General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

Produced by the NASA Center for Aerospace Information (CASI)

UNIVERSITY OF ILLINOIS
URBANA

AERONOMY REPORT
NO. 112

DESIGN AND IMPLEMENTATION OF A
PREPROCESSING SYSTEM FOR A SODIUM LIDAR

by
D. G. Voelz
C. F. Sechrist, Jr.

September 1, 1983

A E R O N O M Y R E P O R T

N O. 112

DESIGN AND IMPLEMENTATION OF A
PREPROCESSING SYSTEM FOR A SODIUM LIDAR

by

D. G. Voelz
C. F. Sechrist, Jr.

September 1, 1983

ABSTRACT

A preprocessing system was designed and constructed for use with the University of Illinois sodium lidar system. The preprocessing system was developed to increase the altitude resolution and range of the lidar system and also to decrease the processing burden of the main lidar computer. This work describes the preprocessing system hardware and the software required to implement the system. Also presented are some preliminary results of an airborne sodium lidar experiment conducted with the preprocessing system installed in the sodium lidar.

PRECEDING PAGE BLANK NOT FILMED

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

1.1 University of Illinois Sodium Lidar Observations

Late in the 1930's a layer of sodium vapor was discovered in the earth's mesosphere. Observations of the layer since that time have determined that the layer is generally confined to an altitude range of 80 to 100 km with a peak density near 90 km. The upper and lower boundaries of the layer are sharply defined and seasonal and daily variations in the layer have been observed. The layer continues to be studied in order to gain a better understanding of the motions in the layer and the neutral and ion chemistry of the region.

Laser radar (lidar) has been used in the remote sensing of atmospheric constituents since the early 1960's. Initial nighttime lidar observations of the mesospheric sodium layer were conducted late in the 1960's in England [Bowman et al., 1969]. Since then, similar measurements by several sodium lidar groups in various locations have been reported. Initial lidar observations of the layer at the University of Illinois were made in 1977.

During the early experiments at the University of Illinois, emphasis was placed on observing relative structural changes in the layer [Richter and Sechrist, 1978]. In conjunction with this effort, a theory of sodium layer ion chemistry was developed [Richter and Sechrist, 1979]. To aid in the analysis of the data, signal processing techniques were developed and implemented [Rowlett and Gardner, 1979]. As the performance of the lidar system improved, theoretical and experimental studies of the response of the layer to gravity wave perturbations were conducted [Shelton and Gardner, 1981]. Recent interest has been directed toward studies of the horizontal structure of the layer and the implementation of a daytime sodium lidar system.

1.2 General and Sodium Lidar Concepts

Lidar is a remote sensing technique utilizing electromagnetic waves at optical frequencies. The detecting and locating of objects are accomplished by transmitting a laser pulse and analyzing the nature of the returned signal. Lidar systems require targets that exhibit scattering characteristics when subjected to a laser pulse such that a signal returning from the target is detectable above noise.

A basic lidar system consists of a pulsed laser, a receiving telescope, a photodetector such as a photomultiplier tube, and data collection electronics. The laser provides the transmitted pulse. Returning photons backscattered by the target are collected by the telescope. The photodetector converts the collected photons to electrical pulses which are analyzed and recorded by the data collection electronics.

The distance or range to the target is determined by measuring the elapsed time t_r between the transmitted and received pulses. The range R is given by

$$R = 1/2 \ ct_r$$

where $c = 3 \times 10$ m/s, the speed of light. The 1/2 accounts for the two-way propagation of the pulse. Further qualities of the target may be ascertained by observing the intensity and/or the pulse shape of the returning signal.

Sodium lidar utilizes the resonant scattering mechanism of neutral sodium atoms in the mesosphere. A pulse of light at a wavelength corresponding to a sodium transition line and incident on a mesospheric sodium atom causes a fluorescence which results in an enhancement of the returning signal. A tunable dye laser operating at 589.0 nm, the D_2 transition line

of sodium, is commonly used to provide the transmit pulse.

Sodium lidar receiving systems generally operate in a photon-counting
mode and use a range gate collection technique. As the transmit pulse
propagates through the mesosphere, backscattered photons are counted over
short time intervals Δt. The set of counts from the sequential intervals is
a record of the photocounts versus range. The "range gate width" ΔR is the
range resolution of this type of system and is given by

$$\Delta R = 1/2c\Delta t.$$

The interval time Δt is often referred to as the "range gate time." "Range
bin" is a term applied loosely to the storage location of the counts for a
particular interval. The range R_i corresponding to the ith range bin is
given by

$$R_i = 1/2c(t + \Delta t/2)$$

where t is the time elapsed between the transmitting of the laser pulse and
the beginning of the collection of the counts in the ith range bin. An
estimate of the density of a target at a particular range (for example,
sodium density at 90 km) is obtained by examining the number of counts in
the corresponding range bin.

Unfortunately, in most sodium lidar systems the signal-to-noise ratio
for a single laser pulse is quite poor. A common method of increasing the
signal-to-noise ratio is to integrate over many laser pulses. This simply
involves collecting counts for single laser pulses as described above and
then adding the range bin counts to corresponding range bin counts of
previous laser pulses. The completed integration process results in a set
of range bin counts which are collectively termed a "profile." The term
"profile integration sum," in this report, refers to the range bin counts of
an incomplete profile. The integration technique does increase the signal-

to-noise ratio but the time resolution of the system is decreased.

1.3 Objectives of This Investigation

The main objectives of this investigation were to design and implement a preprocessing system for use in the receiving section of the University of Illinois sodium lidar system. The system was designed and developed to increase the altitude resolution and range of the lidar system and also to decrease the processing burden of the main lidar computer by performing the integration of the data collected from successive laser shots.

A secondary objective is to present some preliminary results of an airborne sodium lidar experiment conducted by the University of Illinois Aeronomy Laboratory (Lidar Group) in conjunction with the National Aeronautics and Space Administration (NASA). The initial testing of the preprocessing system with the complete lidar system was conducted during this experiment, and the experimental results establish a high confidence level of operation for the preprocessing system.

2. UNIVERSITY OF ILLINOIS LIDAR SYSTEM

2.1 Introduction

Observations of the mesospheric sodium layer with lidar techniques began in 1977 at the University of Illinois [Richter and Sechrist, 1978]. The lidar system at this time consisted of a flashlamp-pumped dye laser developed at the University, a 0.38 m diameter Fresnel lens receiving telescope, a liquid nitrogen cooled photomultiplier tube (PMT), and a signal pulse discriminator and counter developed in a Physics Department research project. The system was located at the Aeronomy Laboratory Field Station near Urbana, Illinois. A Digital Equipment Corporation (DEC) PDP-15 computer at the Field Station directed the collection and storage of the data.

The next few years produced several changes in the Urbana lidar system. A counter interface was built to replace the Physics Department's pulse counter [Kinter, 1977]. The interface was designed to operate with an HP 2114A microcomputer which allowed the system to be independent of the Field Station PDP-15. In 1979, a 1.22 m diameter Fresnel lens telescope superseded the 0.38 m telescope [Rowlett and Gardner, 1979]. Also in 1979, a Digital Group Z-80 microcomputer replaced the HP 2114A [Teitelbaum and Sechrist, 1979]. Early in 1981, a flashlamp-pumped dye laser, built by the Candela Corporation, was purchased for the lidar system. This laser had better reliability and provided a higher pulse repetition rate than the previous laser. Two other items were also acquired early in 1981: an electrically cooled PMT housing which replaced the liquid nitrogen cooling system; and a pulse discriminator built by Princeton Applied Research, which replaced the Physics discriminator. Later in 1981, the lidar computer system

was again upgraded with a DEC LSI-11/2 microcomputer replacing the Z-80 [Vogel, 1982].

As the reliability and performance of the system increased, sodium lidar observations became feasible at sites other than Urbana. This allowed more diverse sodium lidar observations to be made as the lidar system was integrated with specialized pieces of equipment and facilities at other sites. In June 1981, October 1981, and May 1982 sodium lidar campaigns were conducted at the National Aeronautics and Space Administration (NASA) Goddard Space Flight Center (GSFC) near Greenbelt, MD. A steerable 48-inch astronomical telescope at GSFC allowed scanning-type (off-zenith) observations of the sodium layer to be made. In March 1983, an airborne sodium lidar experiment was conducted at the NASA Wallops Flight Center, Wallops Island, VA. Chapter 5 contains some preliminary results of this experiment.

The most recent addition to the lidar system is the preprocessing system described in the following chapters of this report. The preprocessing system replaces the old counter interface while also reducing the data processing burden of the LSI-11 main computer. The following sections of this chapter describe the lidar system hardware at the present time. A block diagram of the lidar system with the preprocessing system installed is shown in Figure 3.2 of Chapter 3.

2.2 Transmitting System

The principal component of the transmitting system is a Candela LFDL-1 flashlamp-pumped tunable dye laser utilizing Rhodamine 6G Perchlorate in a methyl alcohol and water solution. The laser consists of three subsystems: (1) a laser head which contains the optical cavity, dye cell, and flashlamp; (2) a refrigeration/circulation (chiller) unit used for both cooling and dye flow systems; and (3) a control unit/high voltage power supply. The laser

parameters are given in Table 2.1.

Laser tuning is achieved through a grating and an etalon in the laser cavity. Coarse tuning adjustments are made manually with the grating while fine adjustments are made with the etalon which is positioned by a Burleigh motor-micrometer. The wavelength of the laser beam is monitored by diverting a small fraction of the beam to a monochromator and to a hollow-cathode sodium discharge tube. The monochromator, in combination with a Reticon diode array line scanner (part numbers: RC 301, RL 256G), is used to display the intensity of the laser beam portion as a function of wavelength on an oscilloscope. The output of a sodium lamp is also displayed as a reference. Coarse tuning adjustments are monitored with this arrangement. Fine tuning adjustments are monitored by observing the laser-induced changes in the voltage between the electrodes of the gas discharge plasma in the hollow-cathode sodium tube (opto-galvanic effect). The electrode voltage is displayed on an oscilloscope.

Completing the transmitting system are the final laser beam output optics which generally include a beam expander to reduce beam divergence, and dielectric-coated mirrors to steer the laser beam.

2.3 Receiving System

The major components of the receiving system are the telescope, the photomultiplier tube (PMT) and its associated electronics, and the pre-processing and main computer systems. A 1.22 meter diameter f/1.56 acrylic Fresnel lens is the collecting element for the telescope utilized at the Urbana site. A detailed discussion of the telescope is provided by Rowlett and Gardner [1979]. A 48-inch Cassegrain astronomical telescope with Coude focus was employed for observations conducted at NASA Goddard Space Flight Center. The telescope is part of a versatile facility that also includes a

TABLE 2.1 CANDELA LFDL-1 LASER PARAMETERS.

Wavelength	589.0 nm
Pulsewidth (FWHM)	2 μsec
Linewidth	1 pm
Energy per pulse	50 mj
Pulse repetition rate	10 Hz
Beam divergence	1 mrad
Flashlamp lifetime	3×10^6 shots

computer-controlled pointing system capable of tracking stars and satellites. The airborne lidar system at NASA Wallops Flight Center utilized a 16-inch diameter primary, Newtonian configured telescope.

The telescope optics required for the 1.22 m Urbana telescope are shown in Figure 2.1. The collected photons are focussed by the Fresnel lens onto the plane of a field-stop iris. Beyond the iris, the beam is collimated by a Nikon 35 mm f/1.4 lens prior to its passage through an interference filter. The interference filter is used to reduce the number of photons due to background sky illumination. Two temperature controlled filters with bandwidths of 5 and 0.5 nm are currently used with the system. The filter selected for a particular experiment depends on the experimental objectives, received signal intensity at the time of the experiment, etc. A lens in the PMT housing focusses the photons passed by the filter onto the cathode of an RCA C31034A PMT. To reduce dark counts, the PMT is thermoelectrically cooled by a Products for Research Model TE-206TS-RF cooled housing.

Due to the relatively long focal lengths of the 48-inch Goddard telescope (1300 in, f/27) and the 16-inch Wallops telescope (approximately 400 in, f/25), the receiving optics arrangement following these telescopes is simplified from that shown in Figure 2.1. The long focal lengths provide a nearly collimated beam of collected light at the focus of the telescope. Therefore, the field-stop iris and the Nikon collimating lens are not required in front of the interference filter and the PMT focusing lens is positioned at the focal point of the telescope.

The output of the PMT is applied to a Princeton Applied Research Model 1121 Amplifier-Discriminator. The discriminator compares the pulses generated by the PMT to a preset threshold level. NIM standard (18 mA (- 1 V), 12 nsec) and emitter-coupled logic (ECL) level pulses are produced

FRESNEL LENS
122 cm DIA
FOCAL LENGTH = 190.5 cm

COLLIMATING
LENS
(NIKON 35 mm f/1.4)

FOCUSING LENS
f = 3.5 cm

COOLED
HOUSING

45°
MIRROR

FIELD-STOP
IRIS

INTERFERENCE
FILTER

PHOTOMULTIPLIER
(RCA C31034A)
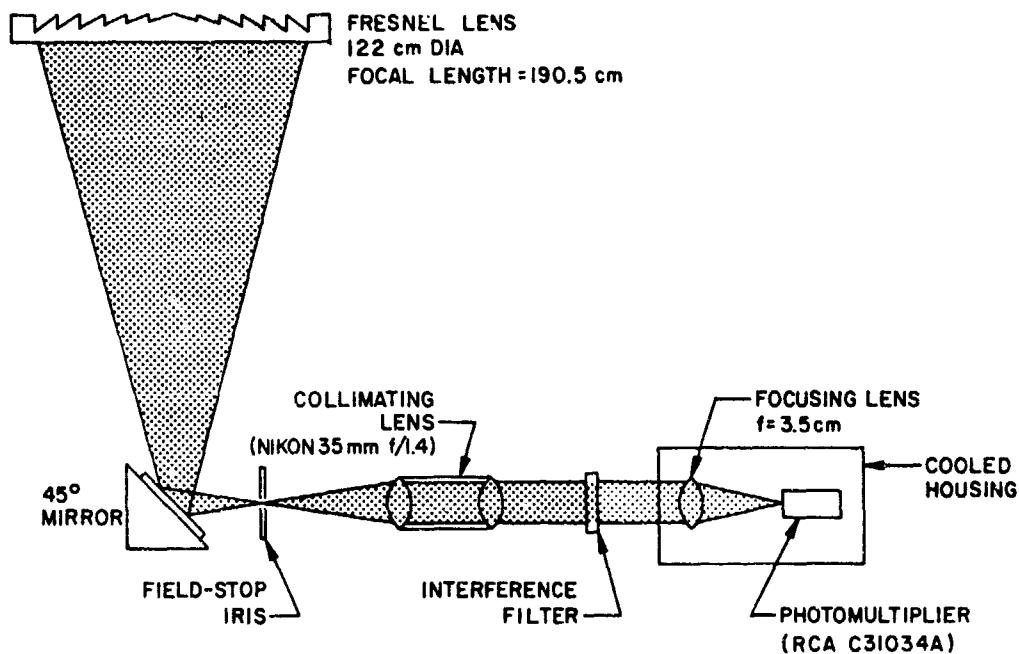
Figure 2.1   Schematic diagram of 1.22 m telescope optics [Rowlett and Gardner, 1979].

by the discriminator after receiving PMT pulses that exceed the threshold. The discriminator pulses (currently from the NIM standard output) are counted by the preprocessing system.

At very high counting rates (above 20 MHz) the PMT is subject to overloading conditions which cause significant errors in the recorded number of detected photons. Strong Rayleigh scattering of the laser beam below 30 km altitude consistently provides very high counting rates that disturb counts collected from altitudes below 30 km as well as those collected from altitudes above 30 km as the PMT is recovering. A PMT blanking controller and a timing controller were constructed to alleviate PMT overloading due to Rayleigh scattering [Shelton and Gardner, 1981].

The blanking controller achieves the blanking or gating of the PMT by reducing the voltage of the first dynode of the PMT with respect to the PMT cathode potential, effectively decreasing the overall gain of the PMT. Measurements on this system indicate the gain is reduced by approximately 3 orders of magnitude. The PMT recovery time after blanking has been observed to be on the order of 6 to 10 µsec. Therefore, the blanking controller can be engaged to blank the PMT as scattered signal below 30 km altitude is collected, and disengaged for data collection above 30 km altitude.

The timing controller synchronizes the firing of the laser and the PMT gating. The triggering of the timing controller is initiated by a pulse received from the preprocessing system. After triggering, the timing controller engages the blanking controller and delays 5 µsec before sending a trigger pulse to the laser. The delay ensures the blanking of the PMT is complete as the laser fires. Approximately 200 µsec after the laser triggering (corresponding to an altitude of 30 km), the blanking controller is disengaged and data collection continues. A variable resistor in the timing

controller sets the blanking duration.

The preprocessing and LSI-11 main computer systems are discussed in the following chapters. In general, the preprocessing system initiates the laser firing, counts the discriminated pulses from the amplifier-discriminator, and integrates the counts from consecutive laser shots. The main computer directs the overall experiment and permanently stores the collected data on a floppy disk.

3. THE PREPROCESSING SYSTEM

3.1 Introduction

The sodium lidar preprocessing system is designed to direct the collection of data for one concentration vs. altitude profile of the mesospheric sodium layer. The system controls laser firing, counts the returning discriminated photon pulses, sections the counts into appropriate range bins, and integrates similar range bins over consecutive laser shots to form a profile. Completed profiles are sent to a main computer for further processing and storage. As mentioned previously, the system was devised to decrease the data processing burden of the main computer.

The system consists of two major pieces of hardware: an Apple II Plus microcomputer and the Sodium Lidar Preprocessing (SLIPP) interface unit. Other hardware items include a Direct Memory Access (DMA) circuit for the Apple-SLIPP unit communications and a RS-232 standard serial communications link between the Apple and the main computer. Software routines drive the Apple and the main computer.

The Apple computer masterminds the profile collection. All communications with both the SLIPP unit and the main computer, laser firing, and integrations are handled through the Apple's software. The range bin counts and the profile integration sum are stored in the Apple's Random Access Memory (RAM). The Apple also displays messages on its video screen to notify the lidar operator of the present status of the system.

The SLIPP unit interfaces the Apple to the transmitting and receiving sections of the lidar system. Line driver circuitry in the SLIPP unit allows the laser to be triggered by the Apple. After a laser shot the received photon pulses are counted in the SLIPP unit. The SLIPP unit has no

memory of its own so each range bin collected is immediately transferred to the Apple RAM through the DMA circuitry.

Although not strictly considered part of the preprocessing system, the lidar main computer and its software are closely associated with the preprocessing system. The main computer for the University of Illinois lidar system is a DEC LSI 11/2 computer. The LSI-11 directs the overall experiment. It is programmed with the information on time to collect a data profile and the number to collect. Through the LSI-11 terminal the lidar operator is able to access and change these experiment parameters. An RS-232 standard serial communications link allows the LSI-11 to send commands to the Apple and the Apple to send data profiles back to the LSI-11. The LSI-11 also gives the operator a summary of each of the completed profiles and stores the profiles on a floppy disk.

3.2 Data Acquisition Timing

The desired pulse repetition rate of the laser and the desired rate of collecting data profiles were two parameters of particular interest during the designing of the preprocessing system. The system had to be able to integrate data from consecutive laser shots witn enough speed to let the laser operate at its maximum pulse rate (10 Hz), and the data transfer from the preprocessing system to the main computer had to be reasonably fast so as not to cause any long delays between data profiles. Figure 3.1 is a timing diagram of the data collection process timing for the completed preprocessing system.

Part (a) of Figure 3.1 shows the collection process for a single laser pulse. The laser is initially triggered by the preprocessing system. Two thousand range bins of data are then immediately collected. This requires about 2000 μsecs as the range gate time is 1 μsec for each bin. Another
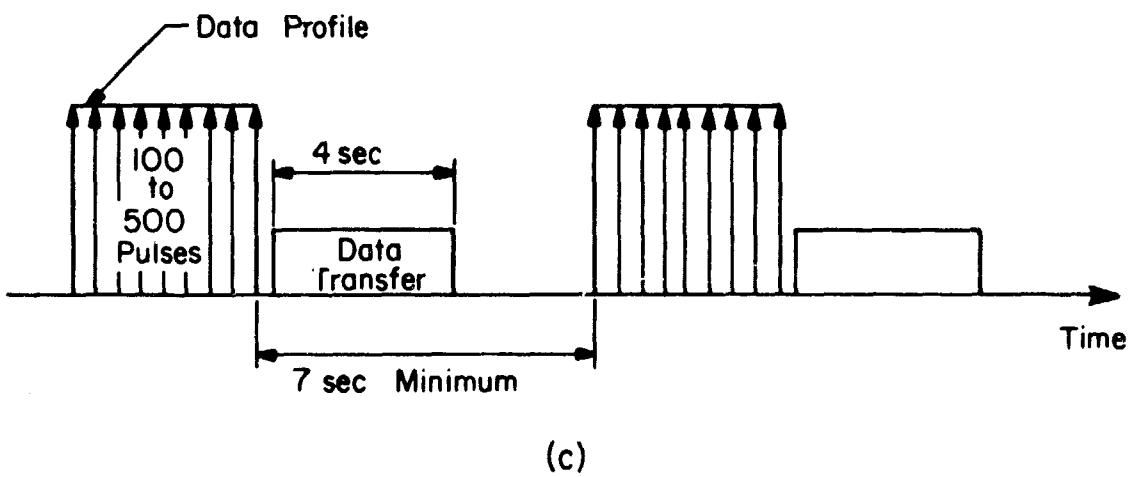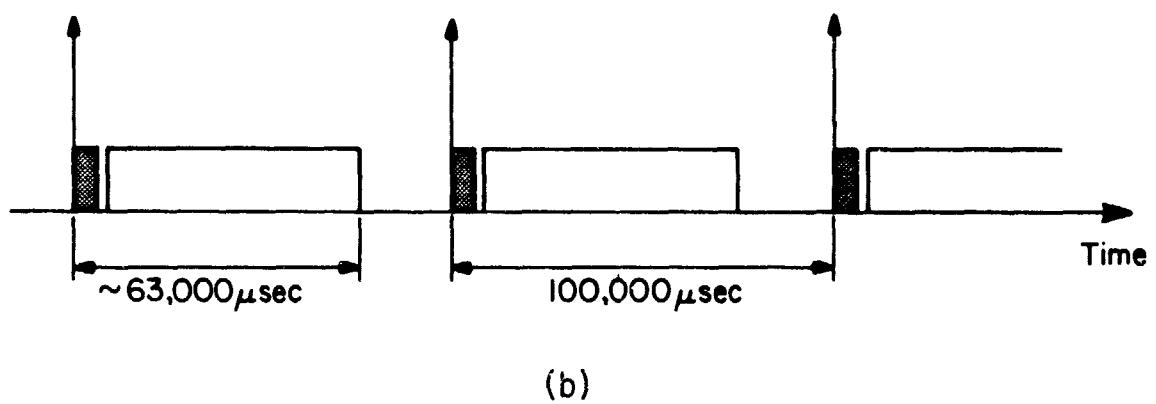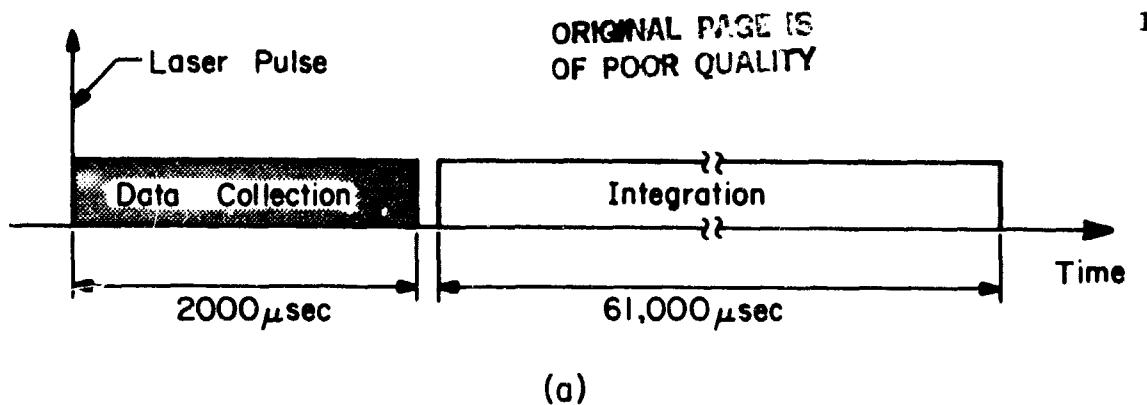
Figure 3.1 Data acquisition timing for: (a) a single laser pulse,
(b) repeating laser pulses, and (c) full profiles.

61,000 µsecs are required by the Apple to integrate the data from the current laser shot into the profile integration sum. Therefore, the data preprocessing requires about 63,000 µsec total per laser pulse.

Repeating laser pulses at 10 Hz are shown in part (b) of Figure 3.1. The data collection and integration times are denoted after each laser pulse. It is evident that the preprocessing system is able to operate with laser pulse rates exceeding 10 Hz.

Part (c) of Figure 3.1 shows the collection of completed profiles. A data profile is stored in Apple RAM as 2000 16-bit words. Therefore, roughly 4 seconds are required for the Apple to send the completed profile over the serial link to the LSI-II at 9600 baud. The LSI-11 uses another 3 seconds to store the profile on a floppy disk and give the operator a summary of the profile. The total delay of 7 seconds between the end of one profile and the beginning of the next is tolerable. More details on system timing are included in Section 3.4 on system parameters.

3.3 System Operation Description

This section describes in general terms the interactions of the preprocessing system with the rest of the lidar system during the collection of a data profile. Figure 3.2 is a block diagram of the lidar system with the preprocessing system installed. Some of the block items in the figure are explained in more detail in Chapter 2 on the lidar system hardware. The figure is a useful reference for the following discussion on the system operation.

A lidar data run begins with the operator inputting or changing experiment parameters on the LSI-11 terminal. These parameters include the number of profiles per set, the profile range bins to be saved, laser rep rate, etc. (see Section 5.2.2). Once the operator has initiated a data run,
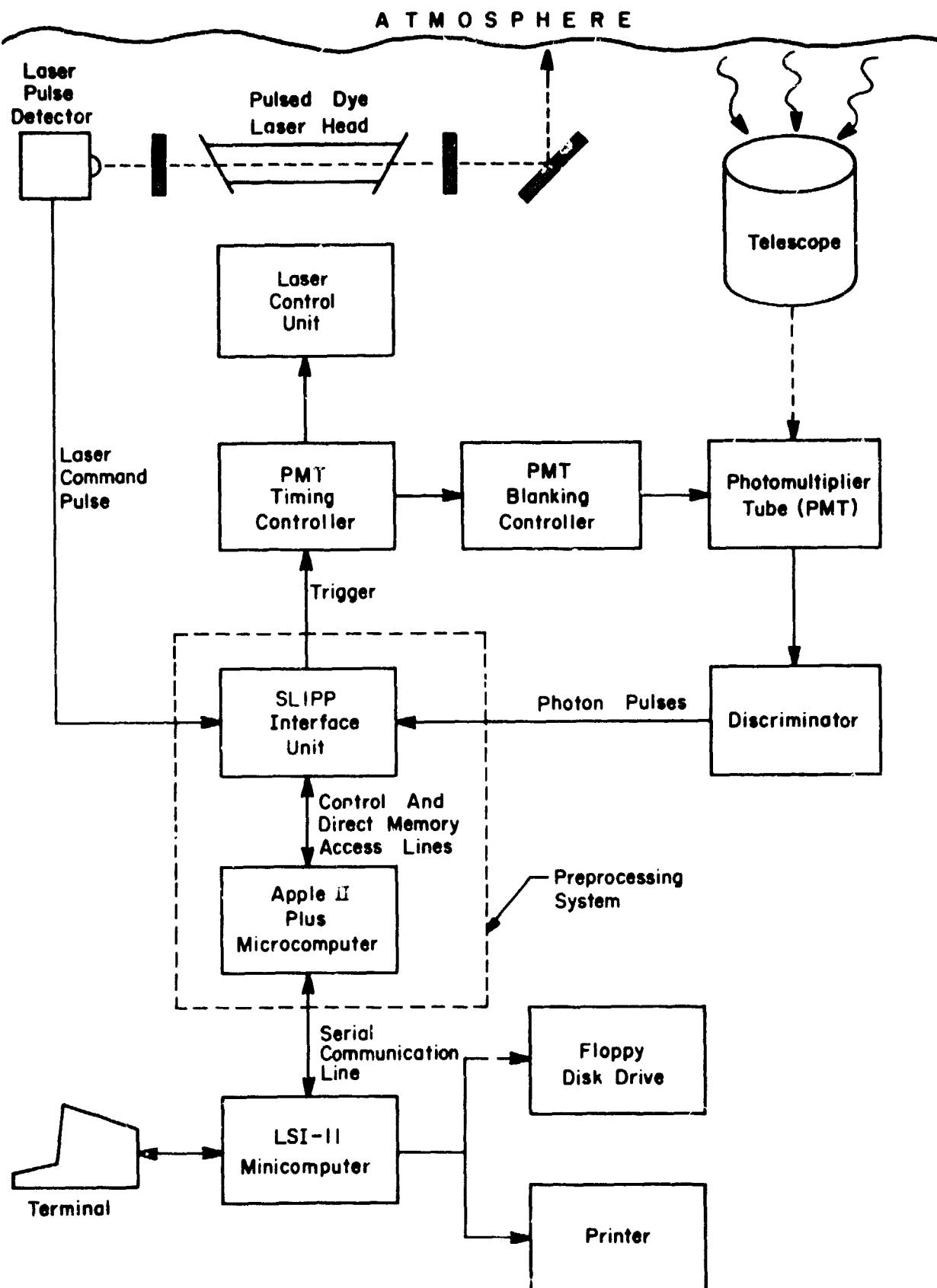
Figure 3.2 Lidar system with preprocessing system installed.

the LSI-II sends two communication frames to the Apple over the serial link. The first frame contains information on the desired laser pulse rep rate and the desired number of laser shots per profile. The second frame signals the Apple to begin collecting a profile.

The first action of the Apple upon the receipt of the "data run" frame is to initialize the DMA circuitry for data transfer from the SLIPP unit to the Apple RAM. Next, the Apple clears the block of RAM that holds the profile integration sum. Finally, the Apple sends the laser trigger pulse to the SLIPP unit. The laser pulse not only passes through the SLIPP unit to trigger the laser and the PMT timing circuitry but also enables the SLIPP unit for counting.

Once enabled, the SLIPP unit waits for the Laser Command Pulse (LCP). This is a positive logical pulse that is generated by a laser pulse detector when the laser fires. The LCP notifies the SLIPP unit that the laser has fired and the unit begins counting photon pulses in 1 μsec intervals. A this point the DMA circuitry takes control of the Apple buses and each interval, or range bin, collected is immediately stored in the Apple RAM through DMA. The SLIPP unit actually uses two toggling counters in order that during any 1 μsec interval, one counter is counting photon pulses while the contents of the other counter are being placed in memory. When all of the range bins are collected, normally 2000, the DMA circuitry disables the SLIPP unit and the Apple regains control of its address and data buses.

Next, the Apple checks a status bit in the DMA circuit to be sure the DMA occurred. If the bit is not set a problem occurred, such as the laser never fired, and the Apple jumps to an error routine. With the bit set, the Apple adds the new range bin counts into the profile integration sum. The collection process is now complete for one laser shot. The Apple checks to

see if the profile requires more laser shots. If so, the Apple sends the next laser trigger pulse, enables the SLIPP unit, and the collection begins again. If the profile is complete it is sent over the serial link to the LSI-11.

After receiving the profile the LSI-11 does some simple calculations and gives the operator a summary of the profile (see Section 5.2.2 for more details). The summary is given both on the terminal and a hard copy printer. The data are stored on a floppy disk and the collection of one profile is complete. If more profiles are to be collected, the LSI-11 sends the "data run" frame to the Apple again. If the set of profiles is finished the LSI-11 alerts the operator and waits for the next command.

3.4 System Parameters

This section contains descriptions and values of the important parameters of the preprocessing system. Definitions of some of the terms used in this section can be found in Section 1.2. A list of the parameter values is given in Table 3.1.

Range Gate:

The range gate time is fixed by the logic in the SLIPP interface unit at 1 μsec. This corresponds to a range gate width of 150 meters for each range bin. This is the maximum resolution of the system.

Range Bins:

The preprocessing system currently collects 2000 range bins of data for each laser shot. With the 1 μsec range gate time mentioned above, the total system range is 300 km. The number of range bins used for data collection may be changed with a few Apple software alterations.

Photon Pulse Count Bandwidth:

The counters and associated logic in the SLIPP interface unit are

TABLE 3.1 PREPROCESSING SYSTEM PARAMETERS.

Range gate time	1 μsec
Number of range bins	2000
Photon pulse input (NIM)	−18 mA($\simeq$ −1 V), 12 nsec
Photon pulse count bandwidth	50 + MHz
Inter-profile period	7 sec

capable of detecting NIM standard (-18 mA (-1 V), 12 nsec) signal pulses
with rates exceeding 50 MHz. However, the discriminator used by the
University of Illinois sodium lidar (P.A.R.C. Model 1121) which provides the
pulses for the counters is rated at only 37 MHz for random pulses. It is
also apparent that the PMT in the sodium lidar system (RCA C31034A)
generally overloads at count rates approaching 20 MHz. The SLIPP unit was
also designed for possible future expansion to a balanced, emitter-coupled
logic (ECL) compatible, 5 nsec pulse count input. This would reduce noise
created in the cable between the discriminator and the SLIPP unit. The
balanced ECL and standard NIM outputs are both provided by the P.A.R.C. 1121
discriminator.

Data Acquisition Rate:

The time necessary to obtain one sodium data profile is, of course,
dependent on the number of laser shots required and the laser repetition
rate. A little more than 4 seconds are necessary for a profile to be
transferred from the Apple to the LSI-11. Three seconds are required by the
LSI-11 to display a summary of the profile on the terminal and printer, and
store the profile on a floppy disk. Therefore, with a laser operating at
10 Hz, a profile requiring 100 shots can be collected in about 17 seconds.

Data Storage:

Apple Memory - The Apple Random Access Memory is used to temporarily
store the incoming range bin counts and the profile integration sum. The
range bin counts are recorded in 2000 8-bit bytes in an area of memory
designated as "buffer memory." Two thousand integration sum bins a.e each
16 bits in length and stored in an area designated as "mainframe memory."
For programming ease 2048 (2K) bytes are allocated as buffer memory and 4096
(4K) bytes are allocated as mainframe memory. The high order bytes and the

low order bytes of the 16-bit words in the mainframe memory are divided into two sections. This is a result of the addition routine used by the Apple to form the profile integration sum.

LSI-Memory - A profile is sent from the Apple to the LSI-11 with the low order bytes of the 16-bit words first and the high order bytes second. The bytes are assembled upon receipt by the LSI-11 into 16-bit words and stored in the array LDATA.

LSI-11 Floppy Disk - The LSI-11 computer permanently stores sets of sodium data profiles on a floppy disk with a DEC floppy disk drive (both RX01 and RX02 type drives have been used). The profile sets are stored as unformatted direct access files. The name assigned to each file is SETxxx.DAT, where xxx is the number of the set. For example, SET001.DAT is the file name for the first set collected in a data run.

