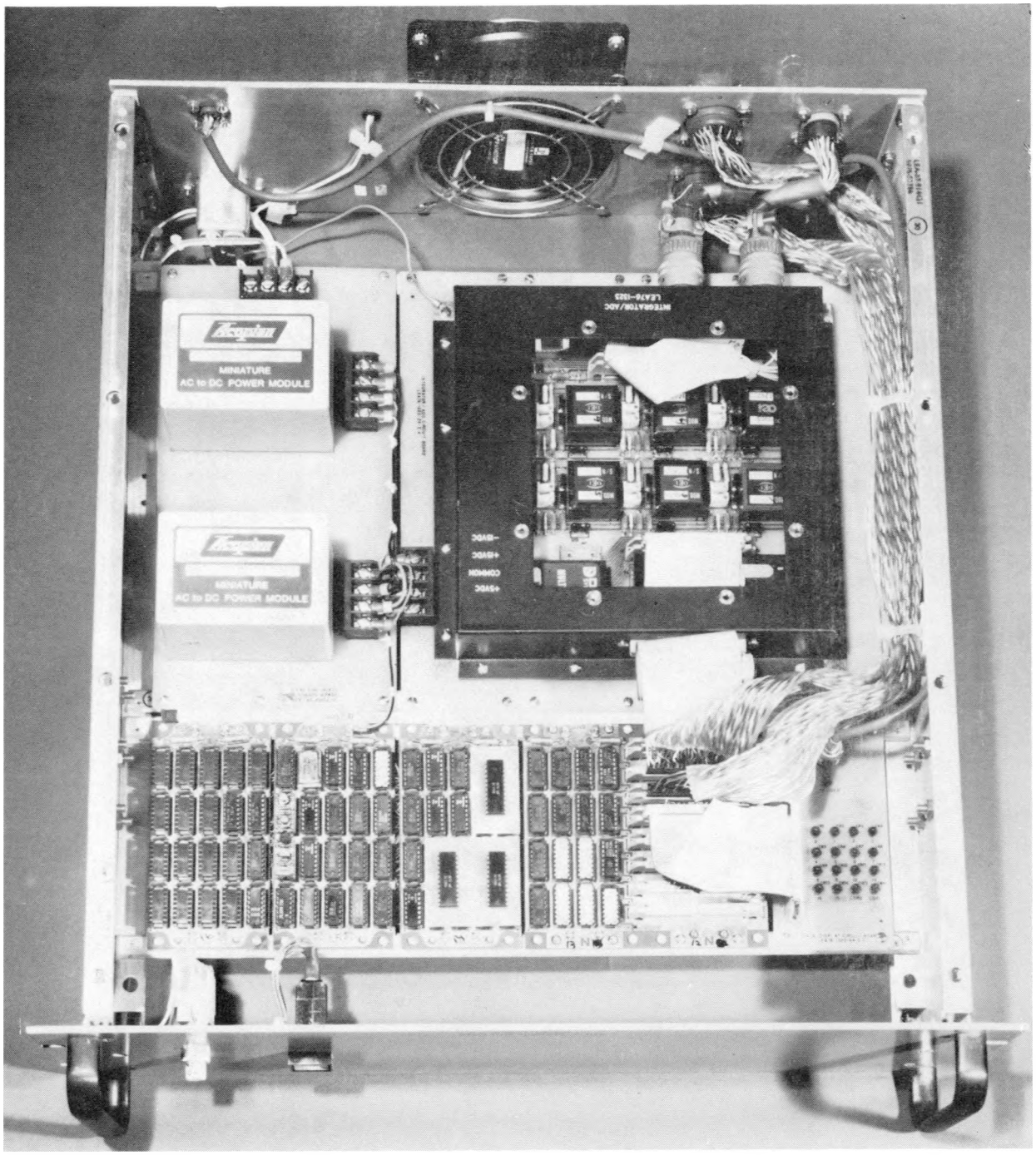


FIG. C-2. Integrator/ADC module.

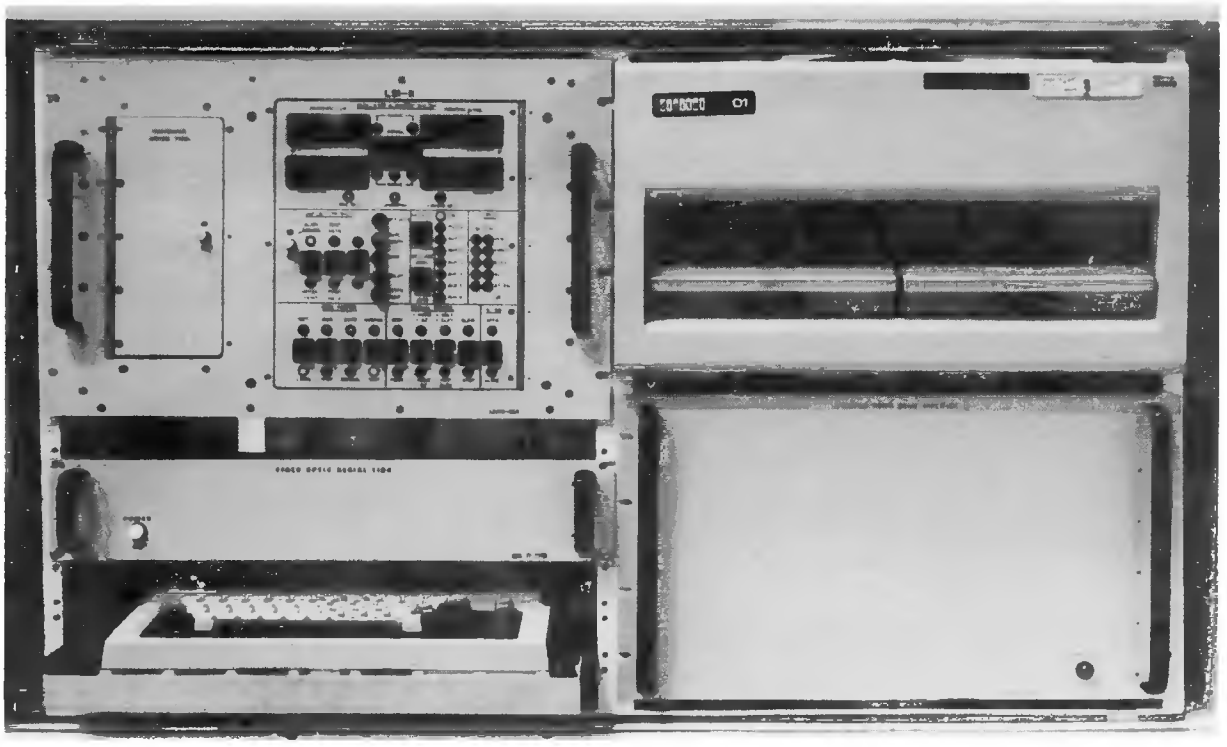


FIG. C-3. OAS control panel and LSI-11.

Page(s) Missing

from

Original Document

INDEX

INDEX (Concluded)