4. CIRCUITS DESCRIPTION

4.1 Apple Direct Memory Access Circuit

4.1.1 <u>Introduction</u>. Direct Memory Access is a fast and efficient method of transferring data to or from a computer's memory. This method involves a specialized device communicating directly with the computer's memory. The data transfer can therefore be performed with minimum interaction from the microprocessor (MPU). The sodium lidar preprocessor design requires the Apple's Random Access Memory to store incoming photon counts from the SLIPP interface unit. The counts are recorded in bytes every microsecond which corresponds to a data transfer rate of 1 Mbyte/sec. This rate is much too fast for the Apple to handle in a software polling loop or an interrupt routine. Therefore, DMA is a necessity. The DMA controller selected for this project was the Motorola MC6844 Direct Memory Access Controller (DMAC) chip. The chip was designed for use with the Motorola MC6800 series microprocessors. However, strong similarities between the MOS 6502 microprocessor in the Apple and Motorola's MC6800 allow the MC6844 DMAC to work ell with the Apple's circuitry.

4.1.2 <u>Apple timing</u>. All clock signals, memory strobe signals, and video signals in the Apple II Plus are derived from a single 14.138 MHz oscillator on the main Apple circuit board. The various signals are obtained by using a number of counters, shift registers and multiplexers. Figure 4.1 is a typical timing diagram for some of the Apple signals. 7M is an intermediate timing signal, Φ0 and Φ1 are system clocks, and Q3 is an asymmetrical general-purpose timing signal. The Φ0 Apple clock is comparable to the Φ2 clock of other microprocessor systems.
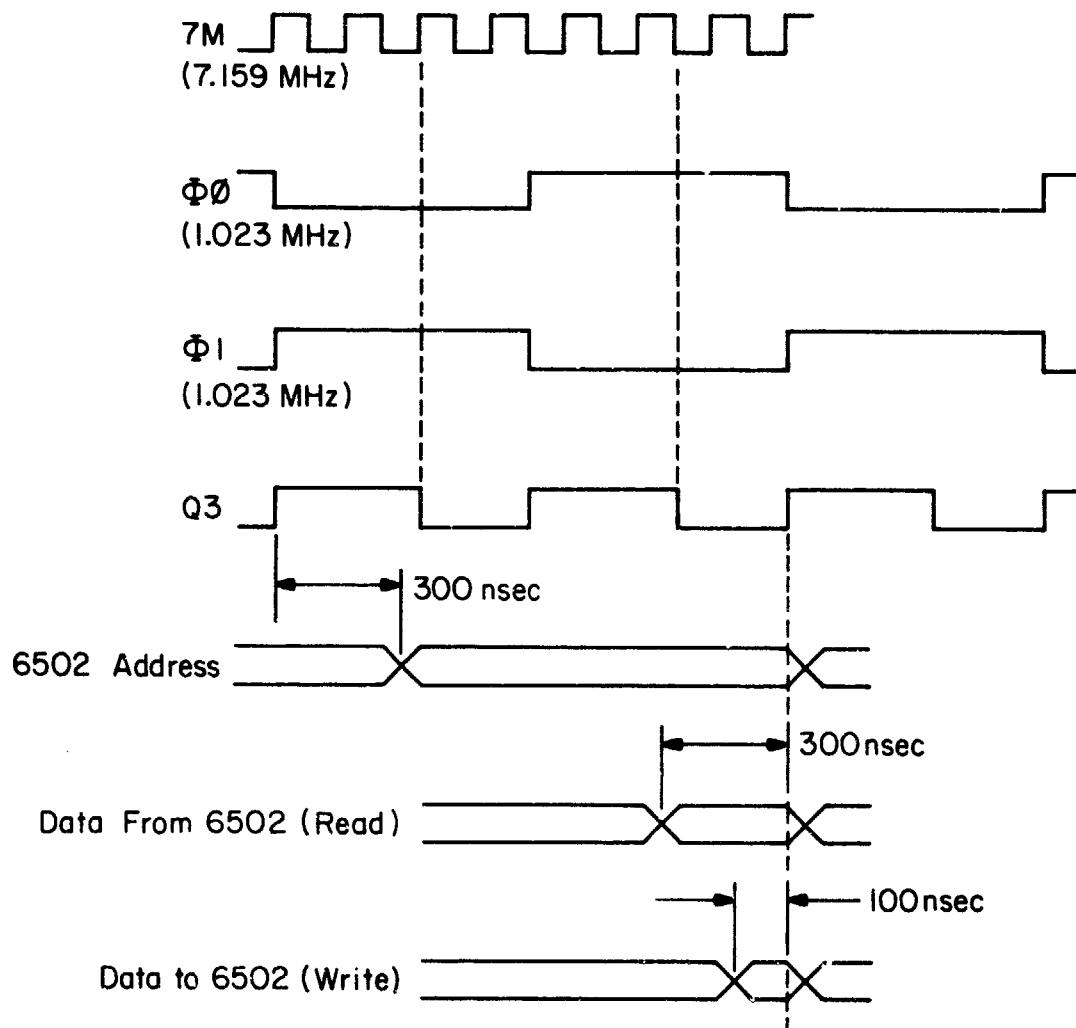
Figure 4.1 Apple timing signals.

During normal operation the Apple's RAM is used by both the 6502 and the Apple's video generating circuitry. The 6502 accesses the RAM only during the high (+5 volts) half of the $\Phi 0$ clock while the video generating circuitry updates the video screen from the RAM during the low (ground) half of the $\Phi 0$ clock. The action of the video circuitry also refreshes all of the Apple's RAM. Because of this feature, the easiest implementation of the DMA controller is to have it essentially replace the 6502 during DMA data transfer periods. This implementation allows the video circuitry to continue to function normally during a DMA period. Another result of this type of DMA architecture is that the data transfer rate is dictated by the microprocessor system clock rate, $\Phi 0$ for the Apple, which would allow a maximum DMA transfer rate of 1.023 Mbyte/sec. The MC6844 DMAC was chosen because it is designed for the type of implementation discussed above and it is capable of operating at the Apple system clock rate.

One peculiarity of the Apple timing system is that every 65th cycle of the $\Phi 0$ and $\Phi 1$ clocks is lengthened by about 140 nanoseconds. This corresponds to a frequency of 0.895 MHz for the long cycle. The lengthening is an artifact of the logic used to obtain the clock signals from the 14.318 MHz oscillator. Because of this anomaly, the SLIPP interface unit was designed with its own oscillator and clock logic rather than using the Apple's. In order to place photon pulse counts in correct range bins the SLIPP unit must be able to determine accurately the time elapsed since the laser firing. This would not be possible with the Apple's inconsistent clock. This design results in the Apple and the SLIPP unit operating asynchronously. To account for the timing differences, an asynchronous First-In/First-Out buffer (FIFO) is used to interface the SLIPP unit and the Apple DMA circuit. For more details on the interfacing see Section 4.2.4.

4.1.3 <u>Apple peripheral input/output</u>. Along the rear edge of the
Apple's main board are eight peripheral connectors. The connectors or
"slots" are labeled 0 through 7 beginning from the left-hand side of the
board. Slots 1 through 7 are used for most peripheral applications while
slot 0 was designed specifically for memory or interface expansion. The
pinout for the slots is shown in Appendix I. The preprocessor DMA card was
designed to be used in any of the slots 1 through 7.

Each slot has specific memory locations assigned to it. This memory-
mapped structure allows the user (or user's program) to access cards in any
of the slots. Each slot is given 16 locations for general input and output
purposes. The locations for the particular slots are listed in Table 4.1.
Whenever the MPU calls an address within the 16-byte allocation of a par-
ticular slot, the $\overline{\text{Device}}$ $\overline{\text{Select}}$ line (pin 41) on that peripheral connector
will become active (drop to ground). By listening to the $\overline{\text{Device}}$ $\overline{\text{Select}}$
line, a peripheral card can determine when a byte in the general I/O space
reserved for it is being addressed.

In addition to the 16 general I/O locations, each slot is assigned 256
locations (one page) of Read Only Memory (ROM) or Programmable ROM (PROM)
space. These locations are listed in Table 4.2. Although allocated as
peripheral card program space, these locations function in a similar manner
as the general I/O locations. Whenever an address within the one page
allocation of a particular slot is called, the $\overline{\text{I/O}}$ $\overline{\text{Select}}$ line (pin 1)
on that peripheral connector will become active (drop to ground). A
peripheral card can determine when its program space is being addressed by
listening to the $\overline{\text{I/O}}$ $\overline{\text{Select}}$ line. The Apple accesses the preprocessor DMA
card through both the general I/O and the program locations. The general
I/O addresses are decoded on the DMA card to provide control signals for

TABLE 4.1 APPLE PERIPHERAL CARD GENERAL PURPOSE I/O
LOCATIONS.

SLOT	LOCATIONS
0	$C080-$C08F
1	$C090-$C09F
2	$C0A0-$C0AF
3	$C0B0-$C0BF
4	$C0C0-$C0CF
5	$C0D0-$C0DF
6	$C0E0-$C0EF
7	$C0F0-$C0FF

TABLE 4.2 APPLE PERIPHERAL CARD PROM LOCATIONS.

SLOT	LOCATIONS
1	$C100-$C1FF
2	$C200-$C2FF
3	$C300-$C3FF
4	$C400-$C4FF
5	$C500-$C5FF
6	$C600-$C6FF
7	$C700-$C7FF

firing the laser, resetting the SLIPP unit, etc. The program locations are used to access the MC6844 DMAC programmable Control Registers.

A DMA transfer with the Apple is accomplished by pulling the $\overline{DMA}$ line (pin 22) on the connector LOW. This disables the 6502's address bus and interrupts the clock to the 6502, effectively halting the microprocessor. A peripheral controller, such as the MC6844 DMAC, is able to supply the Apple RAM addresses while the $\overline{DMA}$ pin is held LOW.

Other line present on the peripheral connector that are used by the DMA card include the clock signals Φ1 and Q3 (pins 38 and 37), the Read/Write line R/$\overline{W}$ (pin 18), and the Reset line $\overline{RES}$ (pin 31).

The Apple peripheral connectors also provide pins to construct a daisy chain priority system for cards that issue interrupt or DMA requests. These pins are labeled INT IN, INT OPUT, DMA IN, and DMA OUT (pins 28, 23, 27 and 24, respectively). The system was designed for the highest priority device to be installed in the left most slot of the Apple slots 1 through 7. It was apparent that in the preprocessing system the DMA card would not have any conflicts with other devices issuing DMA requests. Therefore, rather than constructing some arbitration logic testing for DMA request conflicts, the DMA IN and DMA OUT pins on the card were simple connected together to preserve the DMA daisy chain. The preserve the interrupt daisy chain, the INT IN and INT OUT pins on the card were connected together.

4.1.4 _The MC6844 DMAC_. The MC6844 Direct Memory Access Controller is a TTL compatible chip that directs the DMA data transfer from the SLIPP unit to the Apple. It controls the address bus and the Read/Write line in the Apple in place of the 6502 during a DMA transfer.

The MC6844 has three modes of operation and four DMA channels. The operation modes are as follows: 1) Three State Control (TSC) Steal — in

this mode the MPU 2 clock is _stretched_ while one byte of data is trans-
ferred; 2) Halt Steal - the MPU is _halted_ while one byte of data is trans-
ferred; 3) Halt Burst - the MPU is _halted_ while an entire block of data is
transferred. The mode of operation selected depends on the transfer rate
required with the Halt Burst mode giving the highest rate. The preprocessor
DMA card was designed to use one channel of the MC6844 in the Halt Burst
mode.

The MC6844 pinout is shown in Figure 4.2. Before a DMA transfer begins,
the DMAC must be programmed with the data transfer location and length,
transfer mode, priority of servicing, data chaining, and interrupt control.
The programming is accomplished through the first five address lines (A0-
A5), the data bus, and the chip select pin ($\overline{CS}$) on the DMAC. Tables 4.3
and 4.4 show the programmable registers of the MC6844. These tables are
provided as a quick programming reference. More DMAC programming details
are provided in the Motorola specification sheets on the MC6844.

TxRQ0 through TxRQ3 (pins 29-31) are the transfer request lines. There
is one request line for each of the four channels. A peripheral device
requests a DMA transfer by driving the transfer request line on a par-
ticular channel HIGH. The preprocessor DMA card uses only channel #1 of
the DMAC.

Transfer acknowledge outputs notify the device requesting a transfer
that the request has been received by the MC6844. The transfer acknowledge
signals are TxAKA and TxAKB. The Transfer Strobe line ($\overline{TxSTB}$) is a general
acknowledge line which is also intended for use as the system Valid Memory
Address (VMA) signal.

After a transfer request has been received by the DMAC, the DMAC must
issue a request to the system's MPU. This is done with either of the two
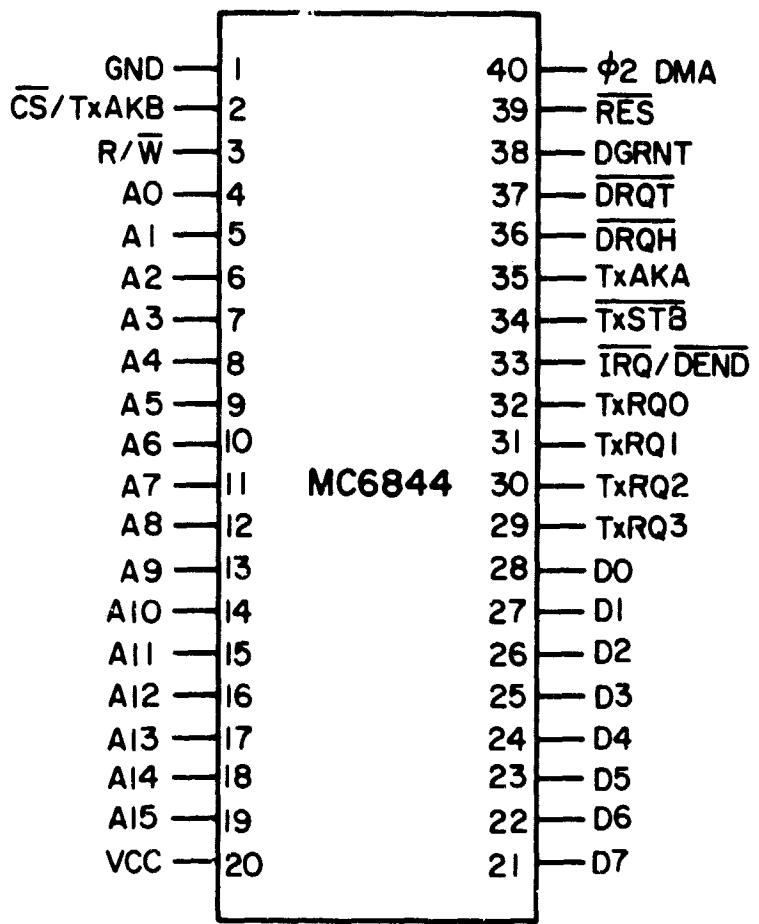
Figure 4.2 MC6844 DMAC pinout.

TABLE 4.3   MC6844 ADDRESS AND BYTE COUNT REGISTERS.

REGISTER	CHANNEL	ADDRESS (HEX)
ADDRESS HIGH	0	0
ADDRESS LOW	0	1
BYTE COUNT HIGH	0	2
BYTE COUNT LOW	0	3
ADDRESS HIGH	1	4
ADDRESS LOW	1	5
BYTE COUNT HIGH	1	6
BYTE COUNT LOW	1	7
ADDRESS HIGH	2	8
ADDRESS LOW	2	9
BYTE COUNT HIGH	2	A
BYTE COUNT LOW	2	B
ADDRESS HIGH	3	C
ADDRESS LOW	3	D
BYTE COUNT HIGH	3	E
BYTE COUNT LOW	3	F

TABLE 4.4   MC6844 CONTROL REGISTERS.

REGISTER	ADDRESS (HEX)	REGISTER CONTENTS							
		BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
CHANNEL CONTROL	1x*	DMA FND	BUSY/ READY	-	-	UP/ DOWN	TSC/ HALT	BURST/ STEAL	READ/ WRITE
PRIORITY CONTROL	14	ROTATE C'NTRL	-	-	-	R'QST ENB 3	R'QST ENB 2	R'QST ENB 1	R'QST ENB 0
INTERRUPT CONTROL	15	DEND IRQ FLAG	-	-	-	DEND IRQ ENB 3	DEND IRQ ENB 2	DEND IRQ ENB 1	DEND IRQ ENB 0
DATA CHAIN	16	-	-	-	-	2/4 CH'NL SELECT	DATA CHAIN SEL B	DATA CHAIN SEL A	DATA CHAIN ENB

*The x represents the binary equivalent of the channel desired.

outputs DMA Request TSC-Steal ($\overline{DRQT}$) or DMA Request Halt-Steal ($\overline{DRQH}$). The $\overline{DRQT}$ line is normally connected to the system clock driver and used to request a MPU clock stretch for the TSC-Steal transfer mode. The $\overline{DRQH}$ line is connected to the MPU Halt pin and requests a transfer in the halt steal or halt burst mode.

After issuing a request to the MPU, the DMAC waits for a bus available signal. This signal is generally output from the MPU or the MPU clock circuitry and is presented to the DMA Grant pin ($\overline{DGRNT}$). The DMA transfer begins once this signal is received.

The Interrupt Request output and the DMA End signal are provided on the dual purpose line $\overline{IRQ}/\overline{DEND}$. The $\overline{IRQ}$ output is used to interrupt the MPU and signal the peripheral device that a DMA transfer has ended. If the Interrupt has been enabled, the $\overline{IRQ}/\overline{DEND}$ line will go LOW _after_ the last byte of a transfer. The $\overline{IRQ}/\overline{DEND}$ line also goes LOW _during_ the last byte of a transfer to signal the DMA End. This occurs whether the Interrupt is enabled or not.

Other pins on the MC6844 include Read/Write (R/$\overline{W}$), the clock input ($\phi 2$ DMA), and Reset ($\overline{RES}$).

One option supported by the MC6844 and used by the preprocessing system is data chaining. Data chaining allows the repetitive reading or writing of a block of data without reloading the DMAC Address and Byte Count registers for each transfer. The chaining is performed by transferring the contents of the Address and Byte Count Registers in channel #3 to the channel selected by the programming of the Data Chain Control Register. The transfer automatically occurs after the Byte Count Register of the selected channel has decremented to zero.

It should be noted that during a DMA transfer the DMAC is only supply-
ing addresses for the system's memory at the MPU clock rate. The peripheral
device requesting the transfer must write (or read) the data on the data bus
at the MPU clock rate.

4.1.5 DMA circuit description. The Apple DMA circuit for the
preprocessing system was designed to transfer a block of data to (or from)
the Apple RAM at a rate of 1.023 Mbytes/sec. The circuit was constructed
on a 2-3/4" x 7" Apple II Peripheral Board and consists of the MC6844 DMAC
chip supported by ten low-power Schottky (LS) TTL chips. Connecting the
Apple and the DMA card to the peripheral device requesting a DMA transfer is
a 34-pin shielded ribbon cable. A connector for the cable is mounted on the
DMA board. The card was designed to operate in any of the Apple peripheral
slots 1 through 7 on the back of the main Apple board (see Section 4.1.3).
The Apple slot pinout and the ribbon calbe pinout are listed in Appendix I.
The locations of the components on the DMA card are noted in Appendix II.
Figure 4.3 is the wiring schematic for the DMA card. The timing for a
typical DMA transfer is shown in Figure 4.4.

Both the Apple ROM I/O locations and general I/O locations are used to
cmomunicate with the DMA card (see Section 4.1.3). The ROM I/O locations
access the DMAC Control Reg. ers while the general I/O locations are
decoded and used as "Control Signals." Table 4.5 lists the DMA card I/O
locations. The DMAC Control Registers are programmed through the first five
address lines (A0-A4), the data bus, and the chip select pin (CS) on the
DMAC (see Section 4.1.4). The CS pin must be pulled LOW while programming
the DMAC registers. This is accomplished by the decoding of the Apple I/O
Select line and the address lines A5 through A7 with a 74LS260 5-input NOR
(chip U6 in Figure 4.3). The decoding allows 32   addresses to activate the

ORIGINAL PAGE IS
OF POOR QUALITY
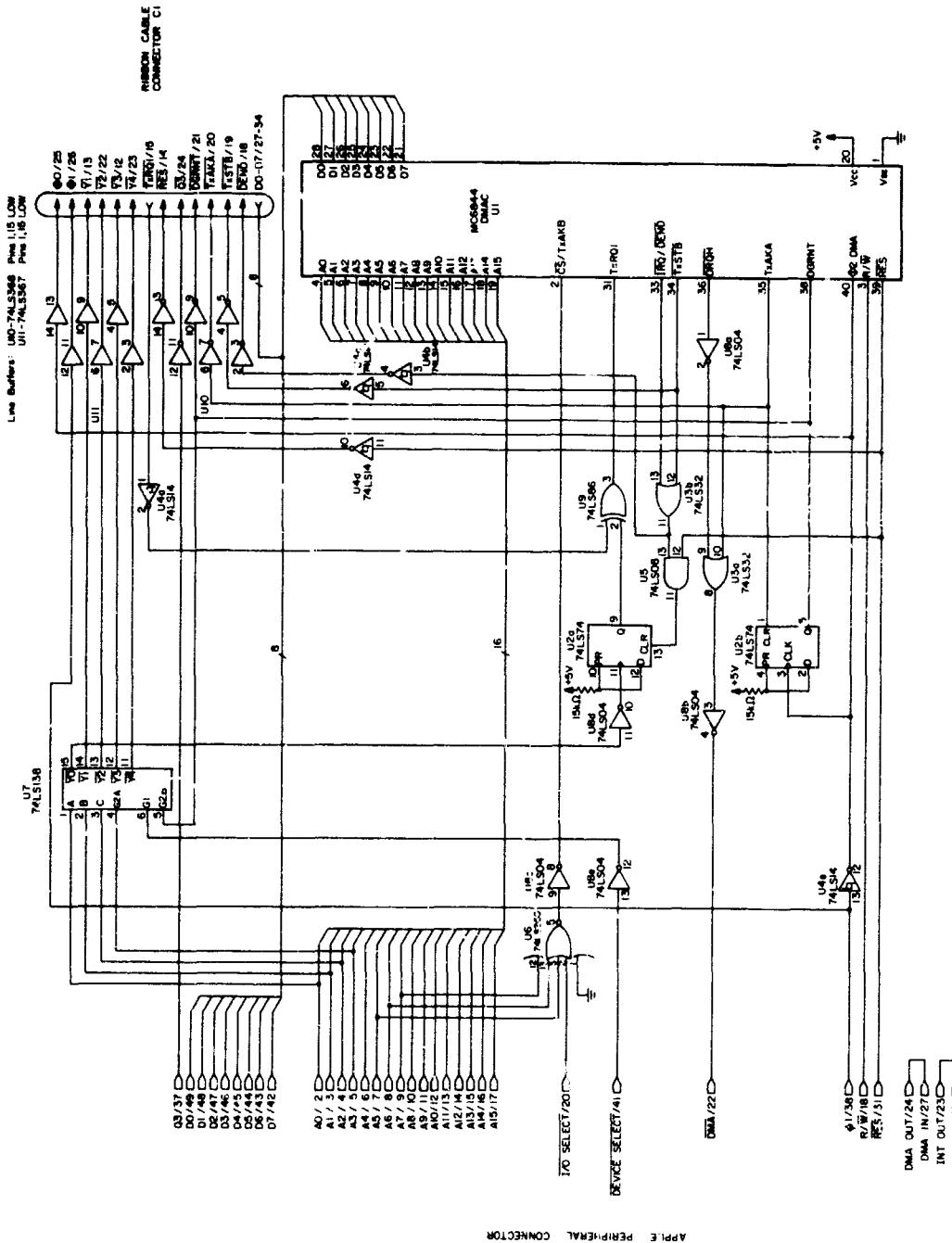
Figure 4.3 Apple DMA card circuitry.
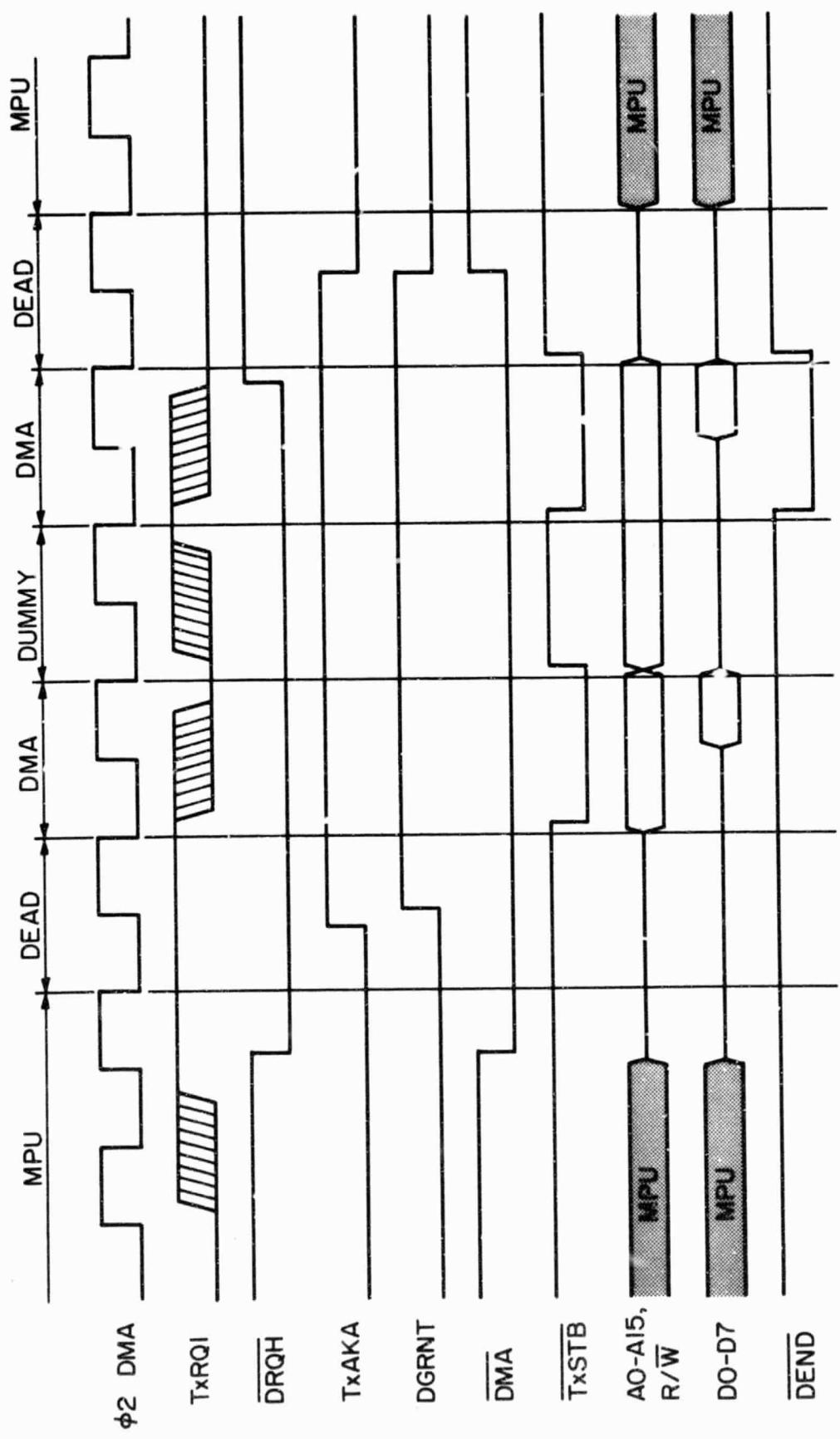
Figure 4.4 Apple DMA card transfer timing.

TABLE 4.5 APPLE DMA CARD I/O LOCATIONS.

LOCATION TITLE	ADDRESS[1]	DESCRIPTION[2]
DMAC CONTROL REGISTERS ($\overline{\text{I/O}}$ $\overline{\text{SELECT}}$)	$Cn00 - $Cn0F[3]	ADDRESS AND BYTE COUNT REGISTERS
	$Cn10 - $Cn13[3]	CHANNEL CONTROL REGISTERS
	$Cn14	PRIORITY CONTROL REGISTER
	$Cn15	INTERRUPT CONTROL REGISTER
	$Cn16	DATA CHAIN REGISTER
CONTROL LINES ($\overline{\text{DEVICE}}$ $\overline{\text{SELECT}}$)	$C0x0	$\overline{\text{Y0}}$ - APPLE DMA TxRQ
	$C0x1	$\overline{\text{Y1}}$ - (LASER TRIGGER)
	$C0x2	$\overline{\text{Y2}}$ - (AUX OUTPUT)
	$C0x3	$\overline{\text{Y3}}$
	$C0x4	$\overline{\text{Y4}}$ - (SLIPP UNIT RESET)

[1]All hexadecimal addresses. n = Apple slot#; x = n + 8.

[2]The items in parentheses are the functions performed by the Control Signals in the preprocessing system.

[3]See Tables 4.3 and 4.4 for addresses of particular registers for each channel.

$\overline{CS}$ line. Because the DMA card only uses channel #1 of the DMAC for trans-
fers and channel #3 for data chaining (see Section 4.1.4), only the pro-
gramming of the Control Registers of those channels is pertinent. The
Control Signals are generated by decoding address lines A0 through A2 with a
74LS138 3-to-8 line decoder (U7). The decoder is enabled by the address
line A3 and the $\overline{Device}$ $\overline{Select}$ line from the Apple. To ensure that no
spurious addresses are decoded during a DMA transfer, the 74LS138 is disabled
by the DGRNT signal.

Once the Control Registers are correctly loaded, a transfer request
issued to the DMAC will initiate a DMA transfer. The DMA card allows
requests from peripheral devices or the Apple itself. Generally, the
request is issued by a peripheral device needing attention. This is done by
pulling pin 15 on the ribbon cable LOW. This causes the TxRQ1 pin on the
DMAC to go HIGH which constitutes a transfer request. The Apple issues a
request through the Control Signal decoding logic (see Table 4.5). Reading
or writing to the transfer request address sets the output of a 74LS74 flip-
flop (U2a) HIGH. This also causes the TxRQ1 pin on the DMAC to go HIGH.
The flip-flop is needed to hold the transfer request HIGH throughout the DMA
transfer.

The DMA card was designed to use the MC6844 in the Halt Burst mode
which means an entire block of data is transferred sequentially. However,
the transfer may be suspended by withdrawing the transfer request on the
TxRQ1 pin of the DMAC. The timing diagram of Figure 4.4 shows this situa-
tion. For the first byte of the Halt Burst mode the DMAC tests the TxRQ1
signal on the rising edge of the $\Phi2$ DMA clock. In succeeding bytes the
TxRQ1 signal is tested on the falling edge of $\Phi2$ DMA, and data are trans-
ferred during the next $\Phi2$ cycle if TxRQ1 is HIGH. After a peripheral device

issues the transfer request (ribbon cable pin 15, LOW), simply withdrawing
the request will suspend the transfer. A peripheral device can also suspend
the transfer when the Apple has issued the transfer request. This is also
done with pin 15 on the ribbon cable. However, in this case, pulling pin 15
LOW suspends the transfer. This is a result of the exclusive-OR gate (U9)
used to combine the Apple and the peripheral request lines.

After receiving a transfer request the DMAC responds with two outputs,
Transfer Acknowledge A (TxAKA) and DMA Request Halt ($\overline{\text{DRQH}}$). One function of
the TxAKA signal is to notify the peripheral device that a request has been
received (ribbon cable pin 20). The TxAKA signal is also NOR-ed with DRQH
(chips U3a, U8a and U8b) and the resulting signal is used to pull the $\overline{\text{DMA}}$
line on the Apple peripheral connector LOW, stopping the 6502 in the Apple.
The TxAKA and $\overline{\text{DRQH}}$ signals are combined in order to hold the $\overline{\text{DMA}}$ line on the
Apple LOW beyond the final DMA transfer cycle (see Figure 4.4).

Once the DMA halt request has been presented to the Apple, the DMAC
expects a DMA Grant signal (DGRNT) to be returned. This must occur before
the transfer begins. Because there is no bus available (or equivalent)
connection in the Apple peripheral slot, DGRNT is synthesized from TxAKA
using a 74LS74 flip-flop (U2b).

During the DMA transfer the Transfer Strobe line ($\overline{\text{TxSTB}}$, ribbon cable
pin 19) acts as the Valid Memory Address (VMA) line. The completed DMA
transfer is marked by the $\overline{\text{DEND}}$ signal (ribbon cable pin 18). This signal
comes from the dual purpose pin $\overline{\text{IRQ}}/\overline{\text{DEND}}$ on the DMAC. The $\overline{\text{IRQ}}/\overline{\text{DEND}}$ and
TxSTB signals are OR-ed (U3b) to produce $\overline{\text{DEND}}$. This prevents false signals
arising from interrupt requests on the $\overline{\text{IRQ}}/\overline{\text{DEND}}$ DMAC pin. By examining the
DMA end flag, bit 7 of the Channel Control Register of channel #1 (location
$Cn11, n = Apple slot #), the Apple may determine if the block DMA transfer

has been completed. Reading of the Channel Control Register resets the DMA end bit.

Except for the data lines, all outputs to the ribbon cable are buffered through either a 74LS367 or a 74LS368 6-input buffer. Other signals not mentioned previously that are available on the ribbon cable include: the Apple clock signals $\Phi 0$, $\Phi 1$, and Q3, the Control Signals $\overline{Y1}$ through $\overline{Y4}$, and the Apple Reset line $\overline{RES}$.

As mentioned in Section 4.1.4 on the MC6844, during a DMA transfer the DMAC only supplies the Apple RAM with addresses and controls the Apple read/write line. The peripheral device sending (or receiving) the data must place the data on the data bus within the restrictions of the Apple read/write timing (see Figure 4.4).

4.2 SLIPP Interface Unit

4.2.1 Introduction. The SLIPP unit was designed to interface the Apple microcomputer to the transmitting and receiving sections of the lidar system. The unit consists of a digital circuit and its required power supplies all mounted in a 5 1/2" x 9" x 12" box. The circuit is constructed on a Vector 3682-2 6.5" long card which plugs into an edge connector inside the box. Two extra edge connectors are also available in the box. Mounted on the back panel of the box are the following connectors: 1) a 34-pin ribbon cable connector for linking the Apple DMA circuit to the SLIPP unit, 2) four isolated BNC connectors labeled LCP, NIM, TRIG and AUX OUT. LCP is the Laser Command Pulse input (+5 V pulse), NIM is the photon pulse input from the discriminator (NIM standard: - 18 mA (-1 V), 12 nsec), TRIG is the laser trigger output (+5 V, 1 μsec pulse), and AUX OUT is an auxiliary pulse output (+5 V, 1 μsec pulse).

Appendix I contains the pinouts for the SLIPP circuit board and the

external ribbon cable. The location of the components on the circuit board is shown in Appendix II. Photographs of the SLIPP unit are shown in Figure 4.5. Figure 4.6 is the wiring schematic for the circuit.

4.2.2 Clock circuitry. The SLIPP clock circuitry is shown in the lower lefthand corner of Figure 4.6. The clock circuitry performs two functions: 1) inhibits the rest of the SLIPP photon counting circuitry until the Laser Command Pulse (LCP) is received; and 2) after receiving the LCP, provides the timing signals that drive the counting circuitry.

Two 74H74 positive-edge-triggered flip-flops are used to provide the enabling signals for the system. The first flip-flop (U12a) must be clocked by the user. This action enables the second flip-flop (U12b) which is then used to detect the LCP and provide the General Enable (GEN ENB) signal to the rest of the circuit. The first flip-flop is needed to ensure that the SLIPP system will not accidentally be enabled by a spurious LCP before the actual data run is to take place. Normally, the clocking of the first flip-flop is done by the pulse sent to trigger the laser (Control Signal $\overline{Y1}$ from the Apple DMA card). However, the Data Acquisition Enable Jumper, shown near the lower-center section of Figure 4.6, can be changed so that the user can clock the first flip-flop at his discretion (using Control Signal $\overline{Y3}$ from the Apple DMA card).

The master timing signal is supplied by a free-running 20 MHz oscillator. The 20 MHz signal is input into a 74162 decade counter (U23) divides the signal down to 2 MHz. However, the decade counter must be enabled by the GEN ENB signal before outputting the 2 MHz signal. In this way, no timing signals are supplied to the rest of the counting circuitry until the LCP has been received. The decade counter allows only up to 50 nsec skew time between the LCP and the beginning of the first 2 MHz clock cycle.

(a)
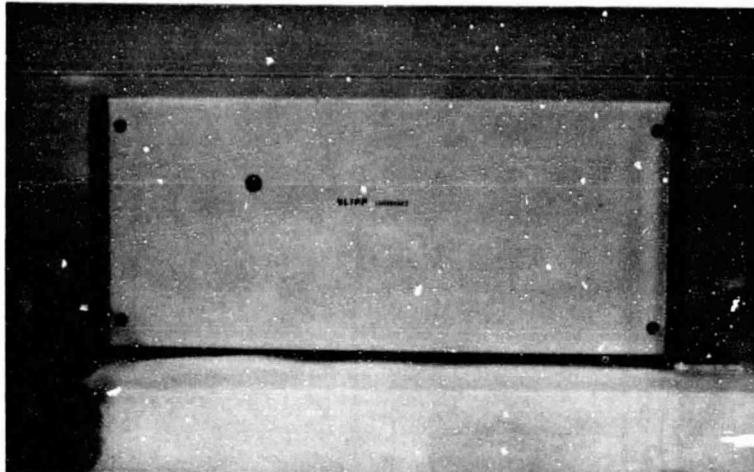
(b)
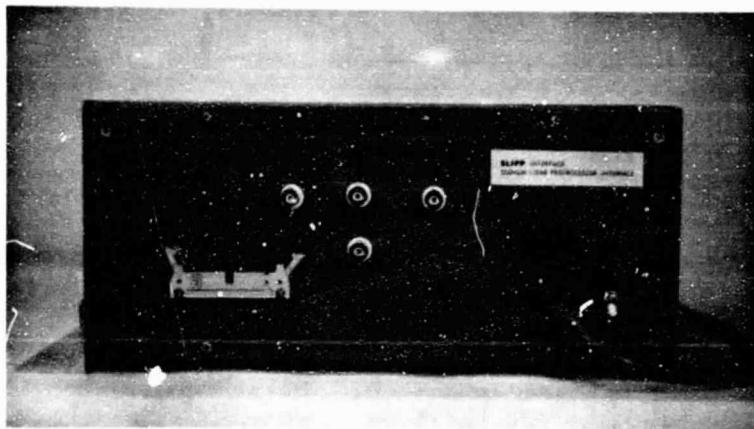
(c)
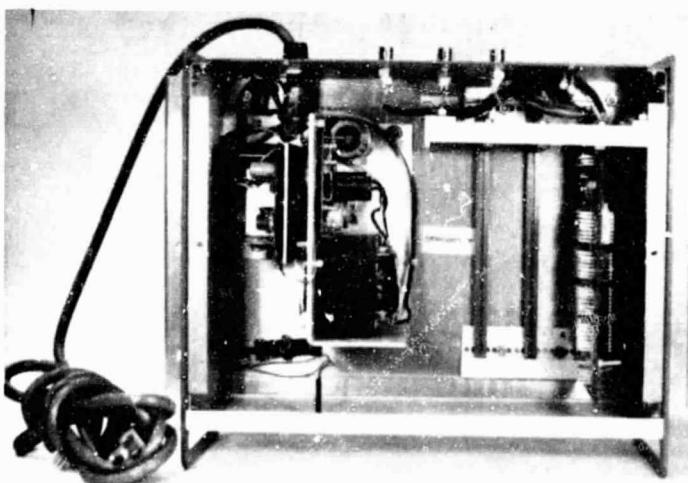
Figure 4.5 Photographs of the SLIPP unit: (a) front panel, (b) back panel, and (c) top view of the open cabinet.

ORIGINAL PAGE IS
OF POOR QUALITY.
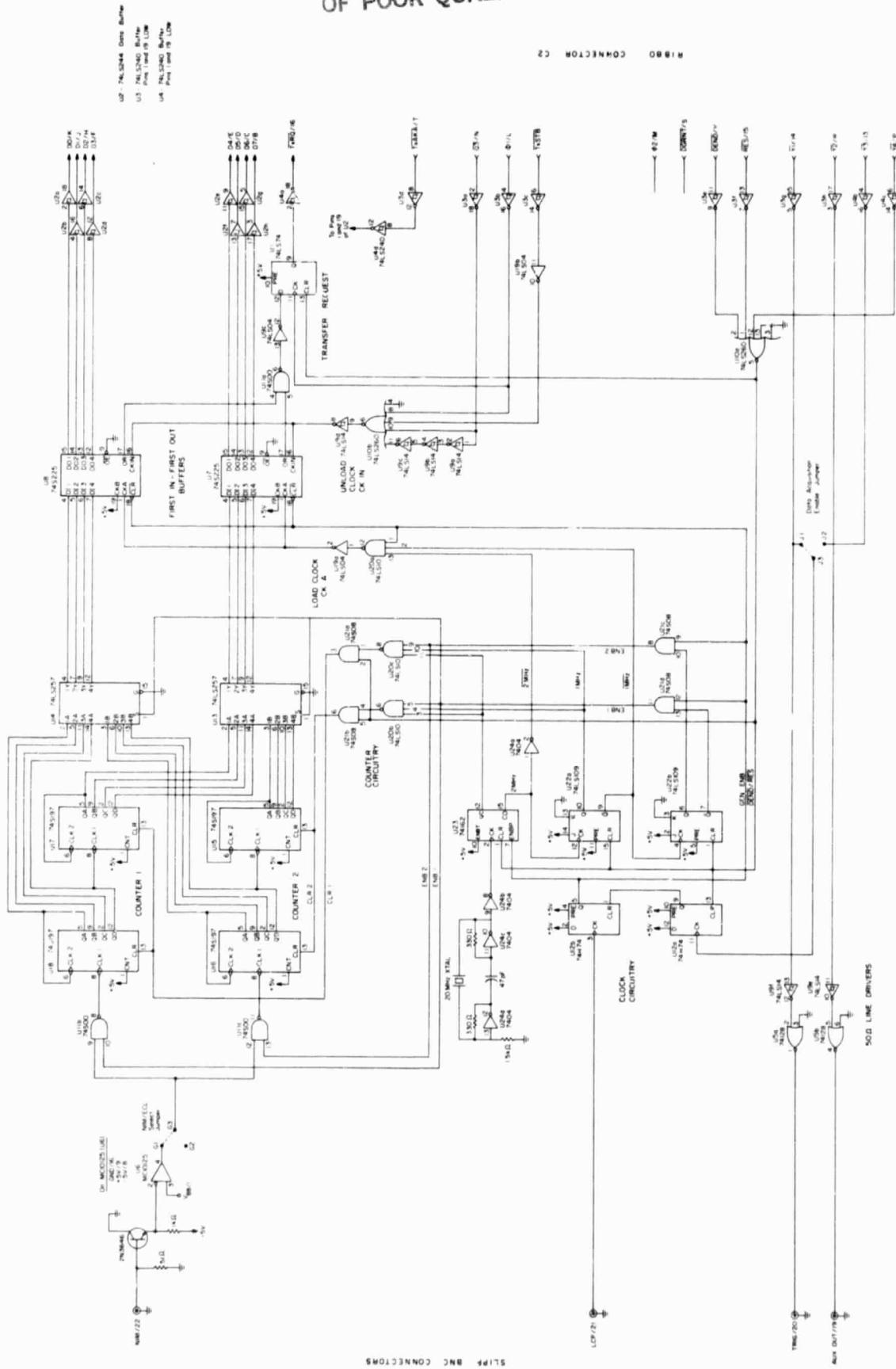
Figure 4.6 SLIPP unit circuitry.

After the decade counter is enabled, two 74LS109 positive-edge-triggered flip-flops (U22a and U22b) are used to divide the 2 MHz signal down to 1 MHz and 0.5 MHz. The inverted signals $\overline{2\text{ MHz}}$, $\overline{1\text{ MHz}}$, and $\overline{0.5\text{ MHz}}$ are also generated by the circuitry. The 0.5 MHz and $\overline{0.5\text{ MHz}}$ signals are labeled ENB 1 and ENB 2, respectively. These signals are used to select the toggling photon counters discussed in Section 4.2.3. Two 74S08 NAND gates (U21c and U21d) are used to hold both ENB1 and ENB 2 LOW while the GEN ENB signal is inactive (LOW).

Figure 4.7 shows the timing relations for various clock circuitry signals immediately following a laser shot. The times noted on the diagram were calculated using typical propagation delay values for the particular chips involved.

4.2.3 <u>Counting circuitry</u>. The counting circuitry is shown in the upper left-hand corner of Figure 4.6. The circuitry takes the NIM standard pulses from the lidar system discriminator and counts them in consecutive 1 μsec intervals.

The counters used in the circuitry are TTL binary counters so it is necessary to shift the NIM pulses to TTL levels. This is done in two stages. First, a 2N3646 transistor configured as an emitter-follower shifts the NIM pulses to emitter-coupled logic levels (ECL: -0.9V = "0", -1.7 V = "1"). A Motorola MC10125 ECL to TTL converter chip (U6) is then used to obtain positive TTL pulses. A NIM/ECL select jumper has been provided in the circuit for possible future upgrade to ECL level inputs.

Two 8-bit counters in a toggling configuration are used in the circuit to obtain the photon counts. At any given 1 μsec interval during a data run, one counter is counting pulses while the contents of the other counter,

Figure 4.7 SLIPP unit clock circuitry enable timing.
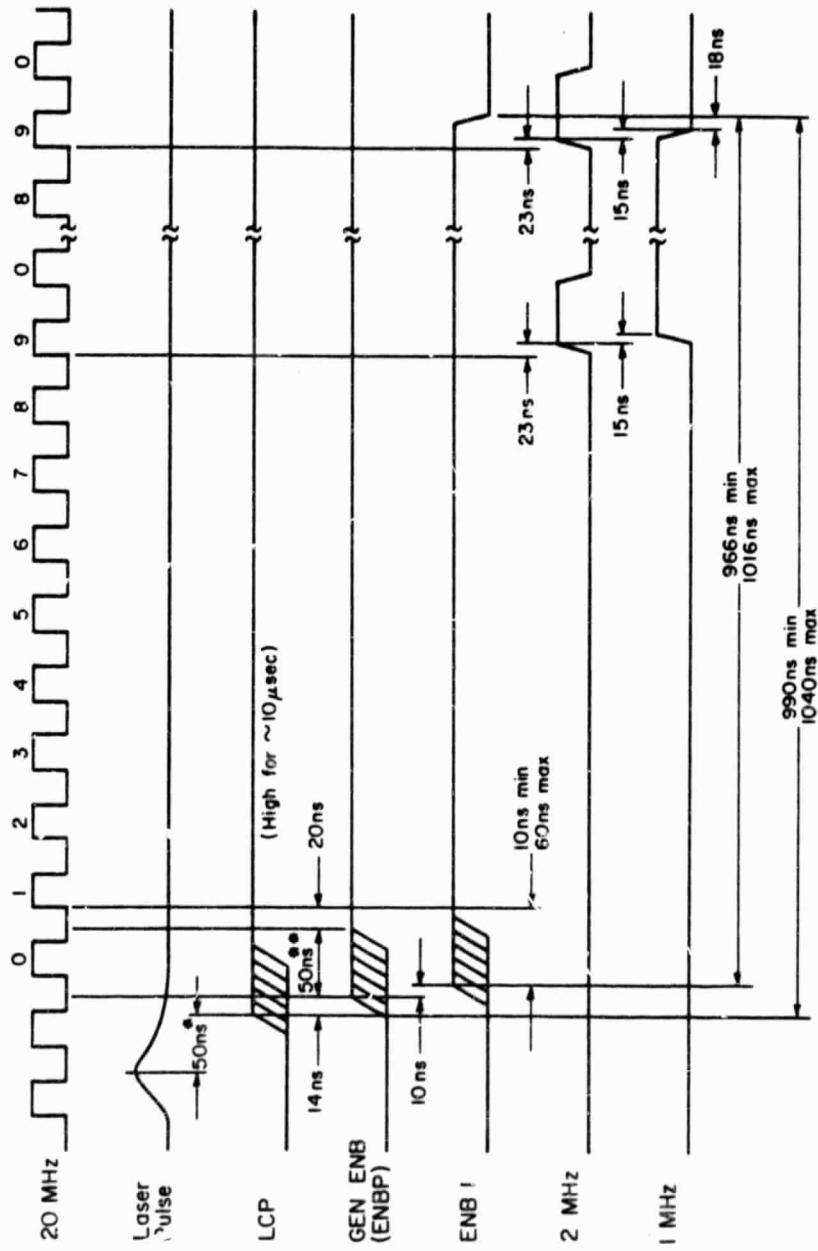
*The laser pulse is actually about 2 μsec in duration. The 50 nsec time shown is an estimate of the delay
in the laser pulse detector from the time the laser pulse first triggers the detector.

**50-nsec skew time.

obtained during the previous interval, are being placed in Apple memory. Each counter is composed of two cascaded 74S197 4-bit binary counters (counter #1: chips U18, U17; counter #2: chips U16, U15).

The positive TTL photon pulses are input to the counters through two 74S00 NAND gates (U11b and U11c). These gates use the ENB 1 and ENB 2 signals from the clock circuitry to section the pulses into 1 μsec intervals and select the counter to receive the pulses. The gates also invert the photon pulses for input to the 74S197 counters.

During the inactive (non-counting) cycle of a counter, two actions are performed. First, the contents of the counter are stored; and second, the counter is cleared. Two 74LS257 quad multiplexer chips (U13, U14) select the counter output to be stored. The ENB 1 signal is used to drive the select input on the chips. The multiplexers can be thought of as operating 180 degrees out of phase with the input NAND gates (U11b, U11c). The output of the multiplexers is clocked into the Apple interface circuitry described in the next section. The Qc (3rd bit) output of the decade counter in the clock circuitry (U23) is used to clear the counters. Two 3-input NAND gates (U20b, U20c) and two AND gates (U21a, U21b) determine when and which counter to clear.

The important timing signals for the counting circuitry are shown in Figure 4.8. The Data Load Clock (CK A) signal in the figure is described in the next section.

4.2.4 Apple Interfacing Logic. The logic interfacing the SLIPP unit and the Apple microcomputer allows the following functions to be performed: 1) the resetting of the SLIPP circuitry, 2) the firing of the laser and the enabling of the SLIPP counting circuitry, and 3) the DMA data transfer from the SLIPP unit to the Apple. The interfacing logic is generally shown in
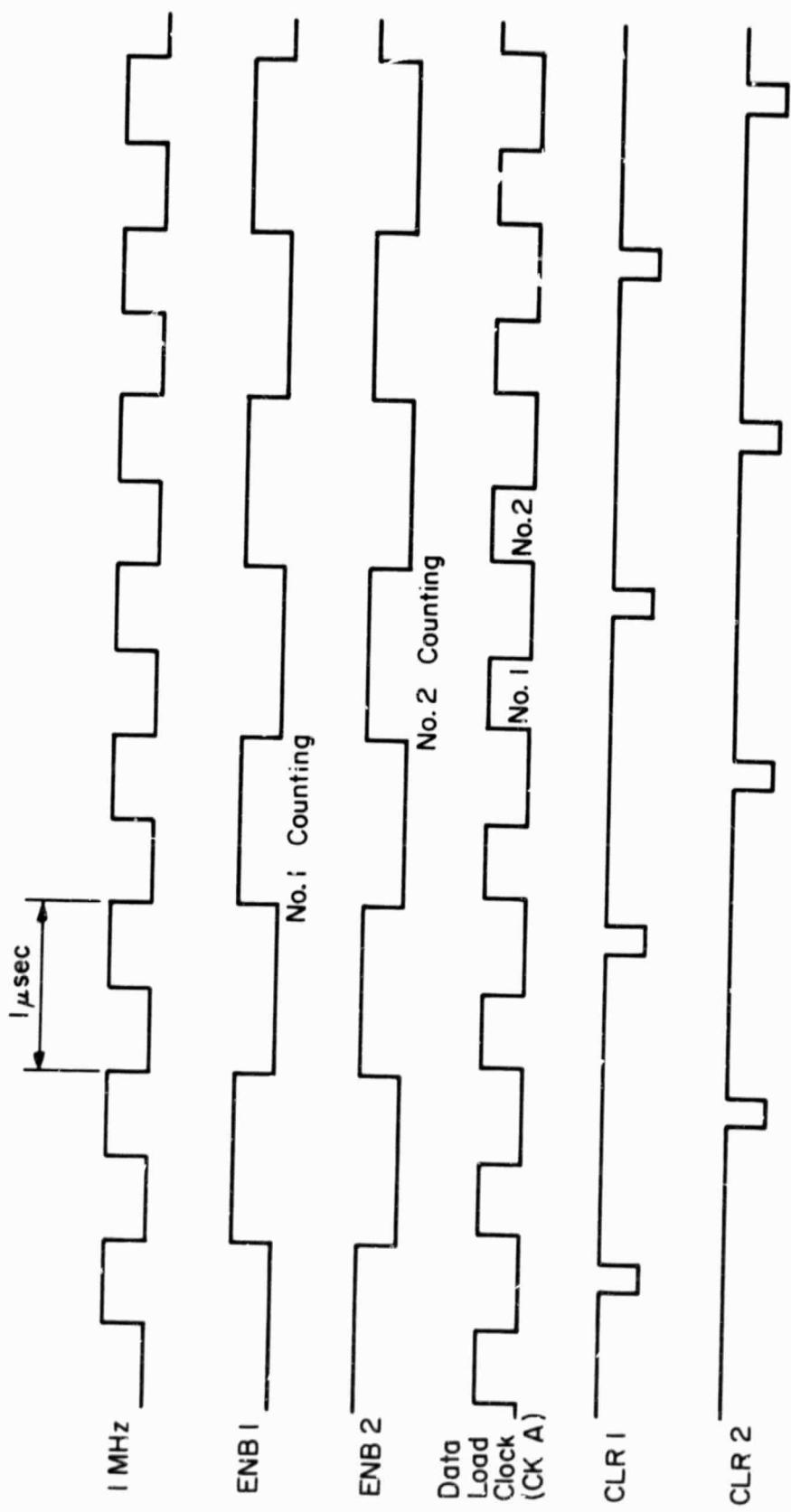
Figure 4.8 SLIPP unit counting circuitry timing.

the SLIPP unit to the Apple. The interfacing logic is generally shown in the right-hand side of Figure 4.6.

Three different signals from the Apple DMA card will reset the SLIPP circuitry. The signals are the Reset signal ($\overline{RES}$) which is connected to the Apple reset line, the $\overline{Y4}$ Control Signal, and the DMA End ($\overline{DEND}$) signal which comes at the end of a DMA transfer. The three signals are input to a 74LS260 5-input NOR gate (U10a) which is shown in the lower right-hand corner of Figure 4.6. The output of the NOR gate is used to reset the clock circuitry flip-flops discussed in Section 4.2.2, clear the counters discussed in Section 4.2.3, and reset the transfer request flip-flop which will be discussed further in this section.

The $\overline{Y1}$ Control Signal from the Apple DMA card is normally used to trigger the laser. The $\overline{Y1}$ signal on the ribbon connector (C2) is input to a 74128 50-Ohm line driver (U5a) shown in the lower left-hand corner of Figure 4.6. The line driver output is connected to the TRIG BNC connector on the back panel of the SLIPP box. Control Signal $\overline{Y1}$ is also used to enable the clock circuitry flip-flops. A second line driver (U5b) is used to drive the Auxiliary Output (AUX OUT), and Control Signal $\overline{Y2}$ is used to pulse the output. More details on the laser firing and the circuit enabling are discussed in Section 4.2.2 on the SLIPP clock circuitry.

As mentioned in Section 4.1.2 on Apple Timing, the Apple and the SLIPP unit operate asynchronously. The SLIPP unit photon counters during a data run output data at a rate of 1.0 Mbytes/sec while the Apple DMA circuitry, operating at the Apple $\Phi0$ clock rate, accepts data at a 1.023 Mbytes/sec rate. To transfer smoothly, the data bytes from the SLIPP unit to the Apple a First-In/First-Out buffer technique was used. The buffer circuitry is shown in the upper right-hand corner of Figure 4.6 and is sectioned into the following parts: the First-In/First-Out buffers, the Load Clock (CK A), the

Unload Clock (CK IN), and the Transfer Request circuit.

Two 74S225 16 x 5 asynchronous First-In/First-Out (FIFO) buffer chips (U7 and U8) are the heart of the buffer system. Data bytes from the SLIPP unit's counting circuitry are clocked into the FIFOs with the Load Clock (CK A) circuitry. The Unload Clock (CK IN) circuitry uses the Apple Clock signals Q3 and Φ1 and the Apple DMA signal $\overline{\text{TxSTB}}$ to clock data out of the FIFOs to the Apple RAM.

Because the Apple DMA circuit transfers data bytes faster than the SLIPP unit can supply them, the DMA circuit must be signaled periodically to wait during a data run. This is effected through the Transfer Request circuitry. The Transfer Request circuit uses the Output Ready (OR) lines on the FIFOs. These lines are active (HIGH) whenever a valid byte is in the FIFO's output register. The OR lines from the two FIFOs are AND-ed together (U11a and U19c) and the resulting signal is input to a 74LS74 positive-edge-triggered D-type flip-flop (U1). The flip-flop essentially tests the composite OR signal on the rising edge of the Apple Φ0 clock. If the OR signal is HIGH when tested, the transfer request to the Apple DMA circuit becomes (or remains) active. If the FIFOs output buffers are not ready when the OR signal is tested (OR signal LOW), the transfer request to the DMA circuit is withdrawn. This will halt the DMA process until the transfer request is restored (normally the next rising edge of Φ0).

All lines connected to the 34-pin Apple-SLIPP ribbon cable are buffered with either inverting or non-inverting line drivers/receivers. The buffer chips consist of two 74LS240 octal inverting buffers (U3 and U4) and one 74LS244 octal non-inverting buffer (U2). The 74LS244 three-state buffers are used to drive the data lines D0-D7 and are activated by the DMA signal

TxAkA during a DMA transfer.

Figure 4.9 is a timing diagram for various SLIPP unit and Apple signals during the data collection for one laser shot. The signals in the upper half of the figure are SLIPP unit signals while Apple signals are shown in the lower half. The left-hand section of the diagram shows the beginning of the data collection. The middle section shows an instance where the Apple DMA transfer is suspended and the right-hand section shows the end of the data collection.

One consequence of the interface logic design described above is that the number of range bins collected by the SLIPP unit is set by the programming of the MC6844 DMAC Byte Count register on the Apple DMA card. If the Byte Count register is programmed for a transfer of N bytes, the SLIPP unit will collect N-1 range bins. The minus one arises from the fact that during the last byte of the DMA transfer the SLIPP unit FIFOs are being reset.

4.2.5 Power supplies. Two power supplies mounted in the SLIPP box supply +5 V and -5 V to the SLIPP circuits (Figure 4.10). A Power One C5 - 6 Amp supply provides +5 V and a Power One HA5 - 1.2 Amp supply is connected as a -5 V supply. The DC ground on the supplies is referenced to the Apple logic ground through the SLIPP-Apple ribbon cable. The AC input to the supplies is fused at 1.5 A and a switch located on the back panel of the SLIPP box turns on the supplies.

50
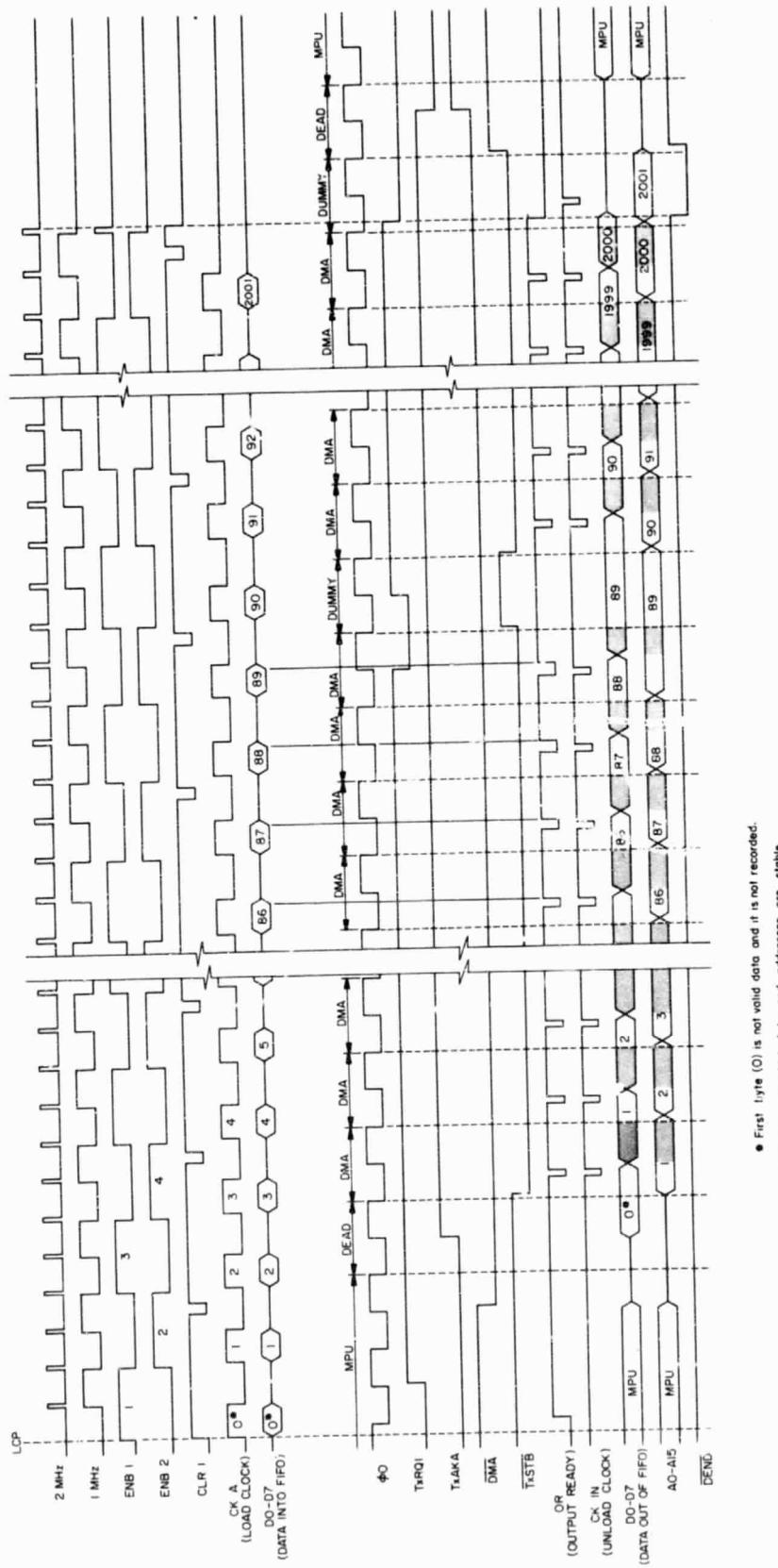
Figure 4.9 SLIPP unit and Apple timing signals during data collection.
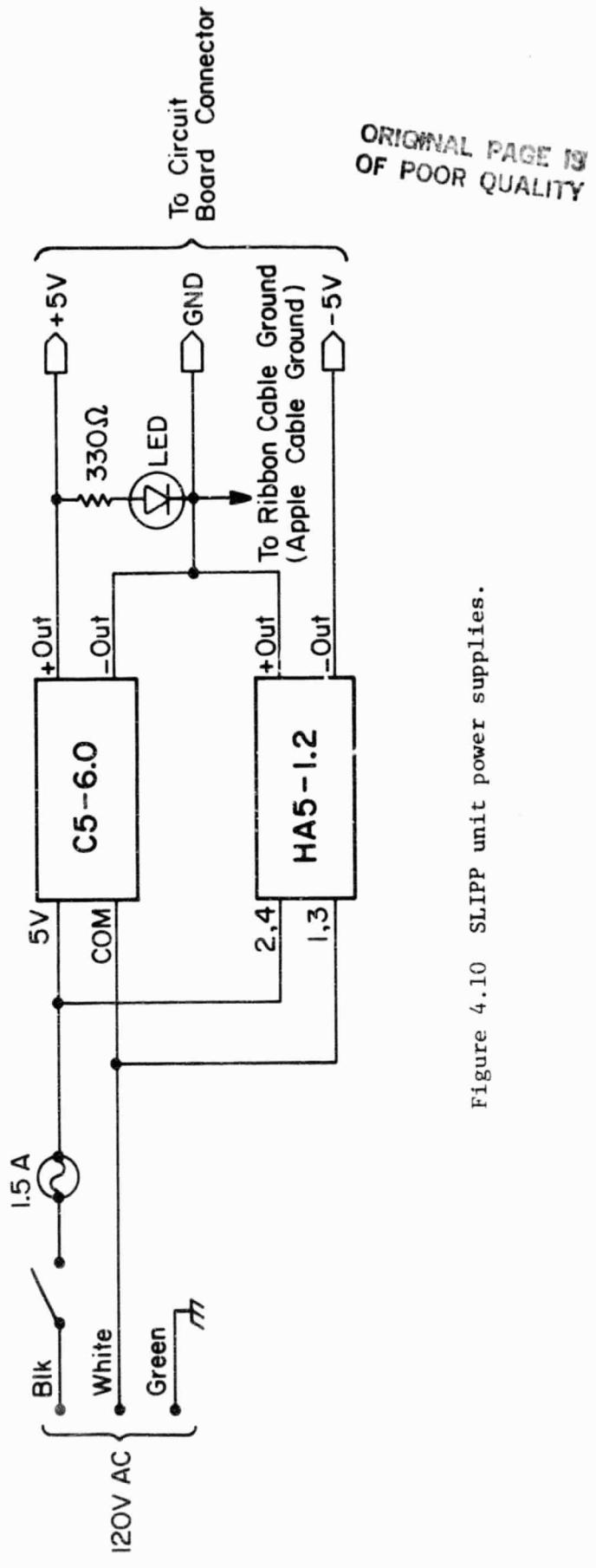
Figure 4.10 SLIPP unit power supplies.

5. SOFTWARE DESCRIPTION

5.1 Introduction

Three software routines were written for use with the preprocessing
system. Two of the routines are run during data collection. They are
SLPAPP.OBJ which drives the Apple in the preprocessing system and FZZ.SAV
which drives the LSI-11. The third routine CONVRT.SAV is run on the LSI-11
after data collection is complete. CONVRT.SAV takes the binary data files
created on a floppy disk during data collection and converts them to ASCII
files. This is a necessary step before transferring the data files to the
University of Illinois CYBER computer for further processing. The Apple
program was written in 6502 machine language with the Apple 6502 Assembler/
Editor on the Apple Tool Kit disk. The LSI-11 programs contain both Fortran
and Macro-11 subroutines written under the RT-11 Version 4 operating system.

A general discussion of the data collection software is contained in
Section 5.2. The routines used by the Apple and the LSI-11 for communi-
cations between each other are an integral part of the collection software.
The communication routines for both computers are quite similar and are
presented in Section 5.3. Section 5.4 contains notes on some of the options
available in the collection software. Section 5.5 is a discussion of the
data file conversion software. All the program listings are presented in
Appendix III.

5.2 Data Collection Software

5.2.1 <u>Apple routines</u>. The file SLPAPP.OBJ, containing the Apple
data collection software, is sectioned into five parts which are labeled:
the Main Program Loop, the Apple Receiver, the Apple Sender, Data Run, and
Printing Routines. The Apple Receiver and Sender sections handle the Apple

side of the Apple-LSI-11 communications. These routines are discussed in Section 5.3 on Apple and LSI-11 communications. The Printing Routines are a set of subroutines which display messages on the Apple monitor. Details of the Main Program Loop and the Data Run sections are discussed below.

Part (a) of Figure 5.1 shows a flowchart of the Main Program Loop section. When the file SLPAPP.OBJ is run, execution begins at this section. First, a keyboard input routine is loaded. This allows keystrokes on the keyboard to be accepted by the Apple. Next, a heading is displayed on the monitor and the program waits for a keypress before continuing. After the keypress the CCS serial card (for communication with the LSI-11) and the interrupt and byte input vectors (INTV and STATE) are initialized for use with the Apple receiving code. Finally, the interrupt disable bit is cleared, the message "IDLE" is displayed on the monitor, and the routine begins the idle wait loop. Presently, the idle loop is a simple waiting loop but in the future the loop can be replaced with a background routine such as a laser tuning algorithm.

The Data Run section of the program is actually a subroutine labeled DATRUN which is called by the Apple Receiver routine after receiving the "data run" frame. A flowchart of DATRUN is shown in part (b) of Figure 5.1. Before DATRUN is called the two variables REPNUM and MSHOTS must be loaded. REPNUM is used by DATRUN to set the laser pulse repetition rate while the desired number of laser shots per profile is contained in MSHOTS. The values to be loaded into the two variables are sent from the LSI-11 to the Apple within a "commands" communication frame.

Once called, DATRUN begins by displaying the message "TAKING DATA" on the Apple monitor. Next, the laser shot counter TSHOT is loaded with the value contained in MSHOTS and the MC6844 DMAC registers are loaded. The
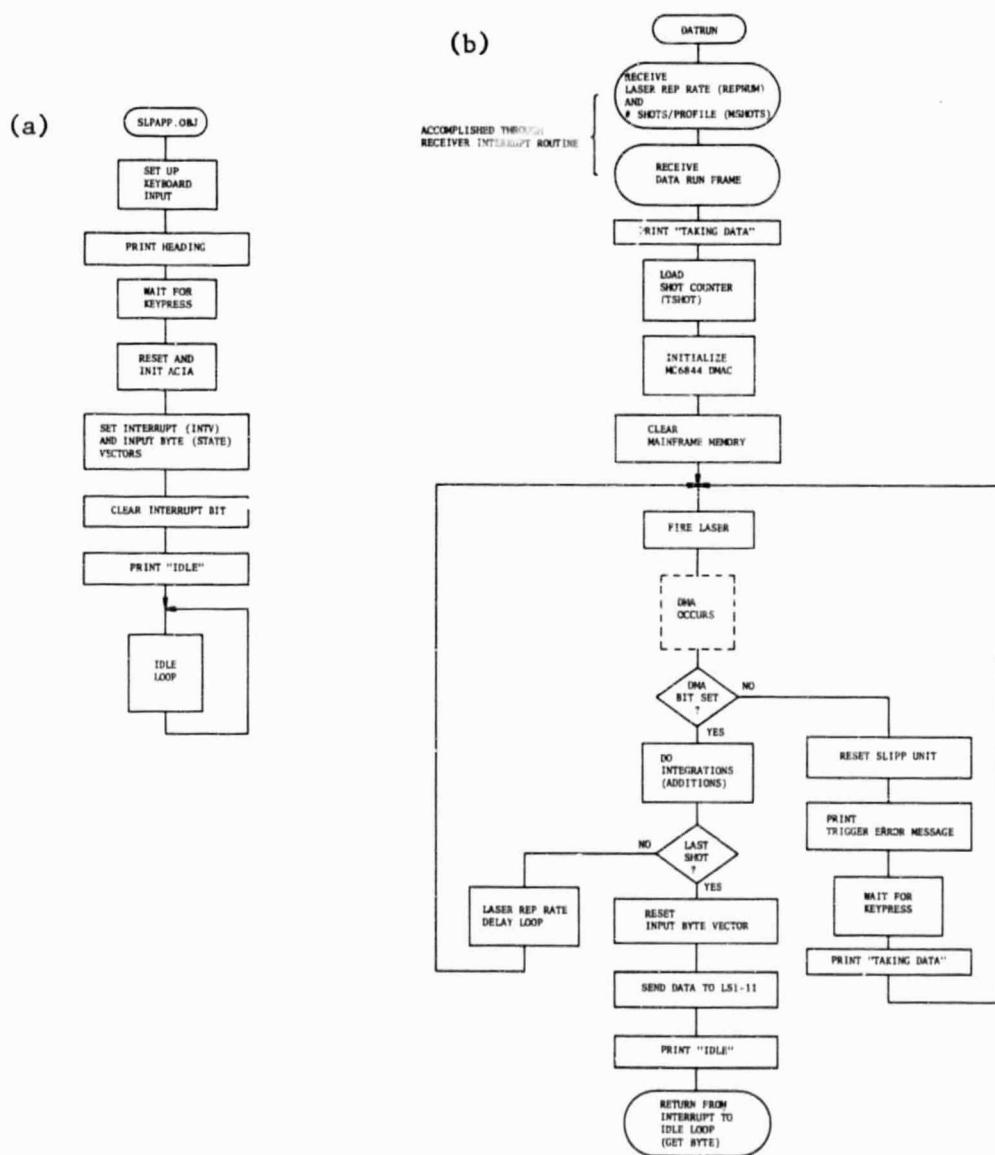
Figure 5.1 Flowchart of (a) SLPAPP Main Program Loop and
(b) SLPAPP Data Run sections.

mainframe memory (the block of memory used to store the profile integration sum) is also initialized.

After the initializations the profile collection begins. The laser and the SLIPP unit are triggered by accessing the trigger address (Control Signal Y1). The routine execution is suspended during the range bin collection and DMA transfer. When the execution resumes, a status bit in the MC6844 DMAC on the Apple DMA card is tested. If the bit is not set the DMA transfer never occurred and control jumps to an error rc .ne. With the bit set, the Apple integrates (adds) the new range bin counts into the profile integration sum in the mainframe memory. The 16-bit addition routine used causes the high order and low order bytes of the 16-bit data words to be stored in separate blocks of memory. This algorithm was selected because of its fast execution time. After the integrations, the shot counter TSHOT is decremented and tested for a zero value. If more laser shots are required for the profile, the routine executes a delay loop and then returns to fire the next laser shot.

The delay loop is required in order to regulate the laser pulse repetition rate. The value stored in the variable REPNUM is used to set the length of the delay. An equation taking into account the number of Apple machine language instructions between successive laser shots, the desired pulse repetition rate, and the structure of the delay loop code is used by the LSI-11 to solve for the value to be stored in REPNUM. REPNUM is a one-byte location so values ranging from 0 to 255 can be assigned to it. As mentioned previously, the REPNUM value is sent from the LSI-11 to the Apple within a "commands" frame.

After the data collection and integration are completed for the last laser shot of a profile, the data are sent to the LSI-11 computer within a

"data" communication frame. (~efore sending the "data" frame, the input byte vector STATE must be reset to point to the beginning of the receiving code. This allows the "acknowledge" communication frame sent by the LSI-11, verifying the successful receipt of the data, to be properly interpreted.) Finally, the "IDLE" message is displayed and a return from interrupt is executed to reenter the idle wait loop in the Main Program Loop.

5.2.2 LSI-11 routines. The LSI-11 data collection software is comprised of twelve Fortran and Macro-11 files which are compiled and linked into one executable file labeled FZZ.SAV. The twelve files are: FZZ.FOR, EXPPAR.FOR, CSTTUS.FOR, LAALN.FOR, ALNRTN.FOR, DATRUN.FOR, EXMPRF.FOR, DISCAL.FOR, RCVER.MAC, SENDER.MAC, DISPLY.MAC, and HDUMP.MAC. In order to decrease file manipulation problems, all the compiled versions of the files (except for FZZ.OBJ) were placed in the library file SLPLIB.OBJ. Figure 5.2 shows a general flowchart for the collection program FZZ.SAV. Descriptions of each of the twelve files in the collection program are presented below.

FZZ is the main program file. All variables and constants for the collection software are initialized by this file. FZZ also initializes the serial port for Apple-LSI-11 communications by calling the subroutine INITR (INITR is contained in the file RCVER). The LSI-11 Job Status Word is set for "no terminal wait state." This allows the use of the Fortran input function ITTINR in the following manner: A character is transferred from the console terminal to the user's program if a character is present; but program execution continues whether a character is present or not (see Section 5.4 for the use of this option). After the initializations, FZZ displays the main collection software menu. Selection of one of the seven available options in the menu results in a call to a particular subroutine. The first six options (1-6) call the subroutines EXPPAR, CSTTUS, LAALN,
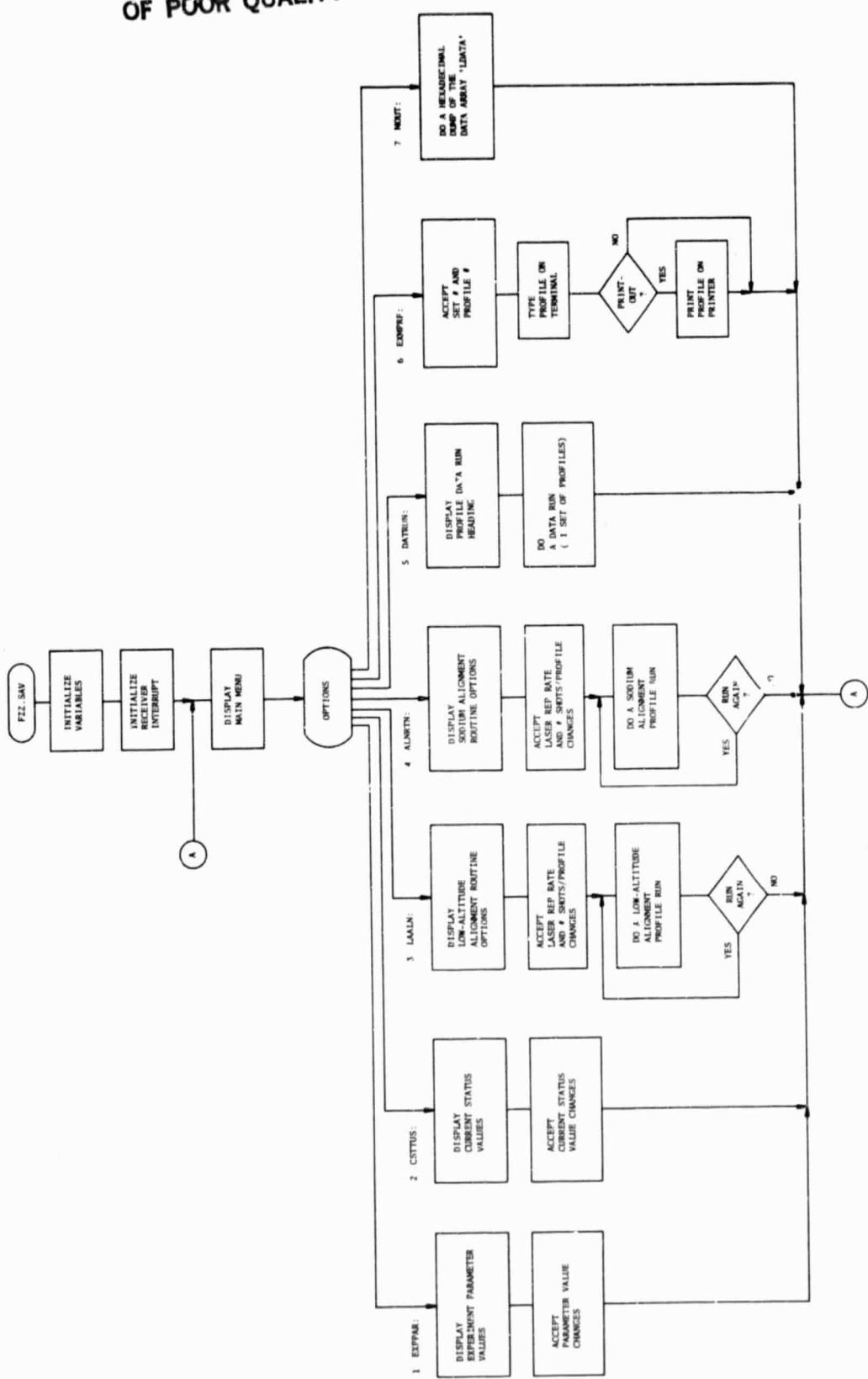
Figure 5.2 Flowchart of LSI-11 main collection program FZZ.

ALNRTN, DATRUN, and EXMPRF, respectively. Option 7 is not actually displayed in the menu but it is a viable option. Selecting option 7 results in a call to the subroutine MOUT which is contained in the file RCVER.

EXPPAR displays the experiment parameters and allows them to be changed by the operator. The parameters are used by the two routines which obtain sodium data: ALNRTN and DATRUN. A brief description of each of the parameters follows:

#Sets - number of data sets of profiles to be collected before return-
ing to the main menu. (Presently, this is a dummy variable and is
not utilized by the software.)

#Profiles Per Set - number of profiles to be placed in a data set.

#Laser Shots Per Profile - number of laser shots per profile.

Inter-Profile Delay - extra delay time desired between the collection
of successive profiles (seconds).

Elevation Angle - angle between horizontal and receiving telescope
line-of-sight (degrees).

Base Altitude - altitude of the lidar system in reference to the
earth's surface (feet and kilometers).

Altitude Range of Data - altitude range of sodium data to be stored
on a floppy disk (kilometers).

Range Bins - the set of range bins to be stored on a floppy disk.
The Elevation Angle, Base Altitude, Altitude Range of Data, and Range Bins parameters are all related through a set of range equations, and changing one of the parameters results in a recalculation of the others (see the EXPPAR listing in Appendix III.3 for details).

CSTTUS displays a table of values that reflect the current status of the experiment. Included in the table are: the current profile and set

numbers (the numbers by which the next profile collected will be identified),
the desired number of laser shots per profile, and the sodium values as
calculated by the subroutine DISCAL for the last recorded profile. CSTTUS
also allows the operator to change the current profile and set numbers.

LAALN directs a low-altitude data collection run for lidar system
alignment purposes. After being called, the first action of LAALN is to
allow the operator to select the laser repetition rate and the number of
laser shots needed for each low-altitude profile. Next, the subroutine
requests the collection of one profile and then displays the completed
profile range bin counts for bins 1 through 600 (corresponding to a range of
0 to 90 km).

The steps followed by LAALN to obtain the profile are similar to those
described in the paragraph on DATRUN, except it is important to note that
the profile is not permanently saved on a floppy disk. After the counts are
displayed the operator is prompted for a decision on whether to run the
routine again or to return to the main menu. The low-altitude counts
observed with the help of this routine are in general due to Rayleigh scat-
tering. By using the routine in an iterative fashion the critical alignment
of the laser beam and the receiving telescope field-of-view can be accom-
plished.

ALNRTN directs a sodium data collection run for lidar system alignment
purposes. ALNRTN has exactly the same format as LAALN, the low-altitude
alignment routine, except that each profile collected is examined for sodium
returns. ALNRTN calls the subroutine DISCAL to provide a summary of the
sodium counts. Also, ALNRTN uses the two experiment parameters Base
Altitude and Elevation Angle as set in the subroutine EXPPAR. The two
parameters are necessary for the selection of the range bins that should

contain sodium counts (see the paragraph on DISCAL). ALNRTN is helpful when making final laser tuning and laser-telescope alignment adjustments.

DATRUN directs the collection of a set of profiles with the details of the collection process set by the experiment parameters of the subroutine EXPPAR. Initially, DATRUN displays the storage space (blocks) needed on the floppy disk to save the forthcoming set of profiles. Also displayed is the record length (double words) of each profile in the set. These values are provided to help keep track of the space remaining on the data floppy disk. Along with the block length and record size values, the operator is prompted to start the data run. Once the run begins, DATRUN updates the file name to be used on the data floppy disk for identifying the forthcoming set of profiles. Also, the range parameters required by DISCAL for the sodium count summary are calculated. Values for the two variables REPNUM and MSHOTS are also determined. These variables are required by the Apple data collection software and are sent from the LSI-11 to the Apple within a "commands" communication frame. A "data run" frame sent from the LSI-11 to the Apple requests the collection of a profile. The setting of the flag DFLAG alerts the routine that the profile data has been collected and received successfully by the LSI-11. An error in the transmission of the profile data from the Apple to the LSI-11 results in the setting of the flag ERRFLG, and program control is returned to the main software collection menu. DATRUN calls the subroutine DISCAL to provide a summary of the sodium counts and the profile is saved on the data floppy disk. If more profiles are required to complete a set, the profile number is updated and collection begins again. A few details of the subroutine DATRUN are also discussed in Section 5.4 on Software Collection Options.

EXMPRF allows the operator to examine the range bin counts of profiles

that have previously been collected and stored on a floppy disk. EXMPRF
prompts the operator to enter the set and profile numbers of the profile to
be examined. The subroutine expects the requested data to be on a floppy
disk in the disk drive designated as the data storage drive (see Section
5.4). The range bin counts are displayed on the console terminal and a
printout of the counts can be obtained by responding to the prompt at the
end of the terminal display.

DISCAL is called by the routines ALNRTN and DATRUN to provide the
operator with a summary on the console terminal of the sodium counts in a
profile. DISCAL also provides a printout of the sodium count summary for
the routine DATRUN. The flag RUNFLG is used by DISCAL to determine whether
ALNRTN or DATRUN is requesting its services. DISCAL expects the profile
data to be recorded in the array LDATA and values for the following range
parameters must be determined (by the calling routine) as they are necessary
for the sodium count calculations:

B30 – 30-km bin pointer. (The number of the range bin that corresponds
to an altitude of 30 km.)

B60 – 60-km bin pointer.

S20 – the number of range bins required for a collection range of 20
km.

DISCAL also uses the experiment parameters accessed through the subroutine
EXPPAR. Below are descriptions of the quantities of the sodium counts
displayed by DISCAL:

Detected Photons – the sum of the counts in the bins for the designated
ranges (at 30 km, 60-80 km, 80-100 km, and 100-120 km).

Total Signal Photons – the detected counts in the 80-100 range minus
the counts in the 60-80 km range.

Signal Photons Per Shot - the Total Signal Photons divided by the laser
shots per profile.

Column Abundance Ratio - the Total Signal Photons divided by the
difference of the counts in the 30-km bin and the average counts
in a bin in the 60-80 km range.

RCVER and SENDER handle the LSI-11 side of the Apple-LSI-11 serial
communications and are discussed in detail in Section 5.3. The RCVER file
also contains the subroutines INITR and MOUT. When called, INITR
initializes the program counter and status word vectors for the serial port
used for the Apple-LSI-11 communications. Also initialized is the input
byte vector STATE (see Section 5.3.5). The subroutine MOUT, with the use of
the file HDUMP, displays on the terminal in hexadecimal form the contents of
the data array LDATA.

The DISPLY file contains a set of subroutines which are called in order
to enable display options available on the console terminal. The subrou-
tines include: CLRSCN - clear screen, FORSCN - inverse or "highlighted"
video, BAKSCN - normal video, and BELSCN - ring the bell. The DISPLY file
listing presented in Appendix III.12 shows display subroutines designed for
use with the DEC VT100 family of terminals. However, the display
subroutines can be used with other types of terminals (for example, the
Hazeltine 1500) by changing the terminal command codes transmitted by the
subroutines to those command codes recognized by the particular terminal.

HDUMP displays a section of memory on the console terminal in hexa-
decimal form. HDUMP must be given the starting memory address and the byte
count of the section to be dumped.

5.3 Data Collection Communication Routines

5.3.1 Introduction. During a data collection run the Apple and the

LSI-11 communicate with each other over an RS-232 standard asynchronous serial link at 9600 baud. Sequential streams of bytes, often referred to as "frames" or "packets," are used to encode the commands or data that are sent between computers. The particular Apple and LSI-11 frames are discussed in Section 5.3.3 on Protocol and Sections 5.3.4 and 5.3.5 on the sending and receiving program structures.

The frame sending sections of the Apple and LSI-11 data collection programs are essentially a set of subroutines which send the desired messages. The receiving sections of the collection programs are interrupt routines which receive and interpret the frames. Each byte received causes an interrupt which gives program control to the receiving section of code and the incoming byte is checked if it is a marker or saved if it is data. Once a complete frame is received, the receiving section of code must interpret the frame and carry out the appropriate action. This interrupt structure was developed so the computers could run background programs during "free" periods of time in the collection process. An example of a background program might be a laser tuning algorithm on the Apple which runs while the LSI-11 is busy storing and doing a printout on the current data profile.

The serial communications link was selected over other types of links (for example, a parallel link) because it was realized that during experiments the Apple and the LSI-11 might be separated by a substantial distance (20-30 feet) and it was felt that the serial link would be less susceptible to noise problems. Many of the ideas used in the design of the communications software are presented in the article "Build an Intercomputer Data Link," by Wingfield (Byte, April 1981).

5.3.2 <u>Serial peripheral cards</u>. Two interface cards are used for
the serial communications. A California Computer Systems (CCS) model 7710A
asynchronous serial interface card is used in the Apple and a DEC DLV11-J
four port asynchronous interface is used in the LSI-11.

The CCS card is designed to be used in any of the Apple peripheral
slots 1 through 7. The card has one channel and uses the Apple I/O
addresses as shown in Table 5.1. A DB 25-pin female connector is provided
on the card for serial input and output. Details on the use of the card are
available in the <u>CCS Model 7710A Owner's Manual</u>. The preprocessing soft-
ware presented in this report was written for the CCS card operating in
Apple slot 2.

The DEC DLV11-J interface card mounts into the LSI-11 computer back-
plane. The DLV11-J has four channels with the register and vector addresses
shown in Table 5.2. Each channel has a 10(2x5) pin connector for serial I/O
purposes. The DLV11-J is described in detail in the <u>DEC Microcomputer</u>
<u>Interfaces Handbook</u> [1980]. In the lidar system the DLV11-J is not only
used for the Apple-LSI-11 communications but also for the console terminal
and the line printer input and output. The channel assignments for the
lidar system are:

Channel 0 = Apple-LSI-11 communications

Channel 1 = open

Channel 2 = line printer

Channel 3 = console terminal

Figure 5.3 shows the connections for the cable between the Apple CCS card
and the LSI-11 DLV11-J.

5.3.3 <u>Communication protocol</u>. Computer protocols are the forms
that are established as appropriate and acceptable in communication between

TABLE 5.1 APPLE CCS SERIAL CARD I/O LOCATIONS

ADDRESS (HEX)	REGISTER
C0x0 (Write)	COMMAND
C0x0 (Read)	STATUS
C0x1 (Write)	TRANSMIT DATA
C0x1 (Read)	RECEIVE DATA

TABLE 5.2 LSI-11 DLV11-J SERIAL INTERFACE I/O LOCATIONS

ADDRESS (OCTAL)	REGISTER	VECTOR ADDRESS	CHANNEL
176500	RCSR		0
176502	RBUF	300	
176504	XCSR		
176506	XBUF	304	
176510	RCSR		1
176512	RBUF	310	
176514	XCSR		
176516	XBUF	314	
176520	RCSR		2
176522	RBUF	320	
176524	XCSR		
176526	XBUF	324	
177560	RCSR		3
177562	RBUF	60	(Console
177564	XCSR		Device)
177566	XBUF	64	
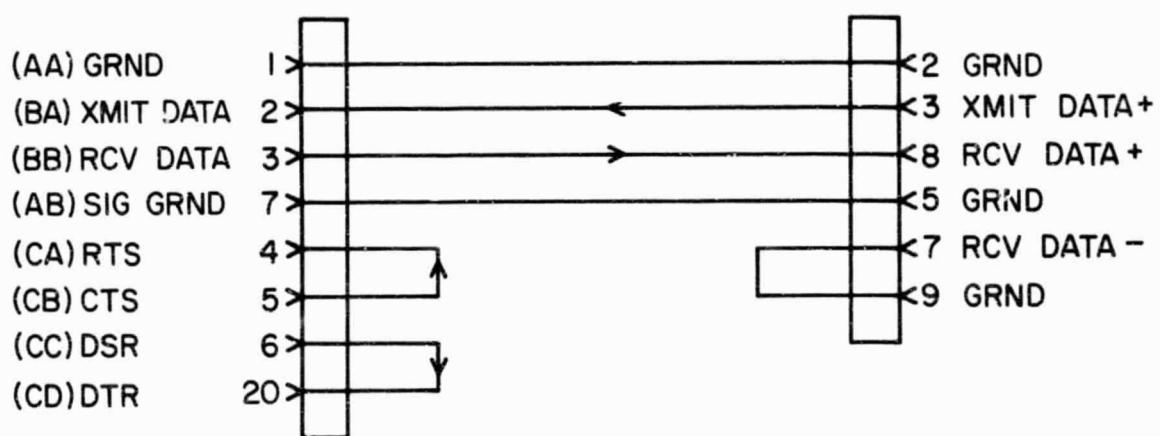
Figure 5.3 Apple CCS card and LSI-11 DLV11-J cable connections.

computers. This section describes the protocol developed for the Apple-LSI-11 serial communication link.

The commands or data sent between the Apple and the LSI-11 are encoded into sequential streams of bytes called "frames" or "packets." The frame format for this application has two parts: the header and the information section. The header part has a beginning-of-frame marker, an operation code or opcode (which determines the use of the frame), an end-of-header marker, and a checksum. The information section has a beginning-of-section marker, the data, an end-of-frame marker, and a checksum. The checksums provide more reliable communications in noisy environments. Figure 5.4 shows the general frame format.

The frame markers consist of two bytes. The first byte is the data-link escape (DLE) which alerts the receiver that the next byte should denote the start of frame (STX), the end of frame (ETX), or the start-of-frame information section (CTX). The values of the marker bytes in hexadecimal are: DLE = $90, STX = $83, ETX = $82, and CTX = $81. To avoid incorrect interpretations of DLE ETX pairs that occur as data, any DLE byte in a data frame is transmitted twice. Upon seeing the DLE DLE pair in a data frame, the receiver saves one of the DLEs and discards the other.

The opcode, following the DLE STX pair in the frame header, identifies the type of frame being transmitted. Five types of frames are used between the Apple and the LSI-11: a commands frame, a data run frame, a data frame, a profile done frame, and an acknowledge frame. The commands and data run frames are only sent from the LSI-11 to the Apple. The commands frame (opcode COM = hexadecimal 32) contains information on the desired laser repetition rate and the number of shots per profile. The data run frame (DRUN = hexadecimal 31) signals the Apple to start collecting a data pro-

| DLE | STX | (OPC) | DLE | ETX | (CHECKSUM LOW) | (CHECKSUM HIGH) | DLE | CTX | | DLE | ETX | (CHECKSUM LOW) | (CHECKSUM HIGH) |

Data Bytes

Header ─── Information Section (Part 2)
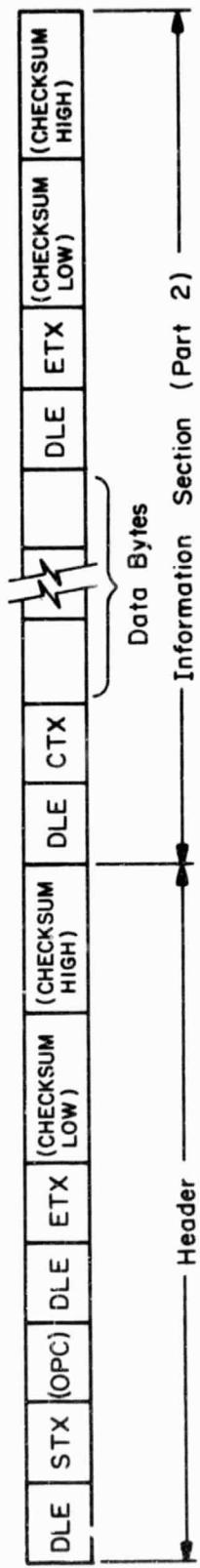
Figure 5.4 General communication frame format.

file. The data and profile done frames are sent one way from the Apple to the LSI-11. The data frames contain the profiles collected by the preprocessing system and the profile done frames (PRDONE = hexadecimal 33) signals the LSI-11 that the profile collection and transmission are completed.

The data frame is unique in that it is of variable length with its maximum length set by a constant (PAKLEN) in the Apple and LSI-11 software. The data frame opcode is also unique in that it is not a constant. This is because of the following two qualities: 1) Due to the method of storing the profile in the Apple RAM the low and high order bytes of the 16-bit profile words are sent in separate data frames; and 2) because of the long length of the high or low byte data frames (2000 bytes each), the data frame was designed to be able to transfer the data with multiple frames as opposed to one frame. The data frame encodes the two qualities given above by using the first hexadecimal digit of the opcode to differentiate the high and low byte frames and the second digit to number the frames (low byte frame = hexadecimal 50 - 5F, high byte frame = hexadecimal E0 - EF). The data frame opcode allows the LSI-11 receiving code to assemble and order the 16-bit profile words correctly upon receipt of the data frames.

After transmitting a command, data run, data, or profile done frame, the sender waits in a timeout loop for the receiver to respond with an acknowledge frame (ACK = hexadecimal 34). If an error is detected in the transmitted frame (for example, a checksum error) no acknowledge is returned, the sender timeout loop expires, and the frame is retransmitted. The sending software attempts to retransmit the frame a number of times (set by a constant, normally 2) before displaying a transmission error message on the terminal or monitor.

The five types of frames are illustrated in Figure 5.5. Only the commands frame and the data frame use the frame information section.

The checksum is the 16-bit summation of all the bytes in the frame (or frame section) except for the first DLE and the final ETX bytes. The header checksum is used to verify the frame opcode and the information checksum is used to detect transmission errors in the data. The header checksum in the commands and data frames is retained to verify that the opcode of the frame is correct before overwriting the old data values in memory with the new data values.

5.3.4 <u>Send routine structure</u>. The sending routines are a set of subroutines in the Apple and LSI-11 data collection software that are responsible for sending the communications frames to the receiving routine of the other computer. The Apple sending subroutines are contained in the Apple Sender program section and are labeled: SNDDAT - send data frame, SNDPRD - send profile done frame, and SNDACK - send acknowledge frame. The LSI-11 sending subroutines are contained in the Macro-11 file SENDER.MAC and are labeled: SNDCOM - send commands frame, SNDDTR - send data run frame, and SNDACK - send acknowledge frame.

Apple subroutines SNDDAT and SNDPRD along with LSI-11 subroutines SNDCOM and SNDDTR not only output the frame bytes to the communications serial port, but also wait after transmission in a timeout loop for a returning acknowledge frame. If the timeout loop is completed before the acknowledge is received, the frame is retransmitted. An error index is decremented each time a retransmission is necessary, and an error message is displayed on the sender's console after two retransmissions fail (the total number of transmission attempts is set at 3 by the constant ERRCNT, although ERRCNT can easily be changed). After a frame transmission failure

| DLE | STX | COM | DLE | ETX | (CHECKSUM LOW) | (CHECKSUM HIGH) | DLE | CTX | REPNUM | SHOTS/PRF LOW BYTE | SHOTS/PRF HIGH BYTE | (CHECKSUM LOW) | (CHECKSUM HIGH) |

a) Commands frame

| DLE | STX | DRUN | DLE | ETX | (CHECKSUM LOW) | (CHECKSUM HIGH) |

b) Data run frame

| DLE | STX | (DATA OPC) | DLE | ETX | (CHECKSUM LOW) | (CHECKSUM HIGH) | DLE | CTX | | (CHECKSUM LOW) | (CHECKSUM HIGH) |

Data Bytes

c) Data frame

| DLE | STX | PR-DONE | DLE | ETX | (CHECKSUM LOW) | (CHECKSUM HIGH) |

d) Profile done frame

| DLE | STX | ACK | DLE | ETX | (CHECKSUM LOW) | (CHECKSUM HIGH) |

e) Acknowledge frame

Figure 5.5 Formats for different types of communication frames.

(including the retransmission attempts), the operator has a choice of attempting the frame transmission again or exiting the sending routine. The choice is prompted by the question "Try again?" which is displayed on the sending computer console after the transmission error message.

The flowchart of the LSI-11 subroutine SNDCOM in Figure 5.6 illustrates the general structure of all the sending subroutines. First, each routine loads its respective opcode into the transmit opcode variable (OPCDE). Next, the retransmission error count (ERRORS) and the acknowledge flag (ACKFLG) are initialized. Finally, the frame bytes are sent to the communications serial port and the routine begins to execute the timeout wait loop. If an acknowledge frame is returned to the sending computer, during the wait loop, the receiving routine sets ACKFLG which drops the sending routine out of the timeout loop.

Although the general program structure of each frame sending routine is similar, there are some structure differences due to the nature of the frames. For example, the SNDACK subroutines do not incorporate the timeout loop as they have no use for it. Also, the final task of the SNDCOM subroutine is to call SNDDTR, as a commands frame will always be followed by a data run frame.

Because of the multiple low byte and high byte data frames discussed in Section 5.3.3 on Protocol, the Apple subroutine SNDDAT has some complications added to the general sending subroutine structure. A flowchart for SNDDAT is shown in Figure 5.7. SNDDAT actually uses the subroutine DATOUT to output the data frames. DATOUT is called by SNDDAT, once to send the low byte data frames and once to send the high byte data frames. Three variables must be loaded before calling DATOUT: the base data frame opcode (hexadecimal 50 or E0) is loaded into DATOPC, ADDRES holds the starting
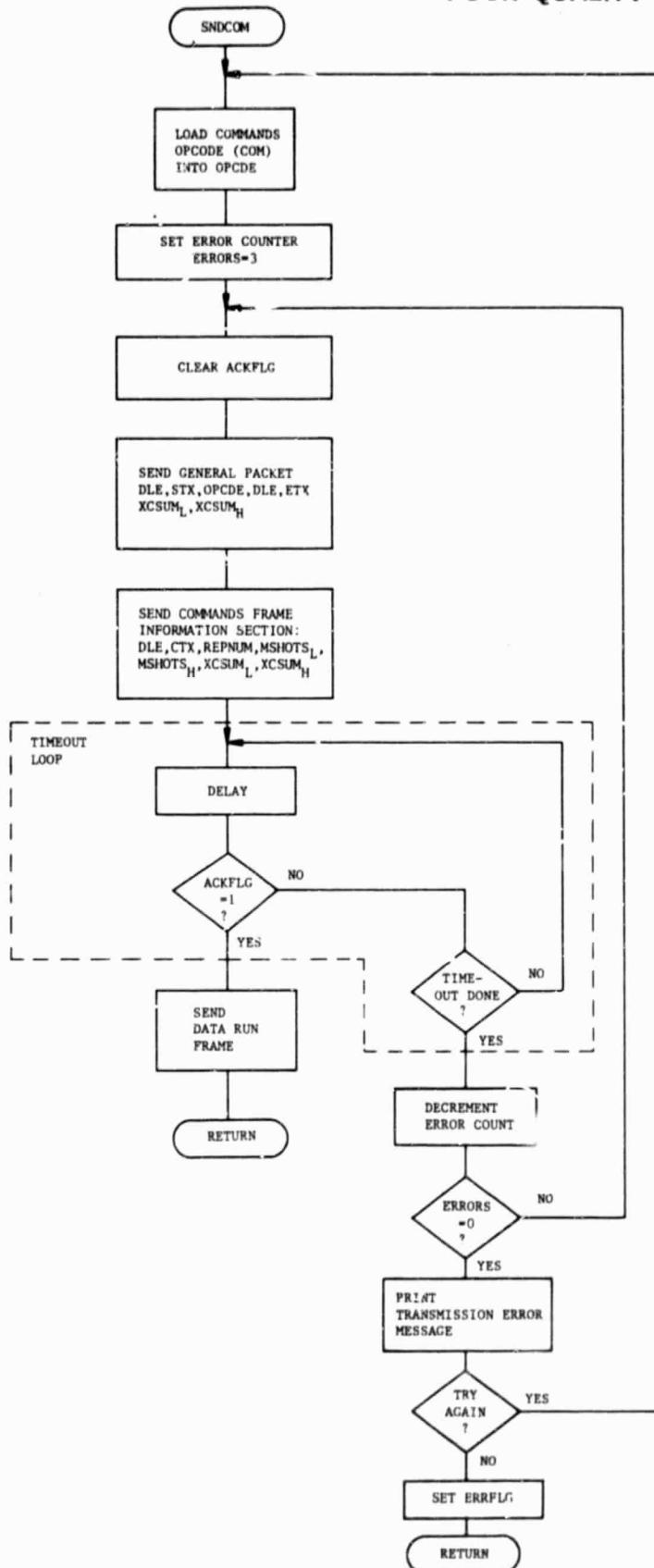
Figure 5.6 Flowchart of LSI-11 subroutine SNDCOM.
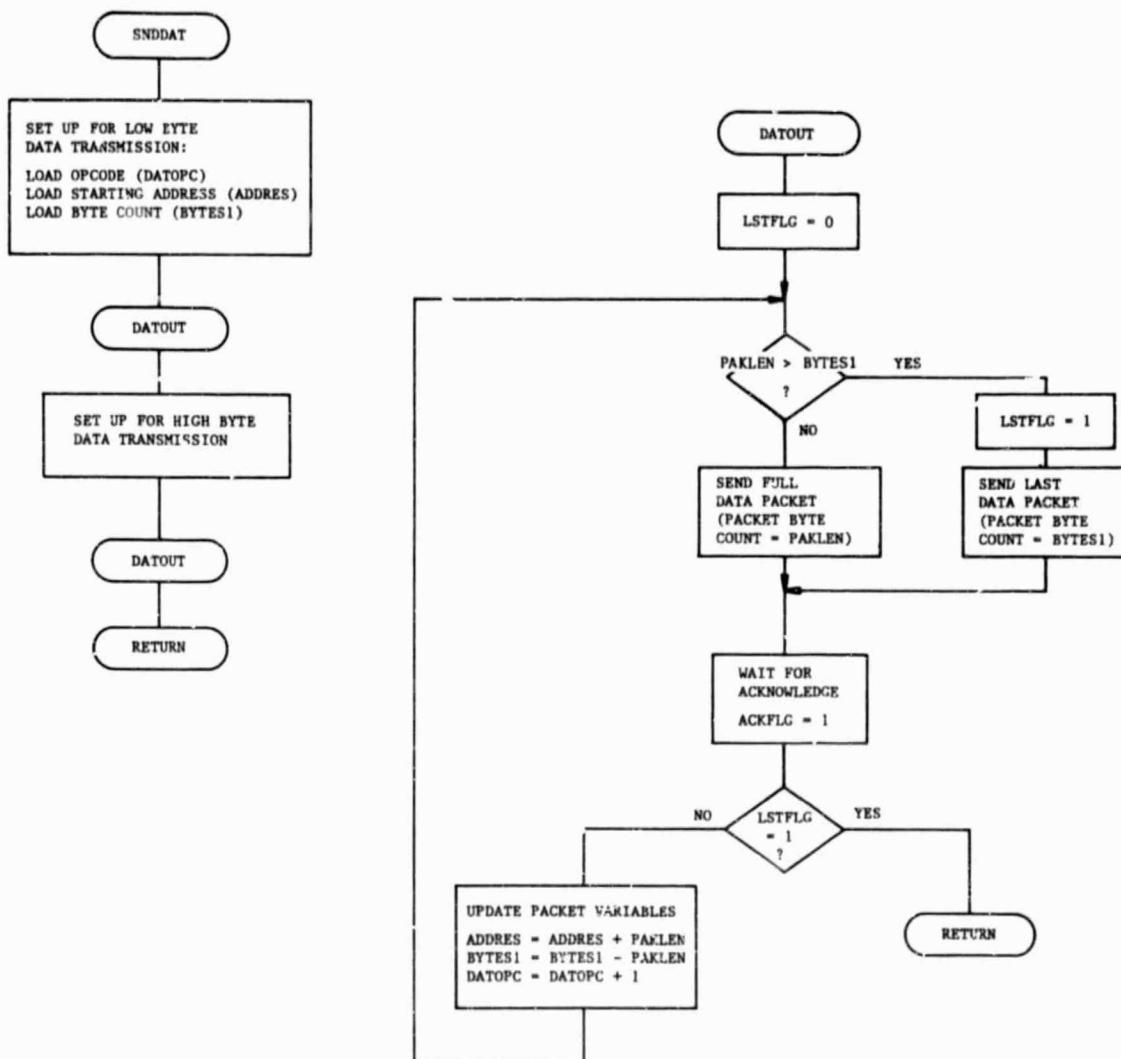
Figure 5.7 Flowchart of Apple subroutine SNDDAT.

location of the block of bytes to be transferred, and BYTES1 contains the transfer byte count. DATOUT uses the variable values to break the blocks of data into multiple data frames. The constant PAKLEN sets the maximum number of data bytes in one frame. The DATOUT routine was designed so that changing the maximum packet length is simply a matter of changing PAKLEN and recompiling the Apple program. Section 5.4 on Data Collection Software Options contains a more detailed discussion on changing the maximum data frame length.

In the data frames, data bytes that happen to have the same value as the DLE marker byte are doubled to avoid a false end-of-frame marker. Upon receipt, the LSI-11 receiving routine saves one of the DLEs as data and discards the other.

5.3.5 Receive routine structure. Figures 5.8 and 5.9 show the receiving routine flowcharts for the Apple and LSI-11, respectively. These routines receive and decode the frames sent by the sending routine of the other computer. Both Apple and LSI-11 routines are interrupt service routines with differences between the routines arising from the receiving of different types of frames. The Apple routine is contained in the Apple Receiver program section and the LSI routine is contained in the Macro-11 file RCVER.MAC.

Each incoming byte on the communication serial port (of either computer) causes an interrupt which results in the program control jumping to the receiving routine. The incoming byte essentially enters the receiving routine at one of the GET BYTE blocks shown in the flowcharts. (The GET BYTE blocks actually represent the subroutine GTBYTE in the receiving code. When called, GTBYTE saves the return address in the variable STATE and executes a return from interrupt. The interrupt from the next incoming byte
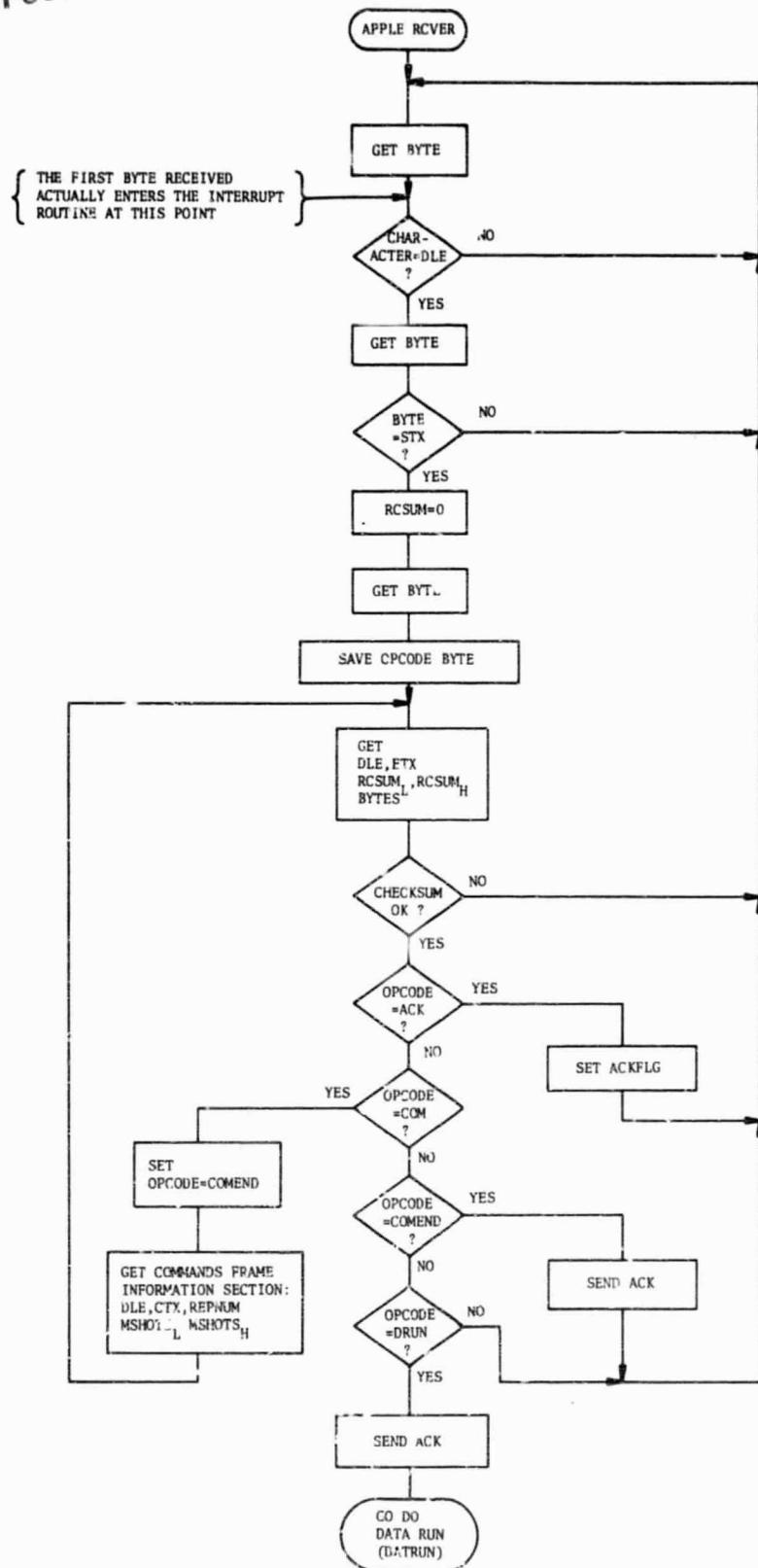
Figure 5.8 Flowchart of Apple receiving routine.
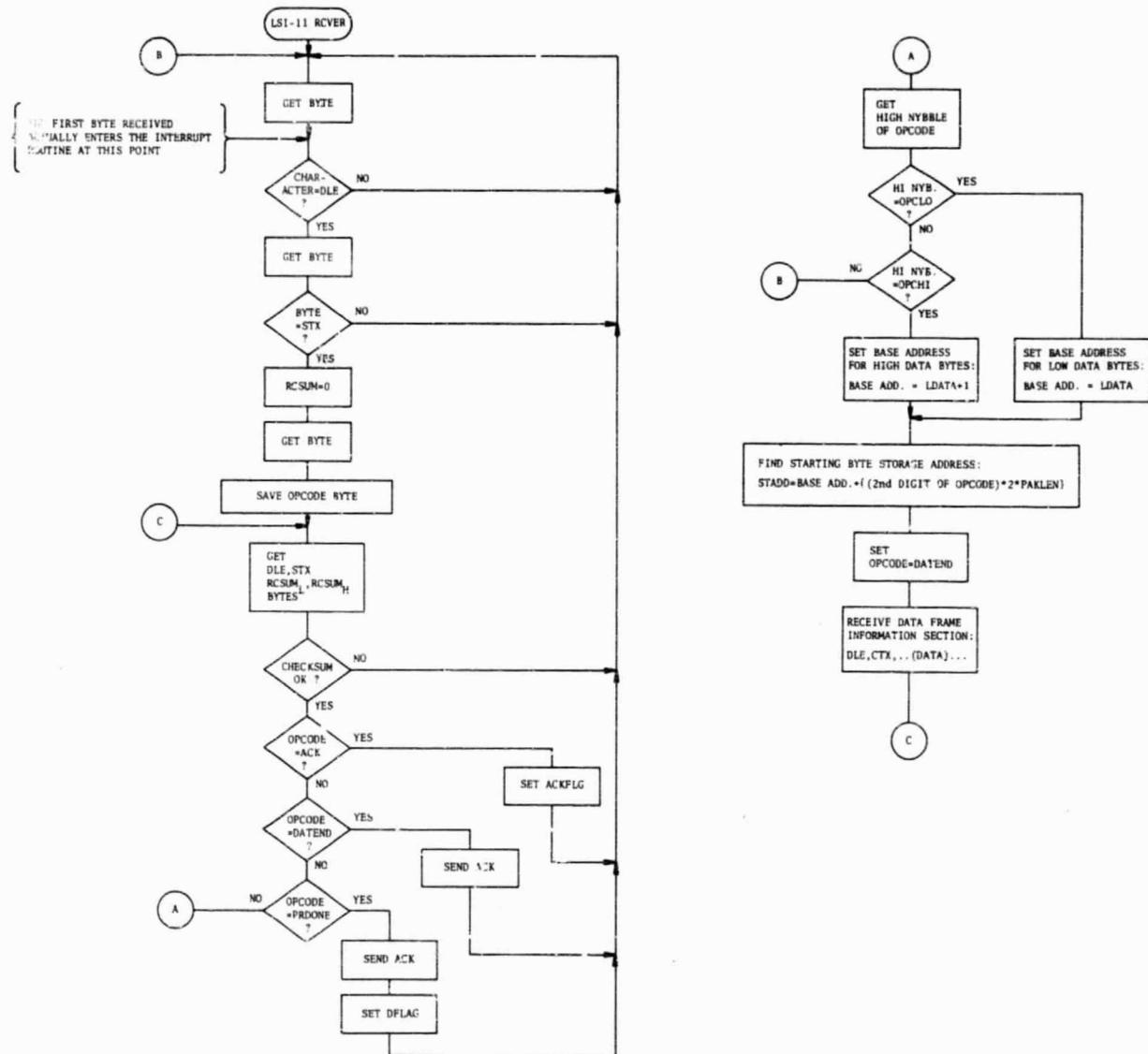
Figure 5.9   Flowchart of LSI-11 receiving routine.

results in the reentering of the receiving routine at the address stored in STATE and the incoming byte is processed by the correct portion of the receiving code. However, it is conceptually easier to think of the GET BYTE blocks as simply a wait followed by an incoming byte.)

The initial portion of the receiving code checks for the DLE STX beginning-of-frame marker. If the bytes are not DLE STX, control returns to the beginning of the receiving code.

After receiving the DLE STX pair, the next incoming byte is assumed to be the frame opcode and is saved for interpretation after the header checksum is received.

Next, the DLE ETX pair and the two bytes of the header checksum are received. If the received checksum equals the calculated checksum, the routine interprets the frame opcode. If not, control returns to the beginning of the receiving code.

In both the Apple and LSI-11 routines an ACK opcode causes the acknowledge flag (ACKFLG) to be set. In the Apple routine the COM opcode implies that the commands frame information section will be transmitted next. The DRUN opcode implies a data run is being requested.

In the LSI-11 routine the PRDONE opcode indicates a data profile has been transmitted successfully and the data flag DFLAG is set. The data frame opcode is the last to be checked for by the LSI-11 routine. The first hexadecimal digit of the data frame opcode designates the frame as high or low byte data, and the second hexadecimal digit is used to place the frame in the correct section of the array LDATA. The low order byte data are stored in the even byte address locations within LDATA and the high order byte data are stored in the odd byte address locations. This procedure results in the separately transmitted high and low bytes of the data profile

being paired as 16-bit words in the array LDATA.

The opcodes COMEND in the Apple receiving code and DATEND in the LSI-11 code are dummy opcodes which are loaded into the opcode variable while the information section of a commands frame or a data frame is being received. The dummy opcodes are essentially flags which allow the checksum testing code (before the opcode identification) to be reused for the end of the information frame section.

After a frame is successfully received and the appropriate action is taken, control returns to the beginning of the receiving code so that the next incoming frame may be interpreted correctly.

5.4 Data Collection Software Options

This section describes the real time interactive options available in the collection software that are not obvious to an inexperienced operator. Also mentioned in this section are some important program changes which allow the collection software to be adapted for use in various experiment situations.

For the most part, all of the real time interaction between the main computer/preprocessing system and the lidar operator is accomplished through the LSI-11 console terminal. Occasionally, the Apple keyboard must be accessed to allow the preprocessing system to continue after a laser triggering error. With the collection software running, the LSI-11 console terminal should initially display the main collection menu (see Section 5.2.2). Selecting one of the menu options results in a jump to a corresponding subroutine, and in general, each option subroutine halts at some point to prompt the operator for an input of some type. At this point, in menu options 1 through 5, typing a carriage return will result in a return to the main menu. Menu option 6, the profile data examine routine (EXMPRF),

initially asks for a set a. d profile number. Inputting the value zero (0) for either number will result in a return to the main menu. Menu option 7, the hexadecimal dump of the data array LDATA, supplies no prompts and the array values are simply displayed on the terminal. After the display is completed a return to the main menu is automatically executed.

The three data collection options in the main menu (options 3, 4, and 5; routines LAALN, ALRTN, and DATRUN) all execute a waiting loop while the preprocessing system is collecting data. Normally, the loop is exited when the profile is successfully received by the LSI-11. However, the operator can force the routine to exit the waiting loop at any time by typing "s" and a carriage return. This option allows routine execution to be returned to the main collection menu when problems occur with the laser (or any other system hardware) while a data profile is being collected. This option is created through the use of the Fortran function ITTINR.

The preprocessing system requires the use of two peripheral cards in the Apple computer peripheral slots: A CCS serial communications card and the preprocessor DMA card. The Apple collection software presented in Appendix III.1 requires the CCS card to be mounted in slot 2 and the DMA card to be mounted in slot 4. However, these slot assignments can be changed by first altering the values of the constants CSR, DMAREG, and SLPIO in the program and then recompiling the program. CSR is the location of the CCS card status register, DMAREG is the beginning location for the DMA card programmable registers, and SLPIO is the beginning location for the DMA card Control Signals (see Appendix III.1).

Two floppy disk drives are required by the LSI-11 collection software to collect data in an efficient manner. One drive contains the system floppy disk and the other drive contains the data floppy disk. The system

disk is generally assigned as the drive on which the computer is booted. The data drive is assigned in the main program file FZZ.FOR in the first four locations of the array variable FNAME. Presently the data disk is assigned as DY0:.

As mentioned throughout Section 5.3 on Data Collection Communication Routines, the data frames sent from the Apple to the LSI-11 are of variable length with the maximum data byte count set by the constant PAKLEN. Changing the maximum data byte count is done by first setting the PAKLEN constant in both the Apple Sender program section and the LSI-11 RCVER section to the desired value and then recompiling the programs. Because of the error checking feature (checksum) of the frame communications technique, in a noisy environment many short data frames may provide more efficient transmission than a fewer number of long data frames. For this reason, a simple method of changing the maximum data byte count in a data frame was provided.

5.5 Data Conversion Software

During data collection, the collection software permanently stores sets of profiles on a floppy disk in unformatted direct access files. The data values in these files are represented in a binary format. After the collection is completed the data are transferred to the University of Illinois CYBER computer for further processing. However, the binary data files must be converted to files with an ASCII format before the transfer can occur. This is due to hardware restrictions in the LSI-11-CYBER link used to transfer the data. The routine CONVRT.SAV is used to accomplish the binary-to-ASCII file conversion.

CONVRT initially requests the name of the device containing the binary files and the name of the device on which the converted files will be

stored. Typical responses might be DY0: and DM1:. Next, a set number is requested and also the total number of profiles in the set must be entered. CONVRT also requests the input of four values that are not recorded in the binary data files. These values are the ground speed (kts), altitude (ft), latitude (deg.-min.), and longitude (deg.-min.) at the time the profile was collected. These are values that were recorded manually during the airborne experiment discussed in Chapter 6; and CONVRT inserts them into the ASCII data files.

The binary-to-ASCII conversion is accomplished with the use of the Fortran functions ENCODE and DECODE. The binary files designated by the name SETxxx.DAT are converted and saved in new files designated by the name SETxxx.ASC, where xxx is the set number. The listing of CONVRT in Appendix III.14 shows the format of the ASCII data files.

6. PRELIMINARY RESULTS OF AN AIRBORNE EXPERIMENT

6.1 Introduction

In March 1983, the University of Illinois Aeronomy Laboratory (Lidar Group) conducted an airborne sodium lidar experiment aboard a National Aeronautics and Space Administration (NASA) aircraft. The primary goal of the experiment was to determine the feasibility of obtaining mesospheric sodium density measurements with an airborne lidar system. In reference to this report, the initial testing of the new sodium lidar preprocessing system with the complete lidar system was conducted during this airborne experiment.

Initial sodium lidar measurements by the University of Illinois Lidar Group were made from ground-based stations [Richter and Sechrist, 1978; Shelton and Gardner, 1981]. Generally, these experiments provided information on the temporal structure of the mesospheric sodium layer as the layer drifted above the test site. In addition, the use of a steerable receiving telescope at NASA Goddard Space Flight Center provided limited observations of the horizontal structure of the layer. An airborne lidar system holds two distinct advantages over these ground-based systems. First, attenuation of the transmitted laser pulse due to low-altitude cloud cover and haze can be alleviated by flying above the obstructions. Second, the aircraft allows observations to be recorded over a long horizontal base line, an essential requirement for good horizontal structure measurements of the layer. This particular airborne experiment was designed as a forerunner to more extensive airborne sodium lidar observations.

6.2 Airborne Sodium Lidar System

The sodium lidar system was flown aboard the NASA Lockheed Electra

L-188 turboprop aircraft (Figure 6.1). The aircraft is stationed at NASA
Wallops Flight Center, Wallops Island, Virginia. The University of Illinois
lidar equipment was used in conjunction with the NASA Electra Lidar
Facility. This Lidar Facility consists of three standard optical table
tops and a 16-inch, f/2.5 primary, receiving telescope all mounted on a
table support structure.

The telescope is mounted near the center of the support table in an
upward pointing position. Two of the optical table tops are positioned on
top of the support table on either side of the telescope. The third optical
table top is mounted in the support structure underneath the larger upper
table top. The support structure is bolted to the floor of the aircraft
with the telescope positioned beneath a quartz window in the roof of the
plane's cabin.

The dye laser and transmitting optics for the lidar system were
fastened to the larger upper table top of the Lidar Facility. The laser
beam was steered through the quartz window, located directly over the tele-
scope, with dielectric-coated optics. The receiving components, including
the preprocessing system, were fastened to the smaller upper table top of
the Lidar Facility. Power supplies and laser tuning equipment were mounted
on the lower table top. Figure 6.2 shows the equipment configuration on
the receiving side of the Lidar Facility. Figure 6.3 is a photograph of
the equipment from the transmitting side of the Facility. The laser high-
voltage power supply and chiller unit in Figure 6.3 were bolted to the floor
of the aircraft. Also shown in Figure 6.3 are the nitrogen gas bottle,
which provided nitrogen for purging the laser dye of oxygen, and the line
printer used by the LSI-11 main computer.
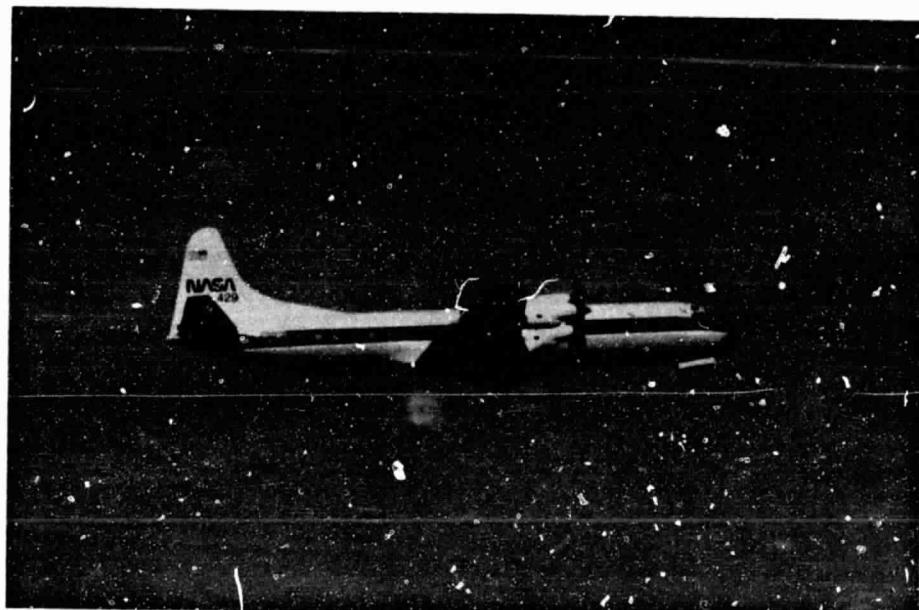
Figure 6.1 Photograph of the NASA Electra aircraft.
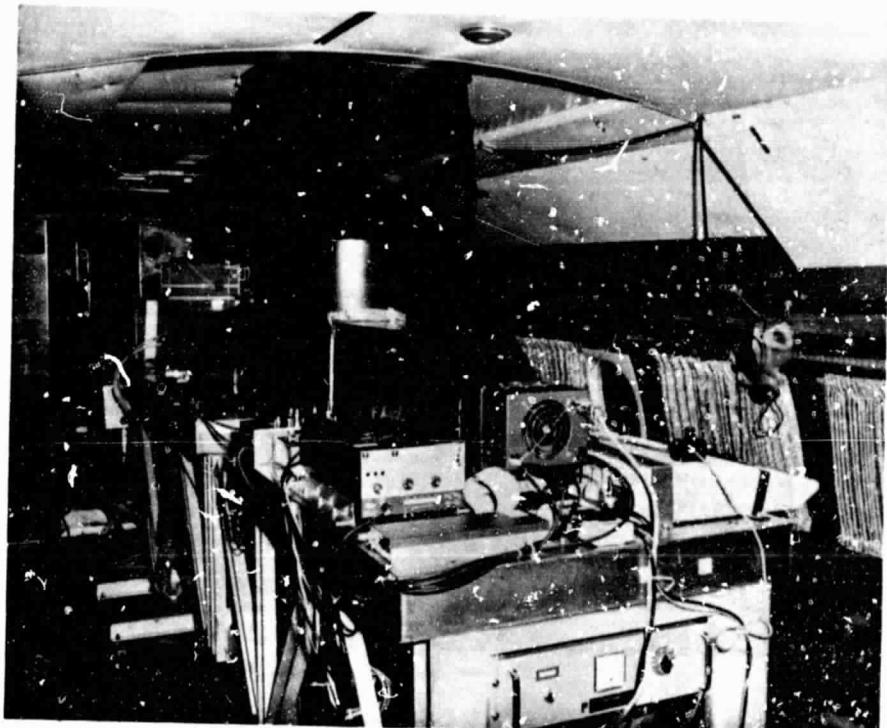
Figure 6.2 Photograph of the lidar receiving system equipment aboard the
NASA Electra. On top of the table in the foreground from left
to right are the amplifier-discriminator, PMT housing, and the
preprocessor Apple microcomputer.
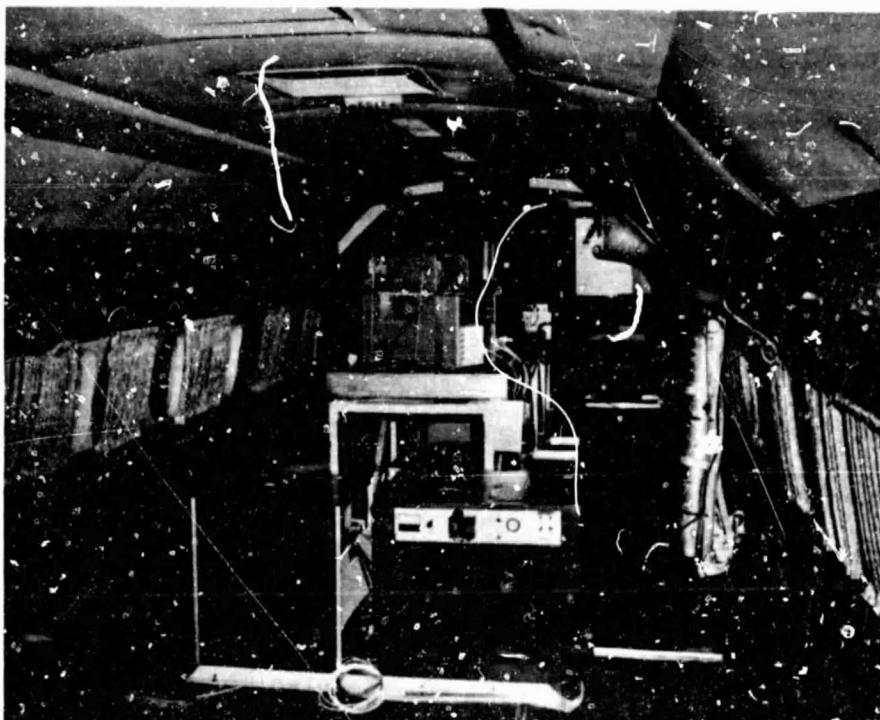
Figure 6.3 Photograph of the lidar transmitting system equipment aboard the NASA Electra. In the foreground on the floor are the laser high-voltage power supply and chiller unit. On the table are the laser tuning monitor oscilloscopes. The laser head is located on the table behind the oscilloscopes.

Figure 6.4 shows the computer rack. This rack was positioned across the aisle from the line printer. Mounted in the rack were the LSI-11 computer, disk drive, terminal, monitor, and the preprocessing system Apple monitor.

6.3 Results

The airborne lidar test was conducted on the night of March 30, 1983 beginning at approximately 2130 EST. The flight originated at Wallops Flight Center (38 °N, 74 °W), reached a turn-around point near Albany, NY (42 °N, 73 °W), and terminated at Wallops at approximately 2330 EST. Although data collection began during the ascent, scattering of the laser beam by high cirrus clouds prevented collection of good sodium data until the aircraft reached an altitude above 25,000 feet. A similar collection problem occurred during the descent. Also, occasional cirrus above 30,000 feet attenuated the received sodium signal photons at various points in the experiment.

Plots of the spatial variations in estimates of the sodium density are shown in Figure 6.5. Figure 6.5(a) shows sodium density profiles collected on the outbound leg of the flight and Figure 6.5(b) shows profiles collected on the return leg. Each profile required 2,000 laser shots. The peak sodium concentration is seen to be located above 90 km which is consistent with previous measurements obtained during the month of March with the ground-based systems. The distribution of the sodium density around the peak is also consistent with previous ground-based observations. The profiles seem to have some related structural features such as the descending local peaks beneath the principal peaks in the three profiles on the right side of Figure 6.5(b). Another interesting feature, most notable in Figure 6.5(a), is a possible secondary layer appearing above 100 km.
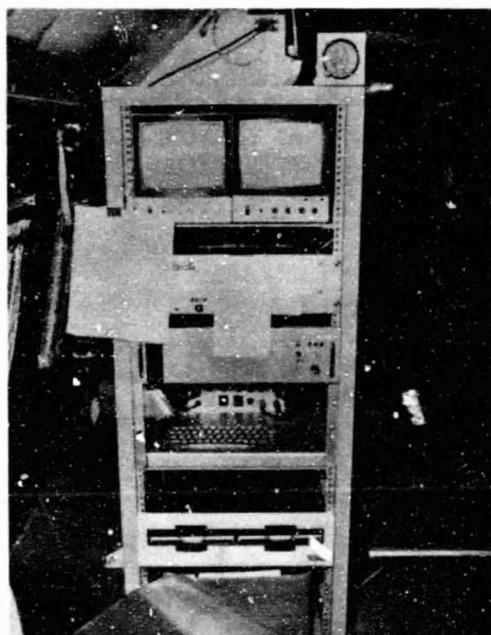
Figure 6.4   Photograph of the computer rack aboard the NASA Electra.
Mounted in the rack from top to bottom are the LSI-11 and
Apple monitors, the LSI-11 power supply and computer back-
plane, the LSI-11 keyboard, and the LSI-11 floppy disk
drives.

C-2

MARCH 30, 1983
TRAVELING NORTH
(TIME INCREASING TO THE RIGHT)

(a)

ALTITUDE (km)

120

100

80

TIME (EST)	-	22:06:43	22:15:26	22:19:55	22:24:12	22:29:00
LATITUDE (°N)	-	40.11	40.57	41.20	41.40	42.04
LONGITUDE (°W)	-	73.57	73.35	73.32	73.38	73.44

MARCH 30, 1983
TRAVELING SOUTH
(TIME INCREASING TO THE LEFT)

(b)

ALTITUDE (km)

120

100

80

TIME (EST)	-	22:59:20	22:55:18	22:51:05	22:47:00	22:42:30	22:33:40
LATITUDE (°N)	-	39.48	40.05	40.24	40.41	41.02	41.45
LONGITUDE (°W)	-	74.22	74.12	74.01	73.52	73.46	73.45

Figure 6.5 Time and location history of the estimated altitude profiles
of sodium density observed on (a) the outbound leg and (b) the
return leg of the airborne experiment conducted on March 30,
1983. A Hamming window lowpass filter with a cutoff of .35 km^{-1}
was used to spatially filter the profiles.

7. CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

The preliminary results of the airborne sodium lidar experiment presented in Chapter 6 indicate that the preprocessing system is functioning as designed. The sodium layer profiles collected during the experiment contain features that are similar to those typically observed by sodium lidar systems. Noise in the data resulting from cloud cover and other factors makes it difficult to evaluate the performance of the preprocessing system in an absolute manner. However, no obvious contamination of the data directly attributable to the preprocessing system was noticed.

Desirable future additions to the preprocessing and main computer systems would include the capability of further real-time processing of the data during collection. This might involve digitally filtering the data and displaying the filtered profiles on a graphics terminal. This type of immediate feedback would help in maintaining the lidar system at an optimal operation level. For example, drifting of the laser tuning would be clearly noticeable in the displayed profiles. Also, the displayed profiles would indicate the quality of the data and any interesting features of the profiles.

Because of the interrupt structure of the preprocessing system software, both the Apple and LSI-11 computers may be employed to run background routines in addition to the data collection routines. This provides the option of controlling any of the lidar system equipment with a computer. Examples would be computer-directed laser tuning and computer-controlled alignment of the transmitted and received beams.

I.1 Apple Peripheral Card Connector Pinout

PIN	NAME	FUNCTION
1	I/O SELECT	MEMORY MAPPED SELECT LINE
2	A0	⎫
3	A1	
4	A2	
5	A3	
6	A4	
7	A5	
8	A6	
9	A7	⎬ ADDRESS BUS
10	A8	
11	A9	
12	A10	
13	A11	
14	A12	
15	A13	
16	A14	
17	A15	⎭
18	R/$\overline{\text{W}}$	READ/WRITE
19	N.C.	
20	I/O STROBE	MEMORY MAPPED STROBE LINE
21	RDY	6502 'READY' LINE
22	$\overline{\text{DMA}}$	DIRECT MEMORY ACCESS CONTROL LINE
23	INT OUT	INTERRUPT DAISY CHAIN OUTPUT
24	DMA OUT	DMA DAISY CHAIN OUTPUT
25	+5V	+5 VOLTS
26	GND	LOGICAL GROUND
27	DMA IN	DMA DAISY CHAIN INPUT
28	INT IN	INTERRUPT DAISY CHAIN INPUT
29	$\overline{\text{NMI}}$	NON MASKABLE INTERRUPT REQUEST
30	$\overline{\text{IRQ}}$	INTERRUPT REQUEST
31	$\overline{\text{RES}}$	RESET LINE
32	$\overline{\text{INH}}$	ROM INHIBIT LINE
33	-12V	-12 VOLTS
34	-5V	-5 VOLTS
35	N.C.	
36	7M	7 MHZ CLOCK
37	Q3	2 MHZ (ASYMMETRIC) CLOCK
38	Φ1	PHASE 1 CLOCK
39	USER1	INTERNAL I/O ADDRESS DISABLE
40	Φ0	PHASE 0 CLOCK (ALSO Φ2)
41	DEVICE SELECT	MEMORY MAPPED SELECT LINE
42	D7	⎫
43	D6	
44	D5	
45	D4	⎬ DATA BUS
46	D3	
47	D2	
48	D1	
49	D0	⎭
50	+12V	+12 VOLTS

I.2   Apple DMA Card Ribbon Connector Pinout

PIN	NAME	FUNCTION
1	GND	
2		
3		
4		
5		
6		⎬ LOGIC GROUND
7		
8		
9		
10		
11	GND	
12	$\overline{Y3}$	CONTROL LINE $\overline{Y3}$
13	$\overline{Y1}$	CONTROL LINE $\overline{Y1}$
14	$\overline{RES}$	RESET LINE
15	$\overline{TxRQ}$	DMA TRANSFER REQUEST
16	N.C.	
17	N.C.	
18	$\overline{DEND}$	DMA END SIGNAL
19	$\overline{TxSTB}$	DMA TRANSFER STROBE
20	$\overline{TxAKA}$	DMA TRANSFER ACKNOWLEDGE
21	$\overline{DGRNT}$	DMA BUS GRANT
22	$\overline{Y2}$	CONTROL LINE $\overline{Y2}$
23	$\overline{Y4}$	CONTROL LINE $\overline{Y4}$
24	$\overline{Q3}$	INVERTED APPLE Q3 CLOCK
25	$\Phi0$	APPLE $\Phi0$ CLOCK
26	$\Phi1$	APPLE $\Phi1$ CLOCK
27	D0	
28	D1	
29	D2	
30	D3	⎬ DATA LINES
31	D4	
32	D5	
33	D6	
34	D7	

I.3 SLIPP Unit External Ribbon Connector Pinout

PIN	NAME	FUNCTION
1	$\overline{\text{DEND}}$	DMA END SIGNAL
2	$\overline{\text{TxSTB}}$	DMA TRANSFER STROBE
3	$\overline{\text{TxAKA}}$	DMA TRANSFER ACKNOWLEDGE
4	$\overline{\text{DGRNT}}$	DMA BUS GRANT
5	$\overline{\text{Y2}}$	CONTROL LINE $\overline{\text{Y2}}$
6	$\overline{\text{Y4}}$	CONTROL LINE $\overline{\text{Y4}}$
7	Q3	INVERTED APPLE Q3 CLOCK
8	Φ0	APPLE Φ0 CLOCK
9	Φ1	APPLE Φ1 CLOCK
10	D0	
11	D1	
12	D2	
13	D3	DATA LINES
14	D4	
15	D5	
16	D6	
17	D7	
18	GND	
19		
20		
21		
22		
23		LOGIC GROUND
24		
25		
26		
27		
28	GND	
29	$\overline{\text{Y3}}$	CONTROL LINE $\overline{\text{Y3}}$
30	$\overline{\text{Y1}}$	CONTROL LINE $\overline{\text{Y1}}$
31	$\overline{\text{RES}}$	RESET LINE
32	$\overline{\text{TxRQ}}$	DMA TRANSFER REQUEST
33	N.C.	
34	N.C.	

I.4 SLIPP Unit Internal Card Edge Connector Pinout

PIN	NAME	FUNCTION
A	+5V	+5 VOLTS
B	D7	
C	D6	
D	D5	
E	D4	DATA LINES
F	D3	
H	D2	
J	D1	
K	D0	
L	$\Phi 1$	APPLE $\Phi 1$ CLOCK
M	$\Phi 0$	APPLE $\Phi 0$ CLOCK
N	$\overline{Q3}$	INVERTED APPLE Q3 CLOCK
P	$\overline{Y4}$	CONTROL LINE $\overline{Y4}$
R	$\overline{Y2}$	CONTROL LINE $\overline{Y2}$
S	$\overline{DGRNT}$	DMA BUS GRANT
T	$\overline{TxAKA}$	DMA TRANSFER ACKNOWLEDGE
U	$\overline{TxSTB}$	DMA TRANSFER STROBE
V	$\overline{DEND}$	DMA END SIGNAL
W	N.C.	
X	N.C.	
Y	N.C.	
Z	GND	LOGIC GROUND
1	-5V	-5 VOLTS
2	GND	
3		
4		
5		
6		
7		LOGIC GROUND
8		
9		
10		
11		
12	GND	
13	$\overline{Y3}$	CONTROL LINE $\overline{Y3}$
14	$\overline{Y1}$	CONTROL LINE $\overline{Y1}$
15	$\overline{RES}$	RESET LINE
16	$\overline{TxRQ}$	DMA TRANSFER REQUEST
17	N.C.	
18	N.C.	
19	AUX OUT	AUXILIARY OUTPUT
20	TRIG	LASER TRIGGER OUPUT
21	LCP	LASER COMMAND PULSE INPUT
22	NIM	FAST NIM DISCRIMINATOR INPUT

Figure II.1   Apple DMA card component diagram.

ORIGINAL PAGE IS
OF POOR QUALITY
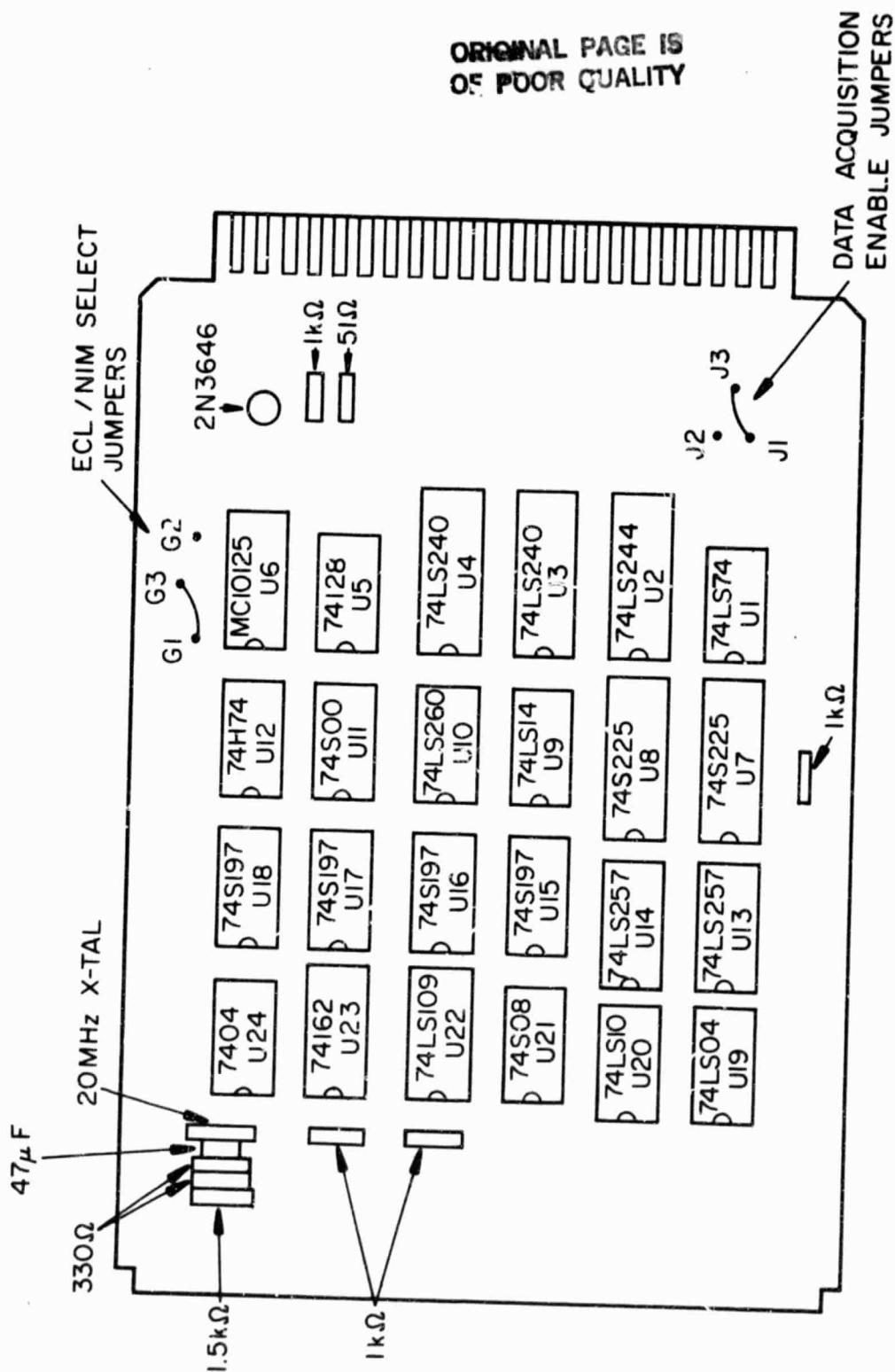
Figure II.2 SLIPP unit circuit component diagram.

APPENDIX III SOFTWARE LISTINGS

III.1 SLPAPP

```
SOURCE FILE: SLPAPP
 ---- NEXT OBJECT FILE NAME IS SLPAPP.OBJ
0C28:         1          ORG   $0C28
0C28:         2 *********************************
0C28:         3 *
0C28:         4 *           APPLE
0C28:         5 *        SLIPP SOFTWARE
0C28:         6 *           D.VOELZ
0C28:         7 *        FEB. 20, 1983
0C28:         8 *
0C28:         9 *********************************
0C28:        10 *
0C28:        11 *  I/O  LOCATIONS
0C28:        12 *
C0A0:        13 CSR     EQU   $C0A0       ;SERIAL CARD IN SLOT 2
C0A1:        14 XRDR    EQU   CSR+1
0C28:        15 *
C400:        16 DMAREG  EQU   $C400       ;DMA BOARD IN SLOT 4
C0C0:        17 SLPIO   EQU   $C0C0
C0C1:        18 TRIG    EQU   SLPIO+1
C0C4:        19 SLPRST  EQU   SLPIO+4
0C28:        20 *
0C28:        21 *  GENERAL PARAMETERS
0C28:        22 *
0003:        23 ERRCNT  EQU   3           ;ERROR COUNT
0032:        24 DEL1    EQU   50          ;DELAY PAR. 1
0031:        25 DRUN    EQU   $31         ;OPCODES -DATA RUN
0032:        26 COM     EQU   $32         ;        -COMMANDS
0033:        27 PRDONE  EQU   $33         ;        -PROFILE DONE
0034:        28 ACK     EQU   $34         ;        -ACKNOWLEDGE
0045:        29 COMEND  EQU   $45         ;        -COMMANDS DONE
0081:        30 CTX     EQU   $81         ; 'CTX' CHAR
0082:        31 ETX     EQU   $82         ; 'ETX' CHAR
0083:        32 STX     EQU   $83         ; 'STX' CHAR
0090:        33 DLE     EQU   $90         ; 'DLE' CHAR
03FE:        34 INTV    EQU   $03FE       ;INTERRUPT VECTOR
FCA8:        35 WAIT    EQU   $FCA8       ;DELAY ROUTINE
FF3F:        36 IOREST  EQU   $FF3F       ;RESTORE REGISTERS
FF4A:        37 IOSAVE  EQU   $FF4A       ;SAVE REGISTERS
0C28:        38 *
0C28:        39 *  DATA PACKET PARAMETERS
0C28:        40 *
0050:        41 OPCLO   EQU   $50         ;LO BYTE DATA OPCODE
00E0:        42 OPCHI   EQU   $E0         ;HI BYTE DATA OPCODE
6000:        43 BUFADD  EQU   $6000       ;INPUT BUFFER MEMORY
9000:        44 STADDL  EQU   $9000       ;LO BYTE MAINFRAME MEMORY
9800:        45 STADDH  EQU   STADDL+$800 ;HI BYTE MAINFRAME MEMORY
```

```
07D0:       46 PAKLEN  EQU   2000    ;PACKET LENGTH (MAX=32767)
07D0:       47 BYTCNT  EQU   2000    ;BYTE COUNT
0C28:       48 *
0C28:       49 *   MONITOR REFERENCES
0C28:       50 *
0024:       51 CH      EQU   $24     ;HORIZONTAL TAB LOCATION
0025:       52 CV      EQU   $25     ;VERTICAL TAB LOCATION
0038:       53 KSW     EQU   $38     ;KEYBOARD SWITCH
008D:       54 CR      EQU   $8D     ;CARRIAGE RETURN
C010:       55 KBDSTB  EQU   $C010   ;KEYBOARD STROBE
FC58:       56 HOME    EQU   $FC58   ;CLEAR SCREEN
FC22:       57 VTAB    EQU   $FC22   ;VERTICAL TAB
FD0C:       58 RDKEY   EQU   $FD0C   ;READ KEY ROUTINE
FD1B:       59 KEYIN   EQU   $FD1B   ;KEY INPUT ROUTINE
FDF0:       60 COUT1   EQU   $FDF0   ;OUTPUT CHARACTER
FE80:       61 SETINV  EQU   $FE80   ;INVERSE SCREEN
FE84:       62 SETNRM  EQU   $FE84   ;NORMAL SCREEN
0C28:       63 *
0C28:       64 *   ZERO PAGE VARIABLES
0C28:       65 *
0000:       66          DSECT
00F9:       67          ORG   $F9
00F9:       68 ADDRES  DS    2       ;TRANSMIT ADDRESS VECTOR
00FB:       69 BYTES1  DS    2       ;TRANSMIT BYTE COUNTER
00FD:       70 PRTADD  DS    2       ;PRINTING ADDRESS VECTOR
00FF:       71 TLBTCT  DS    1       ;TEMPORARY TRANSMIT BYTE COUNTER
0C28:       72          DEND
0C28:       73 *
0C28:       74 *   GENERAL VARIABLES
0C28:       75 *
0000:       76          DSECT
0C00:       77          ORG   $0C00
0C00:       78 REPNUM  DS    1       ;LASER REP NUMBER
0C01:       79 MSHOTS  DS    2       ;SHOTS/PROFILE (MAX=32767)
0C03:       80 *
0C03:       81 ACKFLG  DS    1       ;ACKNOWLEDGE FLAG
0C04:       82 DATOPC  DS    1       ;TRANSMIT OPCODE STORAGE
0C05:       83 ERRORS  DS    1       ;ERROR COUNTER
0C06:       84 LSTFLG  DS    1       ;LAST PAGE TRANSMIT FLAG
0C07:       85 OPCDE   DS    1       ;RECVD OPCODE STORAGE
0C08:       86 *
0C08:       87 COUNT1  DS    2       ;TEMP. TRANSMIT BYTE COUNTER
0C0A:       88 RCSUM   DS    2       ;RECVD CHECKSUM
0C0C:       89 SAVADD  DS    2       ;PLACE TO SAVE ADDRESS
0C0E:       90 SAVSUM  DS    2       ;PLACE TO SAVE CHECKSUM
0C10:       91 STATE   DS    2       ;INPUT BYTE VECTOR
0C12:       92 TSHOT   DS    2       ;SHOT COUNTER
0C14:       93 XCSUM   DS    2       ;TRANSMIT CHECKSUM
0C28:       94          DEND
0C28:       95 *
0C28:       96 ***********************************
0C28:       97 *
0C28:       98 *   MAIN PROGRAM LOOP
0C28:       99 *
0C28:      100 ***********************************
0C28:      101 *
```

```
0C28:             102 *   INITIALIZE RECEIVER
0C28:             103 *
0C28:A9 1B        104          LDA   #>KEYIN    ;LOAD KEYBOARD SWITCH
0C2A:85 38        105          STA   KSW
0C2C:A9 FD        106          LDA   #<KEYIN
0C2E:85 39        107          STA   KSW+1
0C30:             108 *
0C30:2C 10 C0     109          BIT   KBDSTB     ;CLEAR KEYBOARD STROBE
0C33:             110 *
0C33:20 6C 10     111          JSR   HEDPRT     ;PRINT HEADING
0C36:             112 *
0C36:A9 03        113          LDA   #03        ;MASTER RESET ACIA
0C38:8D A0 C0     114          STA   CSR
0C3B:A9 95        115          LDA   #$95       ;SET UP ACIA
0C3D:8D A0 C0     116          STA   CSR
0C40:             117 *
0C40:A9 5E        118          LDA   #>IOINT    ;SET INTERRUPT VECTOR
0C42:8D FE 03     119          STA   INTV       ;  FOR IOINT
0C45:A9 0C        120          LDA   #<IOINT
0C47:8D FF 03     121          STA   INTV+1
0C4A:             122 *
0C4A:A9 85        123          LDA   #>STPAK1   ;SET INPUT BYTE VECTOR
0C4C:8D 10 0C     124          STA   STATE
0C4F:A9 0C        125          LDA   #<STPAK1
0C51:8D 11 0C     126          STA   STATE+1
0C54:             127 *
0C54:58           128          CLI              ;CLEAR INTERRUPT BIT
0C55:20 B6 10     129          JSR   IDLPRT     ;PRINT 'IDLE'
0C58:             130 *
0C58:EA           131 WLOOP    NOP              ;IDLE LOOP
0C59:EA           132          NOP
0C5A:EA           133          NOP
0C5B:4C 58 0C     134          JMP   WLOOP
0C5E:             135 *
0C5E:             136 ***********************************
0C5E:             137 *
0C5E:             138 *   APPLE RECEIVER
0C5E:             139 *
0C5E:             140 ***********************************
0C5E:             141 *
0C5E:             142 *   INTERRUPT ENTRY POINT
0C5E:             143 *
0C5E:78           144 IOINT    SEI              ;DISABLE INTERRUPT
0C5F:20 4A FF     145          JSR   IOSAVE     ;SAVE REGISTERS
0C62:AD A0 C0     146          LDA   CSR        ;CLEAR INTR BIT IN CSR
0C65:AD A1 C0     147          LDA   XRDR       ;LOAD ACCUM WITH INPUT BYTE
0C68:6C 10 0C     148          JMP   (STATE)    ;GO TAKE CARE OF INPUT BYTE
0C6B:             149 *
0C6B:             150 *   INTERRUPT EXIT POINT
0C6B:             151 *
0C6B:18           152 GETBT    CLC              ;UPDATE RECD CHECKSUM
0C6C:6D 0A 0C     153          ADC   RCSUM      ;  LO BYTE
0C6F:90 03        154          BCC   SKUPRC
0C71:EE 0B 0C     155          INC   RCSUM+1    ;  HI BYTE
0C74:8D 0A 0C     156 SKUPRC   STA   RCSUM
0C77:             157 *
```

```
0C77:18          158         CLC              ;SET NEW INPUT BYTE VECTOR
0C78:68          159         PLA              ;  PULL LO ADDRESS( -1)
0C79:69 01       160         ADC   #1
0C7B:8D 10 OC    161         STA   STATE
0C7E:68          162         PLA              ;  PULL HI ADDRESS
0C7F:69 00       163         ADC   #0
0C81:8D 11 OC    164         STA   STATE+1
0C84:            165 *
0C84:20 3F FF    166         JSR   IOREST     ;RESTORE REGISTERS
0C87:58          167         CLI              ;ENABLE INTERRUPT
0C88:40          168         RTI
0C89:            169 *
0C89:            170 *-------------------------------
0C89:            171 *   PACKET RECEIVING
0C89:            172 *-------------------------------
0C89:20 6B OC    173 STPAK   JSR   GETBT
0C8C:C9 90       174 STPAK1  CMP   #DLE       ;CHECK FOR 'DLE' BYTE
0C8E:D0 F9       175         BNE   STPAK
0C90:            176 *
0C90:20 6B OC    177         JSR   GETBT
0C93:C9 83       178         CMP   #STX       ;CHECK FOR 'STX' BYTE
0C95:D0 F2       179         BNE   STPAK
0C97:A2 00       180         LDX   #0         ; ZERO RECD CHECKSUM
0C99:8E 0A OC    181         STX   RCSUM
0C9C:8E 0B OC    182         STX   RCSUM+1
0C9F:            183 *
0C9F:20 6B OC    184         JSR   GETBT      ;GET OPCODE
0CA2:8D 07 OC    185         STA   OPCDE      ;  SAVE IT
0CA5:            186 *
0CA5:            187 *   VERIFY CHECKSUM
0CA5:            188 *
0CA5:20 6B OC    189 VERCHK  JSR   GETBT      ;GET 'DLE' BYTE
0CA8:20 6B OC    190         JSR   GETBT      ;GET 'ETX' BYTE
0CAB:AE 0A OC    191 VERIFY  LDX   RCSUM      ;SAVE LO BYTE RECD CHECKSUM
0CAE:8E 0E OC    192         STX   SAVSUM
0CB1:AE 0B OC    193         LDX   RCSUM+1    ;SAVE HI BYTE RECD CHECKSUM
0CB4:8E 0F OC    194         STX   SAVSUM+1
0CB7:            195 *
0CB7:20 6B OC    196         JSR   GETBT      ;GET LO BYTE XMITTED CHECKSUM
0CBA:CD 0E OC    197         CMP   SAVSUM     ;  CHECK IT
0CBD:D0 CA       198         BNE   STPAK
0CBF:            199 *
0CBF:20 6B OC    200         JSR   GETBT      ;GET HI BYTE XMITTED CHECKSUM
0CC2:CD 0F OC    201         CMP   SAVSUM+1   ;  CHECK IT
0CC5:D0 C2       202         BNE   STPAK
0CC7:            203 *
0CC7:            204 *   DECIPHER OPCODE
0CC7:            205 *
0CC7:AD 07 OC    206         LDA   OPCDE      ;LOAD OPCODE
0CCA:C9 34       207         CMP   #ACK       ;  ACK ?
0CCC:F0 18       208         BEQ   ACKN
0CCE:C9 32       209         CMP   #COM       ;  COMMANDS ?
0CD0:F0 1C       210         BEQ   COMN
0CD2:C9 45       211         CMP   #COMEND    ;  COMMAND PACKET COMPLETED ?
0CD4:D0 06       212         BNE   CHKDTR     ;  -NO,GO ON
0CD6:20 1E OD    213         JSR   SNDACK     ;  -YES,SEND ACK
```

```
0CD9:4C 89 0C   214          JMP   STPAK
0CDC:C9 31      215 CHKDTR   CMP   #DRUN    ;  DATA RUN ?
0CDE:D0 A9      216          BNE   STPAK    ;  -NO,BAD OPCODE
0CE0:20 1E 0D   217          JSR   SNDACK   ;  -YES,SEND ACK
0CE3:4C A6 0E   218          JMP   DATRUN   ;   JUMP TO DATA RUN
0CE6:           219 *
0CE6:           220 *----------------------------------
0CE6:           221 *  RECEIVED ACKNOWLEDGE
0CE6:           222 *----------------------------------
0CE6:A9 01      223 ACKN     LDA   #1       ;SET ACK FLAG
0CE8:8D 03 0C   224          STA   ACKFLG
0CEB:4C 89 0C   225          JMP   STPAK
0CEE:           226 *
0CEE:           227 *
0CEE:           228 *----------------------------------
0CEE:           229 *  RECEIVE COMMANDS
0CEE:           230 *----------------------------------
0CEE:A9 45      231 CGMN     LDA   #COMEND  ;STORE 'COMMANDS COMPLETED' OPC
0CF0:8D 07 0C   232          STA   OPCDE
0CF3:20 6B 0C   233          JSR   GETBT
0CF6:C9 90      234          CMP   #DLE     ;CHECK FOR 'DLE' BYTE
0CF8:D0 8F      235          BNE   STPAK
0CFA:           236 *
0CFA:20 6B 0C   237          JSR   GETBT
0CFD:C9 81      238          CMP   #CTX     ;CHECK FOR 'CTX' BYTE
0CFF:D0 88      239          BNE   STPAK
0D01:A2 00      240          LDX   #0       ; ZERO RECD CHECKSUM
0D03:8E 0A 0C   241          STX   RCSUM
0D06:8E 0B 0C   242          STX   RCSUM+1
0D09:           243 *
0D09:20 6B 0C   244          JSR   GETBT    ;GET LASER REP NUMBER
0D0C:8D 00 0C   245          STA   REPNUM
0D0F:20 6B 0C   246          JSR   GETBT    ;GET LO BYTE SHOTS/PROF
0D12:8D 01 0C   247          STA   MSHOTS
0D15:20 6B 0C   248          JSR   GETBT    ;GET HI BYTE SHOTS/PROF
0D18:8D 02 0C   249          STA   MSHOTS+1
0D1B:4C A5 0C   250          JMP   VERCHK
0D1E:           251 *
0D1E:           252 **********************************
0D1E:           253 *
0D1E:           254 *  APPLE SENDER
0D1E:           255 *
0D1E:           256 **********************************
0D1E:           257 *
0D1E:           258 *================================
0D1E:           259 *  SEND ACKNOWLEDGE
0D1E:           260 *================================
0D1E:A9 34      261 SNDACK   LDA   #ACK     ;LOAD ACK OPCODE
0D20:8D 04 0C   262          STA   DATOPC
0D2  20 5C 0E   263          JSR   SNDFRM   ;SEND ACK FRAME
0D26:60         264          RTS
0D27:           265 *
0D27:           266 *================================
0D27:           267 *  SEND PROFILE DONE FRAME
0D27:           268 *================================
0D27:58         269 SNDPRD   CLI
```

```
0D28:A9 33      270           LDA  #PRDONE  ;LOAD 'PRDONE' OPCODE
0D2A:8D 04 0C   271           STA  DATOPC
0D2D:A9 03      272           LDA  #ERRCNT  ;LOAD ERROR COUNT REG
0D2F:8D 05 0C   273           STA  ERRORS
0D32:58         274           CLI
0D33:           275 *
0D33:A9 00      276 SNDPR1    LDA  #0        ;CLEAR ACK FLAG
0D35:8D 03 0C   277           STA  ACKFLG
0D38:20 5C 0E   278           JSR  SNDFRM    ;SEND PROFILE DONE FRAME
0D3B:           279 *
0D3B:A0 32      280           LDY  #DEL1     ;WAIT FOR ACK
0D3D:A9 10      281 ACKWT1    LDA  #16
0D3F:20 A8 FC   282           JSR  WAIT      ;  DELAY
0D42:AD 03 0C   283           LDA  ACKFLG    ;  CHECK ACK FLAG
0D45:D0 0B      284           BNE  OKACK1
0D47:88         285           DEY
0D48:D0 F3      286           BNE  ACKWT1
0D4A:CE 05 0C   287 ERRCK1    DEC  ERRORS
0D4D:D0 E4      288           BNE  SNDPR1
0D4F:4C F7 0F   289           JMP  ERRPRT
0D52:           290 *
0D52:60         291 OKACK1    RTS
0D53:           292 *================================
0D53:           293 *   SEND DATA PACKETS
0D53:           294 *================================
0D53:           295 *
0D53:           296 *   SEND LO BYTE DATA
0D53:           297 *
0D53:20 EA 10   298 SNDDAT    JSR  TDPRT     ;PRINT 'TRANSMITTING DATA'
0D56:58         299           CLI
0D57:A9 50      300           LDA  #OPCLO    ;LOAD BASE OPCODE
0D59:8D 04 0C   301           STA  DATOPC
0D5C:           302 *
0D5C:A9 00      303           LDA  #>STADDL  ;LOAD STARTING ADDRESS
0D5E:85 F9      304           STA  ADDRES
0D60:A9 90      305           LDA  #<STADDL
0D62:85 FA      306           STA  ADDRES+1
0D64:           307 *
0D64:A9 D0      308           LDA  #>BYTCNT  ;LOAD BYTE COUNT
0D66:85 FB      309           STA  BYTES1
0D68:A9 07      310           LDA  #<BYTCNT
0D6A:85 FC      311           STA  BYTES1+1
0D6C:           312 *
0D6C:20 8B 0D   313           JSR  DATOUT    ;GO SEND LO BYTES
0D6F:           314 *
0D6F:           315 *   SEND HI BYTE DATA
0D6F:           316 *
0D6F:A9 E0      317           LDA  #OPCHI    ;LOAD BASE OPCODE
0D71:8D 04 0C   318           STA  DATOPC
0D74:           319 *
0D74:A9 00      320           LDA  #>STADDH  ;LOAD STARTING ADDRESS
0D76:85 F9      321           STA  ADDRES
0D78:A9 98      322           LDA  #<STADDH
0D7A:85 FA      323           STA  ADDRES+1
0D7C:           324 *
0D7C:A9 D0      325           LDA  #>BYTCNT  ;LOAD BYTE COUNT
```

```
0D7E:85 FB      326          STA   BYTES1
0D80:A9 07      327          LDA   #<BYTCNT
0D82:85 FC      328          STA   BYTES1+1
0D84:          329 *
0D84:20 8B 0D   330          JSR   DATOUT    ;GO SEND HI BYTES
0D87:          331 *
0D87:20 27 0D   332          JSR   SNDPRD    ;GO SEND 'PROFILE DONE' FRAME
0D8A:          333 *
0D8A:60         334 DDONE    RTS
0D8B:          335 *
0D8B:          336 *-------------------------------
0D8B:          337 *  DATA PACKET SENDING ROUTINE
0D8B:          338 *-------------------------------
0D8B:          339 *
0D8B:          340 *  CHECK FOR LAST PACKET (SET LAST PACKET FLAG)
0D8B:          341 *
0D8B:A9 00      342 DATOUT   LDA   #0        ;INITIALIZE LAST PACKET FLAG
0D8D:8D 06 0C   343          STA   LSTFLG
0D90:          344 *
0D90:A9 07      345 DTOUT1   LDA   #<PAKLEN  ;  CHECK HI BYTE
0D92:C5 FC      346          CMP   BYTES1+1
0D94:F0 05      347          BEQ   CHKLO
0D96:90 1B      348          BCC   FULPAK
0D98:4C A1 0D   349          JMP   SETFLG
0D9B:A9 D0      350 CHKLO    LDA   #>PAKLEN  ;  CHECK LO BYTE
0D9D:C5 FB      351          CMP   BYTES1
0D9F:90 12      352          BCC   FULPAK
0DA1:          353 *
0DA1:A9 01      354 SETFLG   LDA   #1        ;SET LAST PACKET FLAG
0DA3:8D 06 0C   355          STA   LSTFLG
0DA6:          356 *
0DA6:A5 FB      357          LDA   BYTES1    ;SET BYTE COUNT FOR LAST PACKET
0DA8:8D 08 0C   358          STA   COUNT1
0DAB:A5 FC      359          LDA   BYTES1+1
0DAD:8D 09 0C   360          STA   COUNT1+1
0DB0:4C BD 0D   361          JMP   PAKOUT
0DB3:          362 *
0DB3:A9 D0      363 FULPAK   LDA   #>PAKLEN  ;SET BYTE COUNT FOR FULL PACKET
0DB5:8D 08 0C   364          STA   COUNT1
0DB8:A9 07      365          LDA   #<PAKLEN
0DBA:8D 09 0C   366          STA   COUNT1+1
0DBD:          367 *
0DBD:          368 *  SEND ONE PACKET
0DBD:          369 *
0DBD:A9 03      370 PAKOUT   LDA   #ERRCNT   ;INITIALIZE ERROR COUNT REGISTER
0DBF:8D 05 0C   371          STA   ERRORS
0DC2:A9 00      372 PAKOT1   LDA   #0        ;INITIALIZE ACK FLAG
0DC4:8D 03 0C   373          STA   ACKFLG
0DC7:          374 *
0DC7:20 5C 0E   375          JSR   SNDFRM    ;SEND DATA FRAME PART 1
0DCA:20 0C 0E   376          JSR   SNDPK2    ;SEND DATA FRAME PART 2
0DCD:          377 *
0DCD:A0 32      378          LDY   #DEL1     ;WAIT FOR ACK FLAG
0DCF:A9 10      379 ACKWT2   LDA   #16
0DD1:20 A8 FC   380          JSR   WAIT      ;  DELAY
0DD4:AD 03 0C   381          LDA   ACKFLG    ;  CHECK ACKFLAG
```

```
0DD7:D0 0B      382             BNE    OKACK2
0DD9:88         383             DEY
0DDA:D0 F3      384             BNE    ACKWT2
0DDC:           385 *
0DDC:CE 05 0C   386 ERRCK2      DEC    ERR.RS
0DDF:D0 E1      387             BNE    PAKUT1
0DE1:4C F7 0F   388             JMP    ERRPRT
0DE4:           389 *
0DE4:           390 *   CHECK LAST PACKET FLAG
0DE4:           391 *
0DE4:AD 06 0C   392 OKACK2      LDA    LSTFLG    ;LAST PACKET DONE ?
0DE7:C9 01      393             CMP    #1
0DE9:F0 20      394             BEQ    DONE      ; -YES,GOTO DONE
0DEB:           395 *
0DEB:           396 *   DETERMINE NEW STARTING ADDRESS, BYTE COUNT, & OPCODE
0DEB:           397 *
0DEB:18         398 NEWADD      CLC              ;NEW ADDRESS
0DEC:A9 D0      399             LDA    #>PAKLEN  ; LO BYTE
0DEE:65 F9      400             ADC    ADDRES
0DF0:85 F9      401             STA    ADDRES
0DF2:           402 *
0DF2:A9 07      403             LDA    #<PAKLEN  ; HI BYTE
0DF4:65 FA      404             ADC    ADDRES+1
0DF6:85 FA      405             STA    ADDRES+1
0DF8:           406 *
0DF8:38         407             SEC              ;NEW BYTE COUNT
0DF9:A5 FB      408             LDA    BYTES1    ; LO BYTE
0DFB:E9 D0      409             SBC    #>PAKLEN
0DFD:85 FB      410             STA    BYTES1
0DFF:           411 *
0DFF:A5 FC      412             LDA    BYTES1+1  ; HI BYTE
0E01:E9 07      413             SBC    #<PAKLEN
0E03:85 FC      414             STA    BYTES1+1
0E05:           415 *
0E05:EE 04 0C   416             INC    DATOPC    ;NEW OPCODE
0E08:4C 90 0D   417             JMP    DTOUT1    ;SEND NEXT PACKET
0E0B:           418 *
0E0B:60         419 DONE        RTS              ;DATA BLOCK SENDING COMPLETE
0E0C:           420 *
0E0C:           421 *-----------------------------------
0E0C:           422 *   DATA PACKET PART 2
0E0C:           423 *   SENDING ROUTINE
0E0C:           424 *-----------------------------------
0E0C:A5 FA      425 SNDPK2      LDA    ADDRES+1  ;SAVE HI BYTE OF ADDRESS
0E0E:8D 0C 0C   426             STA    SAVADD
0E11:           427 *
0E11:A9 90      428             LDA    #DLE
0E13:20 99 0E   429             JSR    SNDBYT    ;SEND 'DLE' CHARACTER
0E16:A9 00      430             LDA    #0        ;CLEAR XMIT CHECKSUM
0E18:8D 14 0C   431             STA    XCSUM
0E1B:8D 15 0C   432             STA    XCSUM+1
0E1E:           433 *
0E1E:A9 81      434             LDA    #CTX
0E20:20 8B 0E   435             JSR    UPDTCK    ;UPDATE XMIT CHECKSUM & SEND 'CTX'
0E23:           436 *
0E23:AE 09 0C   437             LDX    COUNT1+1
```

```
0E26:A0 00      438 SENDPG  LDY  #0
0E28:E0 00      439         CPX  #0        ;LAST PAGE TO BE SENT ?
0E2A:F0 11      440         BEQ  LASTPG    ; -YES,SEND LAST PAGE
0E2C:10 08      441         BPL  FULLPG    ; -NO,SEND FULL PAGE
0E2E:AD 0C 0C   442         LDA  SAVADD    ; -PACKET COMPLETED
0E31:85 FA      443         STA  ADDRES+1  ;   RESTORE INITIAL ADDRESS
0E33:4C 74 0E   444         JMP  ENDER     ;   SEND PACKET ENDER
0E36:           445 *
0E36:A9 00      446 FULLPG  LDA  #0        ;SET BYTE COUNT FOR FULL PAGE
0E38:85 FF      447         STA  TLBTCT
0E3A:4C 46 0E   448         JMP  SNDIT1
0E3D:           449 *
0E3D:AD 08 0C   450 LASTPG  LDA  COUNT1    ;SET BYTE COUNT FOR LAST PAGE
0E40:85 FF      451         STA  TLBTCT
0E42:           452 *
0E42:C4 FF      453 SENDIT  CPY  TLBTCT    ;LAST BYTE OF PAGE SENT ?
0E44:F0 10      454         BEQ  NEXTPG    ; -YES,SET UP FOR NEXT PAGE
0E46:B1 F9      455 SNDIT1  LDA  (ADDRES),Y ;-NO,SEND ANOTHER
0E48:20 8B 0E   456         JSR  UPDTCK
0E4B:C9 90      457         CMP  #DLE      ;DATA = 'DLE' BYTE ?
0E4D:D0 03      458         BNE  SKDBL     ; -NO,SKIP DOUBLE 'DLE'
0E4F:20 8B 0E   459         JSR  UPDTCK    ; -YES,SEND SECOND 'DLE'
0E52:C8         460 SKDBL   INY
0E53:4C 42 0E   461         JMP  SENDIT    ;GO TO SEND ANOTHER BYTE
0E56:           462 *
0E56:CA         463 NEXTPG  DEX
0E57:E6 FA      464         INC  ADDRES+1  ;SET ADDRESS FOR NEXT PAGE
0E59:4C 26 0E   465         JMP  SENDPG
0E5C:           466 *
0E5C:           467 *--------------------------------
0E5C:           468 *  HEADER PACKET SENDING ROUTINE
0E5C:           469 *--------------------------------
0E5C:A9 90      470 SNDFRM  LDA  #DLE
0E5E:20 99 0E   471         JSR  SNDBYT    ;SEND 'DLE' CHARACTER
0E61:A9 00      472         LDA  #0        ;CLEAR XMIT CHECKSUM
0E63:8D 14 0C   473         STA  XCSUM
0E66:8D 15 0C   474         STA  XCSUM+1
0E69:           475 *
0E69:A9 83      476         LDA  #STX
0E6B:20 8B 0E   477         JSR  UPDTCK    ;UPDATE XMIT CHECKSUM & SEND 'STX'
0E6E:           478 *
0E6E:AD 04 0C   479         LDA  DATOPC
0E71:20 8B 0E   480         JSR  UPDTCK    ;UPDATE XMIT CHECKSUM & SEND OPCODE
0E74:           481 *
0E74:A9 90      482 ENDER   LDA  #DLE
0E76:20 8B 0E   483         JSR  UPDTCK    ;UPDATE XMIT CHECKSUM & SEND 'DLE'
0E79:           484 *
0E79:A9 82      485         LDA  #ETX
0E7B:20 99 0E   486         JSR  SNDBYT    ;SEND 'ETX' CHARACTER
0E7E:           487 *
0E7E:AD 14 0C   488 SNDCHK  LDA  XCSUM     ;SEND XMIT CHECKSUM
0E81:20 99 0E   489         JSR  SNDBYT    ;  LO BYTE
0E84:AD 15 0C   490         LDA  XCSUM+1
0E87:20 99 0E   491         JSR  SNDBYT    ;  HI BYTE
0E8A:           492 *
0E8A:60         493         RTS
```

```
OE8B:              494 *
OE8B:              495 *---------------------------------
OE8B:              496 *   UPDATE CHECKSUM & SEND BYTE
OE8B:              497 *            ROUTINES
OE8B:              498 *---------------------------------
OE8B:48            499 UPDTCK  PHA                 ;SAVE BYTE
OE8C:18            500         CLC
OE8D:6D 14 OC      501         ADC     XCSUM       ;UPDATE CHECKSUM
OE90:90 03         502         BCC     SKUPXC
OE92:EE 15 OC      503         INC     XCSUM+1
OE95:8D 14 OC      504 SKUPXC  STA     XCSUM
OE98:68            505         PLA                 ;RESTORE BYTE
OE99.              506 *
OE99:48            507 SNDBYT  PHA                 ;SAVE BYTE
OE9A:AD A0 CO      508 LOOP1   LDA     CSR         ;CHECK ACIA STATUS
OE9D:29 02         509         AND     #2
OE9F:F0 F9         510         BEQ     LOOP1
OEA1:              511 *
OEA1:68            512         PLA                 ;RESTORE BYTE
OEA2:8D A1 CO      513         STA     XRDR        ;SEND BYTE
OEA5:60            514         RTS
OEA6:              515 *
OEA6:              516 *********************************
OEA6:              517 *
OEA6:              518 *   DATA RUN
OEA6:              519 *
OEA6:              520 *********************************
OEA6:              521 *
OEA6:20 D0 10      522 DATRUN  JSR     TKPPRT
OEA9:AD 01 OC      523         LDA     MSHOTS      ;LOAD SHOT COUNTER TSHOT
OEAC:8D 12 OC      524         STA     TSHOT
OEAF:AD 02 OC      525         LDA     MSHOTS+1
OEB2:8D 13 OC      526         STA     TSHOT+1
OEB5:              527 *
OEB5:A9 00         528         LDA     #0          ;CHECK LO BYTE SHOT COUNT
OEB7:CD 12 OC      529         CMP     TSHOT       ;  = 0 ?
OEBA:D0 03         530         BNE     IDMAC       ;  -NO,GO ON
OEBC:CE 13 OC      531         DEC     TSHOT+1     ;  -YES,DEC HI BYTE SHOT COUNT
OEBF:              532 *--------------------------------
OEBF:              533 *  INITIALIZE DMAC
OEBF:              534 *--------------------------------
OEBF:A9 60         535 IDMAC   LDA     #<BUFADD    ;LOAD ADDRESS HI BYTE
OEC1:8D 04 C4      536         STA     DMAREG+4    ;  CHANNEL 1
OEC4:8D 0C C4      537         STA     DMAREG+12   ;  CHANNEL 3
OEC7:A9 00         538         LDA     #>BUFADD    ;LOAD ADDRESS LO BYTE
OEC9:8D 05 C4      539         STA     DMAREG+5    ;  CHANNEL 1
OECC:8D 0D C4      540         STA     DMAREG+13   ;  CHANNEL 3
OECF:A9 07         541         LDA     #$07        ;LOAD COUNT HI BYTE
OED1:8D 06 C4      542         STA     DMAREG+6    ;  CHANNEL 1
OED4:8D 0E C4      543         STA     DMAREG+14   ;  CHANNEL 3
OED7:A9 D1         544         LDA     #$D1        ;LOAD COUNT LO BYTE
OED9:8D 07 C4      545         STA     DMAREG+7    ;  CHANNEL 1
OEDC:8D 0F C4      546         STA     DMAREG+15   ;  CHANNEL 3
OEDF:A9 02         547         LDA     #2          ;LOAD CHANNEL CONTROL
OEE1:8D 11 C4      548         STA     DMAREG+17
OEE4:A9 0C         549         LDA     #0          ;LOAD INTERRUPT CONTROL
```

```
OEE6:8D 15 C4   550           STA    DMAREG+21
OEE9:A9 03      551           LDA    #3          ;LOAD DATA CHAIN
OEEB:8D 16 C4   552           STA    DMAREG+22
OEEE:A9 02      553           LDA    #2          ;LOAD PRIORITY CONTROL
OEF0:8D 14 C4   554           STA    DMAREG+20
OEF3:           555  *-------------------------------
OEF3:           556  *   INIT MAINFRAME MEMORY
OEF3:           557  *-------------------------------
OEF3:A9 00      558           LDA    #>STADDL    ;LOAD STARTING ADDRESS
OEF5:85 FD      559           STA    PRTADD
OEF7:A9 90      560           LDA    #<STADDL
OEF9:85 FE      561           STA    PRTADD+1
OEFB:           562  *
OEFB:A2 10      563           LDX    #16         ;LOAD X WITH 16 (PAGES)
OEFD:A9 00      564           LDA    #0          ;CLEAR A & Y
OEFF:A8         565           TAY
OF00:           566  *
OF00:91 FD      567  CLRLP    STA    (PRTADD),Y  ;CLEAR MEMORY
OF02:C8         568           INY
OF03:D0 FB      569           BNE    CLRLP
OF05:E6 FE      570           INC    PRTADD+1
OF07:CA        571           DEX
OF08:D0 F6      572           BNE    CLRLP
OFOA:           573  *-------------------------------
OFOA:           574  *   FIRE LASER & CHECK FOR
OFOA:           575  *      DMA DONE
OFOA:           576  *-------------------------------
OFOA:8D C1 CO   577  TRIG1    STA    TRIG        ;FIRE  LASER
OFOD:EA         578           NOP
OFOE:EA         579           NOP
OFOF:EA         580           NOP
OF10:A9 04      581           LDA    #4          ;'WAIT' DURING DMA
OF12:20 A8 FC   582           JSR    WAIT
OF15:2C 11 C4   583           BIT    DMAREG+17   ;DMA BIT SET ?
OF18:30 0C      584           BMI    INTGRT      ; -YES,GOTO INTGRT
OF1A:8D C4 CO   585           STA    SLPRST      ; -NO,RESET SLIPP INTERFACE
OF1D:20 3A 10   586           JSR    TRGPRT      ;   PRINT ERROR MESSAGE
OF20:20 D0 10   587           JSR    TKPPRT
OF23:4C 0A OF   588           JMP    TRIG1
OF26:           589  *-------------------------------
OF26:           590  *   INTEGRATION
OF26:           591  *-------------------------------
OF26:A0 00      592  INTGRT   LDY    #0          ;INIT Y
OF28:18         593  ADD0     CLC                ;FIRST PAGE ADDITION
OF29:B9 00 60   594           LDA    BUFADD,Y    ; LOW BYTE ADD
OF2C:79 00 90   595           ADC    STADDL,Y
OF2F:99 00 90   596           STA    STADDL,Y
OF32:A9 00      597           LDA    #0          ; HIGH BYTE ADD
OF34:79 00 98   598           ADC    STADDH,Y
OF37:99 00 98   599           STA    STADDH,Y
OF3A:18         600  ADD1     CLC                ;SECOND PAGE ADDITIONS
OF3B:B9 00 61   601           LDA    BUFADD+$100,Y
OF3E:79 00 91   602           ADC    STADDL+$100,Y
OF41:99 00 91   603           STA    STADDL+$100,Y
OF44:A9 00      604           LDA    #0
OF46:79 00 99   605           ADC    STADDH+$100,Y
```

```
0F49:99 00 99  606          STA   STADDH+$100,Y
0F4C:18        607 ADD2     CLC                ;THIRD PAGE ADDITIONS
0F4D:B9 00 62  608          LDA   BUFADD+$200,Y
0F50:79 00 92  609          ADC   STADDL+$200,Y
0F53:99 00 92  610          STA   STADDL+$200,Y
0F56:A9 00     611          LDA   #0
0F58:79 00 9A  612          ADC   STADDH+$200,Y
0F5B:99 00 9A  613          STA   STADDH+$200,Y
0F5E:18        614 ADD3     CLC                ;FORTH PAGE ADDITIONS
0F5F:B9 00 63  615          LDA   BUFADD+$300,Y
0F62:79 00 93  616          ADC   STADDL+$300,Y
0F65:99 00 93  617          STA   STADDL+$300,Y
0F68:A9 00     618          LDA   #0
0F6A:79 00 9B  619          ADC   STADDH+$300,Y
0F6D:99 00 9B  620          STA   STADDH+$300,Y
0F70:18        621 ADD4     CLC                ;FIFTH PAGE ADDITIONS
0F71:B9 00 64  622          LDA   BUFADD+$400,Y
0F74:79 00 94  623          ADC   STADDL+$400,Y
0F77:99 00 94  624          STA   STADDL+$400,Y
0F7A:A9 00     625          LDA   #0
0F7C:79 00 9C  626          ADC   STADDH+$400,Y
0F7F:99 00 9C  627          STA   STADDH+$400,Y
0F82:18        628 ADD5     CLC                ;SIXTH PAGE ADDITIONS
0F83:B9 00 65  629          LDA   BUFADD+$500,Y
0F86:79 00 95  630          ADC   STADDL+$500,Y
0F89:99 00 95  631          STA   STADDL+$500,Y
0F8C:A9 00     632          LDA   #0
0F8E:79 00 9D  633          ADC   STADDH+$500,Y
0F91:99 00 9D  634          STA   STADDH+$500,Y
0F94:18        635 ADD6     CLC                ;SEVENTH PAGE ADDITIONS
0F95:B9 00 66  636          LDA   BUFADD+$600,Y
0F98:79 00 96  637          ADC   STADDL+$600,Y
0F9B:99 00 96  638          STA   STADDL+$600,Y
0F9E:A9 00     639          LDA   #0
0FA0:79 00 9E  640          ADC   STADDH+$600,Y
0FA3:99 00 9E  641          STA   STADDH+$600,Y
0FA6:18        642 ADD7     CLC                ;EIGHTH PAGE ADDITIONS
0FA7:B9 00 67  643          LDA   BUFADD+$700,Y
0FAA:79 00 97  644          ADC   STADDL+$700,Y
0FAD:99 00 97  645          STA   STADDL+$700,Y
0FB0:A9 00     646          LDA   #0
0FB2:79 00 9F  647          ADC   STADDH+$700,Y
0FB5:99 00 9F  648          STA   STADDH+$700,Y
0FB8:          649 *
0FB8:C8        650          INY
0FB9:F0 03     651          BEQ   NXTSHT
0FBB:4C 28 0F  652          JMP   ADD0
0FBE:          653 *
0FBE:          654 *--------------------------------
0FBE:          655 *  CHECK SHOT COUNTER
0FBE:          656 *     SEND PROFILE
0FBE:          657 *--------------------------------
0FBE:CE 12 0C  658 NXTSHT   DEC   TSHOT        ;CHECK SHOT COUNTER
0FC1:D0 05     659          BNE   SKUPTS
0FC3:CE 13 0C  660          DEC   TSHOT+1
0FC6:30 0E     661          BMI   PDONE
```

```
OFC8:            662 *
OFC8:A0 06       663 SKUPTS  LDY   #6
OFCA:AD 00 OC    664 RPLOOP  LDA   REPNUM    ;LASER REP DELAY
OFCD:20 A8 FC    665         JSR   WAIT
OFD0:B8          666         DEY
OFD1:D0 F7       667         BNE   RPLOOP
OFD3:4C 0A OF    668         JMP   TRIG1     ;GO FIRE ANOTHER SHOT
OFD6:            669 *
OFD6:A9 8C       670 PDONE   LDA   #>STPAK1  ;RESET INPUT BYTE VECTOR
OFD8:8D 10 OC    671         STA   STATE
OFDB:A9 OC       672         LDA   #<STPAK1
OFDD:8D 11 OC    673         STA   STATE+1
OFE0:            674 *
OFE0:20 53 OD    675         JSR   SNDDAT    ;SEND DATA
OFE3:20 B3 10    676         JSR   IDLPRT    ;PRINT 'IDLE'
OFE6:4C 89 OC    677         JMP   STPAK
OFE9:            678 *
OFE9:            679 **********************************
OFE9:            680 *
OFE9:            681 *   PRINTING ROUTINES
OFE9:            682 *
OFE9:            683 **********************************
OFE9:            684 *
OFE9:            685 *   PRINT SUBROUTINE
OFE9:            686 *
OFE9:A0 00       687 PRINT   LDY   #0
OFEB:B1 FD       688 PRINT1  LDA   (PRTADD),Y
OFED:F0 07       689         BEQ   ENDP
OFEF:20 F0 FD    690         JSR   COUT1
OFF2:C8          691         INY
OFF3:4C EB OF    692         JMP   PRINT1
OFF6:60          693 ENDP    RTS
OFF7:            694 *---------------------------------
OFF7:            695 *   DATA TRANSMISSION ERROR MESSAGE
OFF7:            696 *---------------------------------
OFF7:20 58 FC    697 ERRPRT  JSR   HOME      ;CLEAR SCREEN
OFFA:            698 *
OFFA:A9 0A       699         LDA   #10       ;SET VERTICAL TAB
OFFC:85 25       700         STA   CV
OFFE:20 22 FC    701         JSR   VTAB
1001:            702 *
1001:A9 09       703         LDA   #9        ;SET HORIZONTAL TAB
1003:85 24       704         STA   CH
1005:20 B0 FE    705         JSR   SETINV    ;SET INVERSE SCREEN
1008:A9 57       706         LDA   #ERRMSG
100A:85 FD       707         STA   PRTADD
100C:A9 11       708         LDA   #<ERRMSG
100E:85 FE       709         STA   PRTADD+1
1010:20 E9 OF    710         JSR   PRINT     ;PRINT ERROR MESSAGE
1013:            711 *
1013:A9 0A       712         LDA   #10       ;SET HORIZONTAL TAB
1015:85 24       713         STA   CH
1017:20 84 FE    714         JSR   SETNRM    ;SET NORMAL SCREEN
101A:A9 72       715         LDA   #>TRYAGN
101C:85 FD       716         STA   PRTADD
101E:A9 11       717         LDA   #<TRYAGN
```

```
1020:85 FE      718             STA     PRTADD+1
1022:20 E9 0F   719             JSR     PRINT       ;PRINT TRY AGAIN
1025:           720 *
1025:20 0C FD   721             JSR     RDKEY       ;GET RESPONSE
1028:C9 D9      722             CMP     #'Y
102A:D0 06      723             BNE     NTAGN
102C:20 58 FC   724             JSR     HOME
102F:4C 53 0D   725             JMP     SNDDAT
1032:20 58 FC   726 NTAGN       JSR     HOME
1035:68         727             PLA                 ;GET OFF STACK
1036:68         728             PLA
1037:4C BA 0D   729             JMP     DDONE
103A:           730 *
103A:           731 *--------------------------------
103A:           732 *  LASER TRIGGER ERROR MESSAGE
103A:           733 *--------------------------------
103A:20 58 FC   734 TRGPRT      JSR     HOME
103D:A9 0A      735             LDA     #10
103F:85 25      736             STA     CV
1041:20 22 FC   737             JSR     VTAB
1044:           738 *
1044:A9 09      739             LDA     #9
1046:85 24      740             STA     CH
1048:20 80 FE   741             JSR     SETINV
104B:A9 85      742             LDA     #>TGMSG1
104D:85 FD      743             STA     PRTADD
104F:A9 11      744             LDA     #<TGMSG1
1051:85 FE      745             STA     PRTADD+1
1053:20 E9 0F   746             JSR     PRINT
1056:           747 *
1056:A9 07      748             LDA     #7
1058:85 24      749             STA     CH
105A:20 84 FE   750             JSR     SETNRM
105D:A9 9F      751             LDA     #>TGMSG2
105F:85 FD      752             STA     PRTADD
1061:A9 11      753             LDA     #<TGMSG2
1063:85 FE      754             STA     PRTADD+1
1065:20 E9 0F   755             JSR     PRINT
1068:           756 *
1068:20 0C FD   757             JSR     RDKEY
106B:60         758             RTS
106C:           759 *
106C:           760 *--------------------------------
106C:           761 *  HEADER MESSAGE
106C:           762 *--------------------------------
106C:20 58 FC   763 HEDPRT      JSR     HOME
106F:A9 06      764             LDA     #6
1071:85 25      765             STA     CV
1073:20 22 FC   766             JSR     VTAB
1076:           767 *
1076:A9 0D      768             LDA     #13
1078:85 24      769             STA     CH
107A:A9 04      770             LDA     #>HDMSG1
107C:85 FD      771             STA     PRTADD
107E:A9 11      772             LDA     #<HDMSG1
1080:85 FE      773             STA     PRTADD+1
```

```
1082:20 E9 0F  774        JSR   PRINT
1085:          775 *
1085:A9 08     776        LDA   #8
1087:85 24     777        STA   CH
1089:A9 15     778        LDA   #>HDMSG2
108B:85 FD     779        STA   PRTADD
108D:A9 11     780        LDA   #<HDMSG2
108F:85 FE     781        STA   PRTADD+1
1091:20 E9 0F  782        JSR   PRINT
1094:          783 *
1094:A9 0F     784        LDA   #15
1096:85 24     785        STA   CH
1098:A9 31     786        LDA   #>HDMSG3
109A:85 FD     787        STA   PRTADD
109C:A9 11     788        LDA   #<HDMSG3
109E:85 FE     789        STA   PRTADD+1
10A0:20 E9 0F  790        JSR   PRINT
10A3:          791 *
10A3:A9 08     792        LDA   #8
10A5:85 24     793        STA   CH
10A7:A9 3E     794        LDA   #>HDMSG4
10A9:85 FD     795        STA   PRTADD
10AB:A9 11     796        LDA   #<HDMSG4
10AD:85 FE     797        STA   PRTADD+1
10AF:20 E9 0F  798        JSR   PRINT
10B2:          799 *
10B2:20 CC FD  800        JSR   RDKEY
10B5:60        801        RTS
10B6:          802 *------------------------------
10B6:          803 *  STATUS MESSAGES
10B6:          804 *------------------------------
10B6:          805 *
10B6:          806 *  IDLE MESSAGE
10B6:          807 *
10B6:20 58 FC  808 IDLPRT JSR   HOME
10B9:A9 0A     809        LDA   #10
10BB:85 25     810        STA   CV
10BD:20 22 FC  811        JSR   VTAB
10C0:          812 *
10C0:A9 10     813        LDA   #16
10C2:85 24     814        STA   CH
10C4:A9 BA     815        LDA   #>IDLMSG
10C6:85 FD     816        STA   PRTADD
10C8:A9 11     817        LDA   #<IDLMSG
10CA:85 FE     818        STA   PRTADD+1
10CC:20 E9 0F  819        JSR   PRINT
10CF:60        820        RTS
10D0:          821 *
10D0:          822 *  TAKING PROFILE MESSAGE
10D0:          823 *
10D0:20 58 FC  824 TKPPRT JSR   HOME
10D3:A9 0A     825        LDA   #10
10D5:85 25     826        STA   CV
10D7:20 22 FC  827        JSR   VTAB
10DA:          828 *
10DA:A9 0A     829        LDA   #10
```

```
10DC:85 24      830        STA   CH
10DE:A9 C3      831        LDA   #>TKPMSG
10E0:85 FD      832        STA   PRTADD
10E2:A9 11      833        LDA   #<TKPMSG
10E4:85 FE      834        STA   PRTADD+1
10E6:20 E9 0F   835        JSR   PRINT
10E9:60         836        RTS
10EA:           837 *
10EA:           838 *   TRANSMITTING DATA MESSAGE
10EA:           839 *
10EA:20 5B FC   840 TDPRT  JSR   HOME
10ED:A9 0A      841        LDA   #10
10EF:85 25      842        STA   CV
10F1:20 22 FC   843        JSR   VTAB
10F4:           844 *
10F4:A9 0A      845        LDA   #10
10F6:85 24      846        STA   CH
10F8:A9 D6      847        LDA   #>TDMSG
10FA:85 FD      848        STA   PRTADD
10FC:A9 11      849        LDA   #<TDMSG+1
10FE:85 FE      850        STA   PRTADD+1
1100:20 E9 0F   851        JSR   PRINT
1103:60         852        RTS
1104:           853 *
1104:           854 *-------------------------------
1104:           855 *   DATA
1104:           856 *-------------------------------
1104:C1 D0 D0   857 HDMSG1 ASC   'APPLE     SOFTWARE'
1107:CC C5 A0
110A:D3 CF C6
110D:D4 D7 C1
1110:D2 C5
1112:8D 8D 00   858        DFB   CR,CR,0
1115:D3 CF C4   859 HDMSG2 ASC   'SODIUM    LIDAR PREPROCESSOR'
1118:C9 D5 CD
111B:A0 CC C9
111E:C4 C1 D2
1121:A0 D0 D2
1124:C5 D0 D2
1127:CF C3 C5
112A:D3 D3 CF
112D:D2
112E:8D 8D 00   860        DFB   CR,CR,0
1131:B2 B0 AD   861 HDMSG3 ASC   '20-FEB-83'
1134:C6 C5 C2
1137:AD B8 B3
113A:8D 8D 8D   862        DFB   CR,CR,CR,0
113D:00
113E:BC A0 D0   863 HDMSG4 ASC   '<       PRESS A KEY TO BEGIN >'
1141:D2 C5 D3
1144:D3 A0 C1
1147:A0 CB C5
114A:D9 A0 D4
114D:CF A0 C2
1150:C5 C7 C9
1153:CE A0 BE
```

```
1156:00              864          DFB   0
1157:C4 C1 D4        865 ERRMSG   ASC   'DATA     TRANSMISSION ERROR'
115A:C1 A0 D4
115D:D2 C1 CE
1160:D3 CD C9
1163:D3 D3 C9
1166:CF CE A0
1169:C5 D2 D2
116C:CF D2
116E:8D 8D 8D       866          DFB   CR,CR,CR,0
1171:00
1172:D4 D2 D9       867 TRYAGN   ASC   'TRY      SENDING AGAIN?'
1175:A0 D3 C5
1178:CE C4 C9
117B:CE C7 A0
117E:C1 C7 C1
1181:C9 CE BF
1184:00              868          DFB   0
1185:CC C1 D3       869 TGMSG1   ASC   'LASER    TRIGGER ERROR'
1188:C5 D2 A0
118B:D4 D2 C9
118E:C7 C7 C5
1191:D2 A0 D3
1194:C3 D2 C5
1197:D7 AD D5
119A:D0
119B:8D 8D 8D       870          DFB   CR,CR,CR,0
119E:00
119F:BC D0 D2       871 TGMSG2   ASC   '<PRESS   A KEY TO TRY AGAIN>'
11A2:C5 D3 D3
11A5:A0 C1 A0
11A8:CB C5 D9
11AB:A0 D4 CF
11AE:A0 D4 D2
11B1:D9 A0 C1
11B4:C7 C1 C9
11B7:CE BE
11B9:00              872          DFB   0
11BA:BC A0 C9       873 IDLMSG   ASC   '<       IDLE >'
11BD:C4 CC C5
11C0:A0 BE
11C2:00              874          DFB   0
11C3:BC A0 D4       875 TKPMSG   ASC   '<       TAKING PROFILE >'
11C6:C1 CB C9
11C9:CE C7 A0
11CC:D0 D2 CF
11CF:C6 C9 CC
11D2:C5 A0 BE
11D5:00              876          DFB   0
11D6:BC A0 D4       877 TDMSG    ASC   '<       TRANSMITTING DATA >'
11D9:D2 C1 CE
11DC:D3 CD C9
11DF:D4 D4 C9
11E2:CE C7 A0
11E5:C4 C1 D4
11E8:C1 A0 BE
11EB:00              878          DFB   0
```

0C03 ACKFLG	0CE6 ACKN	0D3D ACKWT1	0DCF ACKWT2
34 ACK	0F38 ADD0	?0F3A ADD1	?0F4C ADD2
?0F5E ADD3	?0F70 ADD4	?0F82 ADD5	?0F94 ADD6
?0FA6 ADD7	F9 ADDRES	6000 BUFADD	07D0 BYTCNT
FB BYTES1	24 CH	0CDC CHKDTR	0D9B CHKLO
0F00 CLRLP	32 COM	45 COMEND	0CEE COMN
0C08 COUNT1	FDF0 COUT1	BD CR	C0A0 CSR
81 CTX	25 CV	0C04 DATOPC	0D8B DATOUT
0EA6 DATRUN	0D8A DDONE	32 DEL1	9C DLE
C400 DMAREG	0E0B DONE	31 DRUN	0D90 DTOUT1
0E74 ENDER	0FF6 ENDP	?0D4A ERRCK1	?0DDC ERRCK2
03 ERRCNT	1157 ERRMSG	0C05 ERRORS	0FF7 ERRPRT
B2 ETX	0E36 FULLPG	0DB3 FULPAK	0C6B SETR?
1104 HDMSG1	1115 HDMSG2	1131 HDMSG3	113E HDMSG4
106C HEDPRT	FC58 HOME	11BA IDLMSG	10B6 IDLPRT
0EBF IDMAC	0F26 INTGRT	03FE INTV	0C5E IOINT
FF3F IOREST	FF4A IOSAVE	C010 KBDSTB	FD1B KEYIN
38 KSW	0E3D LASTPG	0E9A LOOP1	0C06 LSTFLG
0C01 MSHOTS	?0DEB NEWADD	0E56 NEXTPG	1032 NTAGN
0FBE NXTSHT	0D52 OKACK1	0DE4 OKACK2	0C07 OPCDE
E0 OPCHI	50 OPCLO	07D0 PAKLEN	0DC2 PAKOT1
0DBD PAKOUT	0FD6 PDONE	33 PRDONE	0FE9 PRINT
0FEB PRINT1	FD PRTADD	0C0A RCSUM	FD0C RDKEY
0C00 REPNUM	0FCA RPLOOP	0C0C SAVADD	0C0E SAVSUM
0E42 SENDIT	0E26 SENDPG	0DA1 SETFLG	FE80 SETINV
FE84 SETNRM	0E52 SKDBL	0C74 SKUFRC	0FC8 SKUPTS
0E95 SKUPXC	C0C0 SLPIO	C0C4 SLPRST	0D1E SNDACK
0E99 SNDBYT	?0E7E SNDCHK	0D53 SNDDAT	0E5C SNDFRM
0E46 SNDIT1	0E0C SNDPK2	0D33 SNDPK1	0D27 SNDFRD
9800 STADDH	9000 STADDL	0C10 STATE	0C8? STPAK
0C8C STPAK1	83 STX	11D6 TDMSG	10EA TDPRT
1185 TGMSG1	119F TGMSG2	11C3 TKPMSG	10D0 TKPPRT
FF TLBTCT	103A TRGPRT	C0C1 TRIG	0F0A TRIG1
1172 TRYAGN	0C12 TSHOT	0E8? UPDTCK	0CA5 VERCHK
?0CAB VERIFY	FC22 VTAB	FCA8 WAIT	0C5B WLOOP
0C14 XCSUM	C0A1 XRDR		

III.2 FZZ

FORTRAN IV   V02.5-5   Tue 31-May-83 00:58:58   PAGE 001

```
C *************************************
C *
C *      SODIUM LIDAR PROGRAM
C *          11/16/82
C *          D.VOELZ
C *
C *************************************
C THIS PROGRAM ( ALONG WITH ITS ASSOCIATED SUBROUTINES )
C IS DESIGNED TO CONTROL THE SODIUM LIDAR EXPERIMENT
C AT THE UNIVERSITY OF ILLINOIS.
C THE PROGRAM DISPLAYS A MAIN MENU. CHOOSING ONE OF THE
C OPTIONS DISPLAYED CAUSES THE PROGRAM TO GO TO ONE OF
C THE ASSOCIATED SUBROUTINES.
C
C MAIN SUBROUTINES CALLED IN THE PROGRAM FOLLOW:
C  EXPPAR - EXPERIMENT PARAMETERS
C  CSTTUS - CURRENT STATUS
C  LAALN - LOW-ALTITUDE ALIGNMENT ROUTINE
C  ALNRTN - SODIUM ALIGNMENT ROUTINE
C  DATRUN - DATA RUN
C  EXMPRF - EXAMINE PROFILE ON DISK
C  MOUT - HEX DUMP OF THE ARRAY LDATA
C
C OTHER SUBROUTINES USED:
C  CLRSCN - CLEAR TERMINAL SCREEN
C  BAKSCN - NORMAL (BACKGROUND) SCREEN
C  FORSCN - HIGHLIGHT (FOREGROUND) SCREEN
C  DATE - GET PRESENT DATE
C  TIME - GET PRESENT TIME
C
C A DESCRIPTION OF SOME OF THE VARIABLES USED IN THIS PROGRAM
C AND ITS SUBROUTINES FOLLOWS:
C  BINS - # BINS OF INTEREST        BSAT - BASE ALTITUDE (KM)
C  BSATF - BASE ALTITUDE (FT)       B30 - 30 KM BIN POINTER
C  B60 - 60 KM BIN POINTER          C - SPEED OF LIGHT
C  CLA - COLUMN ABUN'D RATIO        CPROF - CURRENT PROFILE
C  CSET - CURRENT SET               EANG - ELEVATION ANGLE (DEGREES)
C  FNAME - FILE NAME ARRAY          GATTIM - RECEIVER GATE TIME
C  HBI - HI BIN OF INTEREST         HDRB - # HEADER BYTES IN DATA FILE
C  HRI - HI RANGE OF INTEREST       IDATE - DATE ARRAY
C  INTSIZ - FILE BLOCK SIZE         ITIME - TIME ARRAY
C  K60 - PHOTONS 60-80 KM           K80 - PHOTONS 80-100 KM
C  K100 - PHOTONS 100-120 KM        LBI - LO BIN OF INTEREST
C  LDATA - DATA ARRAY               LRI - LO RANGE OF INTEREST
C  LREP - DATRUN REP RATE           LSET - PREVIOUS SET #
C  MSHOTS - SHOTS/PROF (SENT)       PDEL - INTER-PROFILE DELAY
C  REANG - ELEVA'N ANGLE (RAD)      RECSIZ - FILE RECORD SIZE
C  REPNUM - REP NUMBER              REPR3 - LAALN REP RATE
C  REPR4 - ALNRTN REP RATE          RG - RANGE GATE (BIN SIZE)
C  SETS - # OF SETS                 SGP - SIGNAL PHOTONS
C  SHOTSP - DATRUN SHOTS/PROF       SHOTS3 - LAALN SHOTS/PROF
C  SHOTS4 - ALNRTN SHOTS/PROF       S20 - # BINS IN 20 KM
C  TSGP - TOTAL SIGNAL PHOTONS
C
```

FORTRAN IV   V02.5-5   Tue 31-May-83 00:58:58   PAGE 002

```
        C
        C  FLAGS:
        C   ERRFLG - SET WHEN A LSI11-APPLE TRANSMISSION ERROR OCCURS
        C   DFLAG - SET WHEN A COMPLETED PROFILE IS RECEIVED
        C   RUNFLG - DESIGNATES WHICH SUBROUTINE (LAALN,ALRTN,DATRUN) IS
        C           BEING RUN
        C
0001          INTEGER SETS,PROFS,SHOTSP,CSET,CPROF,LSET,LPROF,HBI,LBI
             >,SHOTS3,SHOTS4,INTSIZ,RECSIZ,B30,B60,S20,K30,K60,K80,K100
             >,SGP,TSGP,PDEL
0002          REAL LREP,LRI,HRI,EANG,BSAT,REPR3,REPR4,PI,C,GATTIM,RG
             >,CLA,REANG,BSATF
0003          LOGICAL*1 IDATE(9),ITIME(8)
0004          BYTE FNAME(15)
        C
0005          COMMON /DATTIM/ IDATE,ITIME
0006          COMMON /SETS/ SETS,PROFS,SHOTSP
0007          COMMON /LREPS/ LREP,LRI,HRI,LBI,HBI,BSAT,BSATF,EANG
             >,REANG,PDEL
0008          COMMON /STATS/ CSET,CPROF,LSET,LPROF
0009          COMMON /CALCS/ K30,K60,K80,K100,SGP,TSGP,CLA
0010          COMMON /ALN/ REPR4,SHOTS4
0011          COMMON /LAAN/REPR3,SHOTS3
0012          COMMON /DISK/ INTSIZ,RECSIZ,FNAME
0013          COMMON/PTERS/ B30,B60,S20
0014          COMMON /CONST/ PI,C,GATTIM,RG
        C
        C  VARIABLE DEFAULT VALUES
        C
0015          SETS = 20
0016          PROFS = 10
0017          SHOTSP = 100
0018          LREP = 10.0
0019          PDEL=0
0020          EANG = 90.
0021          LRI = 0.075
0022          HRI = 149.925
0023          BSAT = 0.
0024          BSATF = 0.
0025          LBI = 1
0026          HBI = 1000
0027          CSET = 1
0028          CPROF = 1
0029          LSET = 0
0030          LPROF = 0
0031          K30 = 0
0032          K60 = 0
0033          K80 = 0
0034          K100 = 0
0035          SGP = 0
0036          TSGP = 0
0037          CLA = 0.
0038          REPR3=10.
0039          SHOTS3=100
```

```
0040          REPR4=10.
0041          SHOTS4=100
0042          PI=3.1415927
0043          C=3.0E5
0044          GATTIM=1.0E-6
0045          RG=(GATTIM*C)/2.
0046          REANG=PI/2.
0047          DATA FNAME/'D','Y','O',':','S','E','T',0,0,0,'.','D','A'
              >,'T',0/
      C------------------------------------------
      C  INITIALIZATION
      C------------------------------------------
      C  INIT SERIAL RECEIVER INTERRUPT ROUTINE
0048          CALL INITR
      C  SET JOB STATUS FOR TERMINAL CONTINUE MODE
0049          CALL IPOKE("44,"100.OR.IPEEK("44))
      C------------------------------------------
      C  DISPLAY MAIN MENU
      C------------------------------------------
0050        3 CALL CLRSCN
0051          CALL BAKSCN
0052          TYPE 5
0053          CALL DATE(IDATE)
0054          CALL TIME(ITIME)
0055          TYPE 12,(IDATE(I),I=1,9),(ITIME(J),J=1,8)
0056          TYPE 10
0057          TYPE 15
0058          TYPE 20
0059          TYPE 22
0060          TYPE 25
0061          TYPE 30
0062          TYPE 32
0063          TYPE 35
0064        5 FORMAT(' 'T22'SODIUM LIDAR - UNIVERSITY OF ILLINOIS'/)
0065       10 FORMAT(' 'T35'MAIN MENU'/)
0066       12 FORMAT(' ',T35,9A1,/T35,8A1,//)
0067       15 FORMAT('0'T14'<1>   Experiment Parameters'/)
0068       20 FORMAT(' 'T14'<2>   Current Status'/)
0069       22 FORMAT(' 'T14'<3>   Low Altitude Alignment Run'/)
0070       25 FORMAT(' 'T14'<4>   Sodium Alignment Run'/)
0071       30 FORMAT(' 'T14'<5>   Data Run'/)
0072       32 FORMAT(' 'T14'<6>   Examine Profile on Disk'/)
0073       35 FORMAT('0'/'T5'Enter an option: '$)
      C------------------------------------------
      C  GET AND CHECK RESPONSE
      C------------------------------------------
0074          READ(5,*,ERR=3) OPT
      C  EXPERIMENT PARAMETERS
0075          IF(OPT.EQ.1) CALL EXPPAR
      C  CURRENT STATUS
0077          IF(OPT.EQ.2) CALL CSTTUS
      C  LOW-ALTITUDE ALIGNMENT RUN
0079          IF(OPT.EQ.3) CALL LAALN
      C  SODIUM ALIGNMENT RUN
```

```
FORTRAN IV      V02.5-5    Tue 31-May-83 00:58:58          PAGE 004

0081          IF(OPT.EQ.4) CALL ALNRTN
       C  DATA RUN
0083          IF(OPT.EQ.5) CALL DATRUN
       C  EXAMINE PROFILE ON DISK
0085          IF(OPT.EQ.6) CALL EXMPRF
       C  HEX DUMP OF ARRAY LDATA
0087          IF(OPT.EQ.7) CALL MOUT
0089          GOTO 3
0090          STOP
0091          END
```

```
FORTRAN IV       Storage Map for Program Unit .MAIN.

Local Variables, .PSECT $DATA, Size = 000020 (    8. words)

Name   Type   Offset      Name   Type   Offset      Name   Type   Offset
I      I*2    000006      J      I*2    000010      OPT    R*4    000012

COMMON Block /DATTIM/, Size = 000021 (    9. words)

Name   Type   Offset      Name   Type   Offset      Name   Type   Offset
IDATE  L*1    000000      ITIME  L*1    000011

COMMON Block /SETS  /, Size = 000006 (    3. words)

Name   Type   Offset      Name   Type   Offset      Name   Type   Offset
SETS   I*2    000000      PROFS  I*2    000002      SHOTSP I*2    000004

COMMON Block /LREPS /, Size = 000042 (   17. words)

Name   Type   Offset      Name   Type   Offset      Name   Type   Offset
LREP   R*4    000000      LRI    R*4    000004      HRI    R*4    000010
LBI    I*2    000014      HBI    I*2    000016      BSAT   R*4    000020
BSATF  R*4    000024      EANG   R*4    000030      REANG  R*4    000034
PDEL   I*2    000040

COMMON Block /STATS /, Size = 000010 (    4. words)

Name   Type   Offset      Name   Type   Offset      Name   Type   Offset
CSET   I*2    000000      CPROF  I*2    000002      LSET   I*2    000004
LPROF  I*2    000006

COMMON Block /CALCS /, Size = 000020 (    8. words)

Name   Type   Offset      Name   Type   Offset      Name   Type   Offset
K30    I*2    000000      K60    I*2    000002      K80    I*2    000004
K100   I*2    000006      SGP    I*2    000010      TSGP   I*2    000012
CLA    R*4    000014

COMMON Block /ALN  /, Size = 000006 (    3. words)

Name   Type   Offset      Name   Type   Offset      Name   Type   Offset
.EPR4  R*4    000000      SHOTS4 I*2    000004

COMMON Block /LAAN /, Size = 000006 (    3. words)

Name   Type   Offset      Name   Type   Offset      Name   Type   Offset
REPR3  R*4    000000      SHOTS3 I*2    000004

COMMON Block /DISK /, Size = 000023 (   10. words)

Name   Type   Offset      Name   Type   Offset      Name   Type   Offset
INTSIZ I*2    000000      RECSIZ I*2    000002      FNAME  L*1    000004
```

FORTRAN IV   Storage Map for Program Unit .MAIN.

COMMON Block /PTERS /, Size = 000006 (3. words)

Name	Type	Offset	Name	Type	Offset	Name	Type	Offset
B30	I*2	000000	B60	I*2	000002	S20	I*2	000004

COMMON Block /CONST /, Size = 000020 (6. words)

Name	Type	Offset	Name	Type	Offset	Name	Type	Offset
PI	R*4	000000	C	R*4	000004	GATTIM	R*4	000010
RG	R*4	000014						

Local and COMMON Arrays:

Name	Type	Section	Offset	------Size-----	Dimensions
FNAME	L*1	DISK	000004	000017 (8.)	(15)
IDATE	L*1	DATTIM	000000	000011 (5.)	(9)
ITIME	L*1	DATTIM	000011	000010 (4.)	(8)

Subroutines, Functions, Statement and Processor-Defined Functions:

Name	Type	Name	Type	Name	Type	Name	Type	Name	Type
ALNRTN	R*4	BAKSCN	R*4	CLRSCN	R*4	CSTTUS	R*4	DATE	R*4
DATRUN	R*4	EXMPRF	R*4	EXPPAR	R*4	INITR	I*2	IPEEK	I*2
IPOKE	I*2	LAALN	I*2	MOUT	I*2	TIME	R*4		

III.3 EXPPAR

```
FORTRAN IV     V02.5-5    Tue 31-May-83 14:50:20              PAGE 001
      C ************************************************************
      C
      C   SUBROUTINE EXPPAR - EXPERIMENT PARAMETERS
      C                    D.VOELZ
      C
      C
      C ************************************************************
      C THIS SUBROUTINE DISPLAYS THE LIDAR EXPERIMENT
      C PARAMETER VALUES AND ALLOWS THE VALUES TO BE
      C CHANGED.
      C
0001        SUBROUTINE EXPPAR
0002        INTEGER SETS,PROFS,SHOTSP,LBI,HBI,PDEL
0003        REAL LREP,LRI,HRI,EANG,C,GATTIM,RG,PI,REANG,BSAT,DENOM
0004        REAL BSATF
0005        LOGICAL*1 IDATE(9),ITIME(8)
0006        BYTE CHNG
      C
0007        COMMON /DATTIM/ IDATE,ITIME
0008        COMMON /SETS/ SETS,PROFS,SHOTSP
0009        COMMON /LREPS/ LREP,LRI,HRI,LBI,HBI,BSAT,BSATF,EANG,
           >REANG,PDEL
0010        COMMON /CONST/ PI,C,GATTIM,RG
      C
      C-------------------------------------------
      C DISPLAY PARAMETER MENU
      C-------------------------------------------
      C CLEAR SCREEN AND SET ON BACKGORUND (LO INTENSITY)
0011      5 CALL CLRSCN
0012        CALL BAKSCN
      C GET DATE AND TIME
0013        CALL DATE(IDATE)
0014        CALL TIME(ITIME)
0015        TYPE 20,(IDATE(I),I=1,9),(ITIME(J),J=1,8)
      C SET SCREEN ON FOREGROUND (HI INTENSITY)
0016        CALL FORSCN
0017        TYPE 30,SETS
0018        TYPE 35,PROFS
0019        TYPE 40,SHOTSP
0020        TYPE 45,LREP
0021        TYPE 46,PDEL
0022        TYPE 47,EANG
0023        TYPE 48,BSAT
0024        TYPE 49,BSATF
0025        TYPE 50,LRI,HRI
0026        TYPE 52,LBI,HBI
      C SET SCREEN ON BACKGROUND (LO INTENSITY)
0027        CALL BAKSCN
0028        TYPE 70
0029     20 FORMAT(' ',/,T29,'EXPERIMENT PARAMETERS',/,/T35,9A1,/T35.8A1,//)
0030     30 FORMAT(' ',T14,'<S>  #Sets ',21X,'= ',I4,'                ')
0031     35 FORMAT(' ',T14,'<P>  #Profiles Per Set ',9X,'= ',I4
           >,'              ')
0032     40 FORMAT(' ',T14,'<L>  #Laser Shots Per Profile',3X,'= ',I4
           >,'            ')
```

```
FORTRAN IV      V02.5-5    Tue 31-May-83 14:50:20           PAGE 002

0033      45 FORMAT(' ',T14,'<R>  Laser Rep Rate',13X,'=   ',F4.1,' Hz'
          ),'          ')
0034      46 FORMAT(' ',T14,'<D>  Inter-Profile Delay',8x,'= ',I4,' Sec'
          ),'          ')
0035      47 FORMAT(' ',T14,'<E>  Elevation Angle',12X,'= ',F5.1,' Degrees'
          ),'          ')
0036      48 FORMAT(' ',T14,'<B>  Base Altitude',14X,'=   ',F6.2,' KM'
          ),'          ')
0037.     49 FORMAT(' ',T14,'<F>',29X,'=  ',F7.1,' Ft',',          ')
0038      50 FORMAT(' ',T14,'<H>  Altitude Range of Data',4X,' = ',F6.2,
          +'  To ',F7.2,' KM')
0039      52 FORMAT(' ',T14,'<N>  Range Bins',16X,' =    ',I4,'  To ',I4
          ),'          ')
0040      70 FORMAT('0',///,T5,'To change any parameter value first enter th
          +e corresponding',/T5,'LETTER and then RETURN ',$)
      C----------------------------------------
      C  GET RESPONSE
      C----------------------------------------
0041         READ(5,80,ERR=5) CHNG
0042      80 FORMAT(A1)
      C----------------------------------------
      C  CHECK RESPONSE AND CHANGE VALUES
      C----------------------------------------
0043         CALL FORSCN
      C  CHANGE SETS
0044      81 IF(CHNG.NE.'S') GOTO 90
0046         TYPE 85
0047      85    FORMAT('0',T14,'#Sets = ',$)
0048            READ(5,*,ERR=200)SETS
0049            IF(SETS.LE.0) GOTO 200
0051            GOTO 5
      C  CHANGE PROFILES PER SET
0052      90 IF(CHNG.NE.'P') GOTO 100
0054      92    TYPE 93
0055      93    FORMAT('0',T14,'#Profiles Per Set = ',$)
0056            READ(5,*,ERR=200)PROFS
0057            IF(PROFS.LE.0) GOTO 200
0059            GOTO 5
      C  CHANGE LASER SHOTS PER PROFILE
0060     100 IF(CHNG.NE.'L') GOTO 110
0062     102    TYPE 103
0063     103    FORMAT('0',T14,'#Laser Shots Per Profile = ',$)
0064            READ(5,*,ERR=200)SHOTSP
0065            IF(SHOTSP.LE.0)GOTO 200
0067            GOTO 5
      C  CHANGE LASER REP RATE
0068     110 IF(CHNG.NE.'R') GOTO 115
0070     112    TYPE 113
0071     113    FORMAT('0',T14,'Laser Rep Rate = ',$)
0072            READ(5,*,ERR=200)LREP
0073            IF((LREP.LE.0.0).OR.(LREP.GT.10.0)) GOTO 200
0075            GOTO 5
      C  CHANGE INTER-PROFILE DELAY
0076     115 IF(CHNG.NE.'D')GOTO 120
```

```
0078    117   TYPE 113
0079    118   FORMAT('0',T14,'Inter-Profile Delay = ',$)
0080          READ(5,*,ERR=200) PDEL
0081          IF(PDEL.LT.0)GOTO 200
0083          GOTO 5
      C  CHANGE ELEVATION ANGLE
0084    120   IF(CHNG.NE.'E')GOTO 130
0086    122   TYPE 123
0087    123   FORMAT('0',T14,'Elevation Angle = ',$)
0088          READ(5,*,ERR=200)EANG
0089          IF((EANG.LE.0).OR.(EANG.GT.90)) GOTO 200
0091          REANG=(PI/180.)*EANG
0092          GOTO 300
      C  CHANGE BASE ALTITUDE
0093    130   IF(CHNG.NE.'B')GOTO 140
0095    132   TYPE 133
0096    133   FORMAT('0',T14,'Base Altitude (KM) = ',$)
0097          READ(5,*,ERR=200)BSAT
0098          BSATF=3281.*BSAT
0099          GOTO 300
0100    140   IF(CHNG.NE.'F')GOTO 150
0102    142   TYPE 143
0103    143   FORMAT('0',T14,'Base Altitude (Feet)= ',$)
0104          READ(5,*,ERR=200)BSATF
0105          BSAT=.0003048*BSATF
0106          GOTO 300
      C  CHANGE ALTITUDE RANGE OF DATA
0107    150   IF(CHNG.NE.'H') GOTO 160
0109    152   TYPE 153
0110    153   FORMAT('0',T14,'Altitude Range of Data = ',$)
0111          READ(5,*,ERR=200)LRI
0112          IF(LRI.LT.BSAT) GOTO 200
0114          TYPE 157
0115    157   FORMAT('+',T34,'To    ',$)
0116          READ(5,*,ERR=200)HRI
0117          IF(HRI.LE.LRI) GOTO 200
0119          GOTO 290
      C  CHANGE RANGE BINS
0120    160   IF(CHNG.NE.'N') RETURN
0122    162   TYPE 163
0123    163   FORMAT('0',T14,'Range Bins = ',$)
0124          READ(5,*,ERR=200)LBI
0125          TYPE 167
0126    167   FORMAT('+',T22,'To    ',$)
0127          READ(5,*,ERR=200)HBI
0128          IF(HBI.LE.LBI) GOTO 200
0130          REANG=(PI/180.)*EANG
0131          GOTO 295
      C---------------------------------------------
      C  ERROR MESSAGES
      C---------------------------------------------
0132    200 TYPE 210
0133    210 FORMAT(' ',T7,'?? PARAMETER RANGE ERROR')
0134        GOTO 81
```

FORTRAN IV   V02.5-5   Tue 31-May-83 14:50:20   PAGE 004

```
        C------------------------------------------
        C  RANGE CALCULATIONS
        C------------------------------------------
0135     290 REANG=(PI/180.)*EANG
0136         DENOM=SIN(REANG)*RG
0137         LBI=IFIX((LRI+(RG/2.)-BSAT)/DENOM)
0138         HBI=IFIX((HRI+(1.5*RG)-BSAT)/DENOM)
0139     295 IF(HBI.GT.2000)HBI=2000
0141         IF(LBI.LT.1)LBI=1
0143     300 LRI=FLOAT(((LBI*RG)-(RG/2.))*SIN(REANG)+BSAT)
0144         HRI=FLOAT(((HBI*RG)-(RG/2.))*SIN(REANG)+BSAT)
0145         GOTO 5
        C
0146         STOP
0147         END
```

```
FORTRAN IV        Storage Map for Program Unit EXPPAR

Local Variables, .PSECT $DATA, Size = 000040 (   16. words)

Name   Type  Offset       Name   Type  Offset       Name   Type  Offset
CHNG   L*1   000012       DENOM  R*4   000006        I      I*2   000014
J      I*2   000016

COMMON Block /DATTIM/, Size = 000021 (   9. words)

Name   Type  Offset       Name   Type  Offset       Name   Type  Offset
IDATE  L*1   000000       ITIME  L*1   000011

COMMON Block /SETS  /, Size = 000006 (   3. words)

Name   Type  Offset       Name   Type  Offset       Name   Type  Offset
SETS   I*2   000000       PROFS  I*2   000002        SHOTSP I*2   000004

COMMON Block /LREPS /, Size = 000042 (   17. words)

Name   Type  Offset       Name   Type  Offset       Name   Type  Offset
LREP   R*4   000000       LRI    R*4   000004        HRI    R*4   000010
LBI    I*2   000014       HBI    I*2   000016        BSAT   R*4   000020
BSATF  R*4   000024       EANG   R*4   000030        REANG  R*4   000034
PDEL   I*2   000040

COMMON Block /CONST /, Size = 000020 (   8. words)

Name   Type  Offset       Name   Type  Offset       Name   Type  Offset
PI     R*4   000000       C      R*4   000004        GATTIM R*4   000010
RG     R*4   000014

Local and COMMON Arrays:

Name      Type     Section Offset  ------Size----- Dimensions
IDATE     L*1      DATTIM  000000  000011 (    5.) (9)
ITIME     L*1      DATTIM  000011  000010 (    4.) (8)

Subroutines, Functions, Statement and Processor-Defined Functions:

Name   Type   Name   Type   Name   Type   Name   Type   Name   Type
BAKSCN R*4    CLRSCN R*4    DATE   R*4    FLOAT  R*4    FORSCN R*4
IFIX   I*2    SIN    R*4    TIME   R*4
```

ORIGINAL PAGE IS
OF POOR QUALITY

III.4   CSTTUS

FORTRAN IV   V02.5-5   Tue 31-May-83 14:56:02   PAGE 001

```
      C  ******************************************************
      C
      C       SUBROUTINE CSTTUS - CURRENT STATUS
      C                    D.VOELZ
      C
      C  ******************************************************
      C
      C
      C    THIS SUBROUTINE DISPLAYS THE CURRENT STATUS OF THE EXPERIMENT.
      C    ALSO DISPLAYED ARE THE CHARACTERISTIC DATA VALUES FOR THE LAST
      C    RECORDED PROFILE.  THE SUBROUTINE ALLOWS THE VALUES OF THE
      C    CURRENT SCAN AND PROFILE TO BE CHANGED.
      C
0001          SUBROUTINE CSTTUS
0002          INTEGER SETS,PROFS,SHOTSP,CSET,CPROF,LSET,LPROF,CHNG
             >,K30,K60,K80,K100,SGP,TSGP
0003          REAL EANG,CLA
0004          LOGICAL*1 IDATE(9),ITIME(8)
      C
0005          COMMON /DATTIM/ IDATE,ITIME
0006          COMMON /SETS/ SETS,PROFS,SHOTSP
0007          COMMON /STATS/ CSET,CPROF,LSET,LPROF
0008          COMMON /CALCS/ K30,K60,K80,K100,SGP,TSGP,CLA
      C
      C
      C    DISPLAY CURRENT STATUS MENU----------------------------------
      C    ------------------------------------------------------------
0009      5   CALL CLRSCN
0010          CALL BAKSCN
0011          TYPE 30
0012          CALL DATE(IDATE)
0013          TYPE 35,(IDATE(I),I=1,9),SETS
0014          CALL TIME(ITIME)
0015          TYPE 40,(ITIME(J),J=1,8)
0016          CALL FORSCN
0017          TYPE 45,CSET
0018          CALL BAKSCN
0019          TYPE 50,PROFS
0020          CALL FORSCN
0021          TYPE 55,CPROF
0022          CALL BAKSCN
0023          TYPE 60
0024          TYPE 65,LSET,SHOTSP
0025          TYPE 70,LPROF
0026          TYPE 75,K30,K60,K80,K100
0027          TYPE 95,TSGP
0028          TYPE 100,SGP
0029          TYPE 105,CLA
0030          TYPE 110
      C
0031      30 FORMAT('+',T2,'CURRENT STATUS -------------------------------
             +--------------')
0032      35 FORMAT('0',T5,9A1,21X,'Desired # Sets ',11X,'=',I6)
0033      40 FORMAT(' ',T5,8A1,$)
```

```
FORTRAN IV      VO2.5-5    Tue 31-May-83 14:56:02          PAGE 002

0034     45 FORMAT('+',T20,'<CS>  Current Set ',12X,'=',I6)
0035     50 FORMAT(' ',T35,'Profiles Per Set ',9X,'=',I6)
0036     55 FORMAT(' ',T31,'<CP>  Current Profile',9X,'=',I6)
0037     60 FORMAT('0',T2,'LAST RECORDED PROFILE  -----------------------
           +------------------')
0038     65 FORMAT('0',T5,'Set #       ',I4,16X,'Laser Shots Per Profile  ='
           +,I6)
0039     70 FORMAT(' ',T5,'Profile # ',I4,16X,'Detected Photons:')
0040     75 FORMAT(' ',T50,'at 30 Km',3X,'=',I6,/T47,'60 -  80 Km',3X,
           +'=',I6,/T47,'80 - 100 Km',3X,'=',I6,/T46,'100 - 120 Km',3X,
           +'=',I6)
0041     95 FORMAT(' ',T35,'Total Signal Photons      =',I6)
0042    100 FORMAT(' ',T35,'Signal Photons Per Shot   =',I6)
0043    105 FORMAT(' ',T35,'Column Abundance Ratio    =',F6.2)
0044    110 FORMAT('0',T2,'To change the Current Set or Profile values first
           + enter the',/T2,'corresponding LETTERS then RETURN  ',$)
       C  ============================================================
       C  GET RESPONSE
       C  ------------------------------------------------------------
0045        READ(5,150,ERR=5) CHNG
0046    150 FORMAT(A2)
       C  ------------------------------------------------------------
       C  CHECK RESPONSE AND CHANGE VALUES
       C  ------------------------------------------------------------
0047        CALL FORSCN
       C  CHANGE CURRENT SET
0048    155 IF(CHNG.NE.'CS') GOTO 170
0050    160   TYPE 162
0051    162   FORMAT('0',T14,'Current Set =  ',$)
0052          READ(5,*,ERR=200)CSET
0053          IF(CSET.LE.0)GOTO 200
0055          GOTO 5
       C  CHANGE CURRENT PROFILE
0056    170 IF(CHNG.NE.'CP')RETURN
0058    172 TYPE 173
0059    173 FORMAT('0',T14,'Current Profile =  ',$)
0060        READ(5,*,ERR=200) CPROF
0061        IF(CPROF.LE.0.OR.CPROF.GT.PROFS)GOTO 200
0063        GOTO 5
       C  ------------------------------------------------------------
       C  ERROR MESSAGES
       C  ------------------------------------------------------------
0064    200 TYPE 210
0065    210 FORMAT(' ',T7,'?? PARAMETER RANGE ERROR')
0066        GOTO 155
0067        STOP
0068        END
```

```
FORTRAN IV        Storage Map for Program Unit CSTTUS

Local Variables, .PSECT $DATA, Size = 000020 (    8. words)

Name    Type    Offset      Name    Type    Offset      Name    Type    Offset
CHNG    I*2     000006      EANG    R*4     000010      I       I*2     000014
J       I*2     000016

COMMON Block /DATTIM/, Size = 000021 (    9. words)

Name    Type    Offset      Name    Type    Offset      Name    Type    Offset
IDATE   L*1     000000      ITIME   L*1     000011

COMMON Block /SETS  /, Size = 000006 (    3. words)

Name    Type    Offset      Name    Type    Offset      Name    Type    Offset
SETS    I*2     000000      PROFS   I*2     000002      SHOTSP  I*2     000004

COMMON Block /STATS /, Size = 000010 (    4. words)

Name    Type    Offset      Name    Type    Offset      Name    Type    Offset
CSET    I*2     000000      CPROF   I*2     000002      LSET    I*2     000004
LPROF   I*2     000006

COMMON Block /CALCS /, Size = 000020 (    8. words)

Name    Type    Offset      Name    Type    Offset      Name    Type    Offset
K30     I*2     000000      K60     I*2     000002      K80     I*2     000004
K100    I*2     000006      SGP     I*2     000010      TSGP    I*2     000012
CLA     R*4     000014

Local and COMMON Arrays:

Name      Type      Section Offset    ------Size-----  Dimensions
IDATE     L*1       DATTIM  000000    000011 (    5.)  (9)
ITIME     L*1       DATTIM  000011    000010 (    4.)  (8)

Subroutines, Functions, Statement and Processor-Defined Functions:

Name    Type    Name    Type    Name    Type    Name    Type    Name    Type
BAKSCN  R*4     CLRSCN  R*4     DATE    R*4     FORSCN  R*4     TIME    R*4
```

III.5 LAALN

FORTRAN IV   V02.5-5   Tue 31-May-83 15:00:24   PAGE 001

```
      C ******************************************
      C
      C
      C   SUBROUTINE LAALN.FOR - LOW ALTITUDE ALIGNMENT ROUTINE
      C                      D.VOELZ
      C
      C
      C ******************************************
      C
      C   THIS SUBROUTINE INITIATES A LOW ALTITUDE ALIGNMENT DATA
      C   RUN.
      C
0001        SUBROUTINE LAALN
0002        REAL REPR3
0003        INTEGER SHOTS3,MSHOTS,REPNUM,BCNT,BNMBR,LDATA(2100)
0004        LOGICAL*1 IDATE(9),ITIME(8)
0005        BYTE RUNFLG,DFLAG,ERRFLG,MON,JCHAR,OPT
      C
0006        COMMON/DATTIM/ IDATE,ITIME
0007        COMMON/LAAN/ REPR3,SHOTS3
0008        COMMON/FLAGS/ RUNFLG,DFLAG,ERRFLG
0009        COMMON/SEND/ REPNUM,MSHOTS
0010        COMMON/BUFF/ LDATA
      C
      C   TYPE LOW ALTITUDE ALIGNMENT ROUTINE SCREEN
      C
0011      5 CALL CLRSCN
0012        CALL BAKSCN
0013        CALL DATE(IDATE)
0014        CALL TIME(ITIME)
0015        TYPE 20,(IDATE(I),I=1,9),(ITIME(J),J=1,8)
0016        CALL FORSCN
0017        TYPE 22,SHOTS3
0018        TYPE 24,REPR3
0019        CALL BAKSCN
0020        TYPE 26
0021        TYPE 30
0022     20 FORMAT(' ',/,T25,'LOW ALTITUDE ALIGNMENT ROUTINE',/,/T35
          >,9A1,/T35,8A1,//)
0023     22 FORMAT(' ',T10,'<L>  Laser Shots Per Profile  = ',I4
          >,'   ')
0024     24 FORMAT(' ',T10,'<R>  Laser Rep Rate          = ',F4.1
          +,' Hz',' ')
0025     26 FORMAT(' ',//,T5,'To change any parameter value enter the corres
          +ponding LETTER',/T5,'and then RETURN')
0026     30 FORMAT(' ',//,T5,'To begin the alignment routine press "Y"
          + then RETURN  ',$)
      C ========= =============================
      C   GET & CHECK RESPONSE
      C =====================================
0027        READ(5,80,ERR=5) OPT
0028     80 FORMAT(A1)
      C
0029        CALL FORSCN
      C CHANGE LASER SHOTS PER PROFILE
0030     81 IF(OPT.NE.'L') GOTO 90
```

```
0032          TYPE 85
0033       85 FORMAT('0',T10,'Laser Shots Per Profile = ',$)
0034          READ(5,*,ERR=200)SHOTS3
0035           IF(SHOTS3.LE.0)GOTO 200
0037          GOTO 5
       C  CHANGE LASER REP RATE
0038       90 IF(OPT.NE.'R') GOTO 100
0040          TYPE 95
0041       95 FORMAT('0',T10,'Laser Rep Rate = ',$)
0042          READ(5,*,ERR=200)REPR3
0043           IF((REPR3.LE.0.0).OR.(REPR3.GT.10.0)) GOTO 200
0045          GOTO 5
       C  =======================================
       C  LOW ALTITUDE ALIGNMENT RUN
       C  =======================================
0046      100 IF(OPT.NE.'Y') RETURN
0048          CALL BAKSCN
       C
       C  PARAMETER CALCULATIONS
       C
0049          MSHOTS=SHOTS3
0050          REPNUM=IFIX((SQRT((1.02046*1.E6/REPR3-57251.)*4./15.+
          >(5.4*5.4))-5.4)/2.+.5)
       C
       C  INIT FLAGS & START PROFILE RUN
       C
0051      120 DFLAG=0
0052          ERRFLG=0
0053          CALL SNDCOM
       C
       C  GET DATE AND TIME
       C
0054          CALL TIME(ITIME)
0055          CALL DATE(IDATE)
       C
       C  WAIT FOR PROFILE
       C
0056      130 ICHAR=ITTINR()
0057          JCHAR=ICHAR
0058          IF(JCHAR.EQ.'S') GOTO 180
0060          IF(DFLAG.NE.1)GOTO 130
0062          IF(ERRFLG.EQ.1)GOTO 180
       C
       C  DISPLAY
       C
0064          DO 160 K=1,3
0065          TYPE 135,(IDATE(I),I=1,9),K,(ITIME(I),I=1,8)
0066      135 FORMAT(' ',T5,9A1,25X,I2,/T5,8A1)
0067          BCNT=1+200*(K-1)
0068          DO 150 J=1,20
0069          BNMBR=BCNT+10*(J-1)
0070          TYPE 140,BNMBR,(LDATA(I),I=BNMBR,BNMBR+9)
0071      140 FORMAT(' ',I5,'-',10I6)
0072      150 CONTINUE
```

```
0073          PAUSE
0074      160 CONTINUE
0075      163 TYPE 165
0076      165 FORMAT(' ',//,T19,'Do you want to run the Routine
         > again ? ',$)
0077          READ(5,170,ERR=163)MON
0078      170 FORMAT(A1)
0079          IF(MON.EQ.'Y')GOTO 120
          C
0081      180 RETURN
          C
          C---------------------------------------
          C  ERROR MESSAGES
          C---------------------------------------
0082      200 TYPE 210
0083      210 FORMAT(' ',T7,'?? PARAMETER RANGE ERROR')
0084          GOTO 81
          C
0085          STOP
0086          END
```

```
FORTRAN IV      Storage Map for Program Unit LAALN

Local Variables, .PSECT $DATA, Size = 000046 (   19. words)

Name   Type  Offset    Name   Type  Offset    Name   Type  Offset
BCNT   I*2   000014    BNMBR  I*2   000016    I      I*2   000024
ICHAR  I*2   000030    J      I*2   000026    JCHAR  L*1   000021
K      I*2   000032    MON    L*1   000020    OPT    L*1   000022

COMMON Block /DATTIM/, Size = 000021 (   9. words)

Name   Type  Offset    Name   Type  Offset    Name   Type  Offset
IDATE  L*1   000000    ITIME  L*1   000011

COMMON Block /LAAN  /, Size = 000006 (   3. words)

Name   Type  Offset    Name   Type  Offset    Name   Type  Offset
REPR3  R*4   000000    SHOTS3 I*2   000004

COMMON Block /FLAGS /, Size = 000003 (   2. words)

Name   Type  Offset    Name   Type  Offset    Name   Type  Offset
RUNFLG L*1   000000    DFLAG  L*1   000001    ERRFLG L*1   000002

COMMON Block /SEND  /, Size = 000004 (   2. words)

Name   Type  Offset    Name   Type  Offset    Name   Type  Offset
REPNUM I*2   000000    MSHOTS I*2   000002

COMMON Block /BUFF  /, Size = 010150 ( 2100. words)

Name   Type  Offset    Name   Type  Offset    Name   Type  Offset
LDATA  I*2   000000

Local and COMMON Arrays:

Name    Type   Section Offset  ------Size----- Dimensions
IDATE   L*1    DATTIM  000000  000011 (   5.) (9)
ITIME   L*1    DATTIM  000011  000010 (   4.) (8)
LDATA   I*2    BUFF    000000  010150 ( 2100.) (2100)

Subroutines, Functions, Statement and Processor-Defined Functions:

Name   Type  Name   Type  Name   Type  Name   Type  Name   Type
BAKSCN R*4   CLRSCN R*4   DATE   R*4   FORSCN R*4   IFIX   I*2
ITTINR I*2   SNDCOM R*4   SQRT   R*4   TIME   R*4
```

III.6   ALNRTN

```
      C *******************************************
      C
      C
      C   SUBROUTINE ALNRTN.FOR - SODIUM ALIGNMENT ROUTINE
      C                    D.VOELZ
      C
      C   *******************************************
      C
      C
      C   THIS SUBROUTINE INITIATES A SODIUM ALIGNMENT DATA
      C   RUN.
      C
0001        SUBROUTINE ALNRTN
0002        REAL REPR4,LREP,LRI,HRI,BSAT,EANG,PI,C,GATTIM,RG,REANG
           >,DENOM,BSATF
0003        INTEGER REPNUM,SHOTS4,MSHOTS,LBI,HBI,INTSIZ,RECSIZ,B30
           >,B60,S20,PDEL
0004        LOGICAL*1 IDATE(9),ITIME(8)
0005        BYTE RUNFLG,DFLAG,ERRFLG,MON,JCHAR,OPT
      C
0006        COMMON/PTERS/ B30,B60,S20
0007        COMMON/LREPS/ LREP,LRI,HRI,LBI,HBI,BSAT,BSATF,EANG,
           >REANG,PDEL
0008        COMMON/DATTIM/ IDATE,ITIME
0009        COMMON/ALN/ REPR4,SHOTS4
0010        COMMON/FLAGS/ RUNFLG,DFLAG,ERRFLG
0011        COMMON/SEND/ REPNUM,MSHOTS
0012        COMMON/CONST/ PI,C,GATTIM,RG
      C
      C   TYPE SODIUM ALIGNMENT ROUTINE SCREEN
      C
0013      5 CALL CLRSCN
0014        CALL BAKSCN
0015        CALL DATE(IDATE)
0016        CALL TIME(ITIME)
0017        TYPE 20,(IDATE(I),I=1,9),(ITIME(J),J=1,8)
0018        CALL FORSCN
0019        TYPE 22,SHOTS4
0020        TYPE 24,REPR4
0021        CALL BAKSCN
0022        TYPE 26
0023        TYPE 30
0024     20 FORMAT(' ',/,T28,'SODIUM ALIGNMENT ROUTINE',/,/T35,9A1,/T35
           >,8A1,//)
0025     22 FORMAT(' ',T10,'<L> Laser Shots Per Profile = ',I4
           >,'    ')
0026     24 FORMAT(' ',T10,'<R> Laser Rep Rate          = ',F4.1
           +,' Hz',' ')
0027     26 FORMAT(' ',//,T5,'To change any parameter value enter the corres
           +ponding LETTER',/T5,'and then RETURN')
0028     30 FORMAT(' ',//,T5,'To begin the alignment routine press "Y"
           + then RETURN  ',$)
      C ==========================================
      C   GET & CHECK RESPONSE
      C ==========================================
0029        READ(5,80,ERR=5) OPT
```

```
FORTRAN IV      V02.5-5    Tue 31-May-83 15:03:26          PAGE 002

0030      80 FORMAT(A1)
        C
0031         CALL FORSCN
        C CHANGE LASER SHOTS PER PROFILE
0032      81 IF(OPT.NE.'L') GOTO 90
0034         TYPE 85
0035      85 FORMAT('0',T10,'Laser Shots Per Profile = ',$)
0036         READ(5,*,ERR=200)SHOTS4
0037          IF(SHOTS4.LE.0)GOTO 200
0039         GOTO 5
        C CHANGE LASER REP RATE
0040      90 IF(OPT.NE.'R') GOTO 100
0042         TYPE 95
0043      95 FORMAT('0',T10,'Laser Rep Rate = ',$)
0044         READ(5,*,ERR=200)REPR4
0045          IF((REPR4.LE.0.0).OR.(REPR4.GT.10.0)) GOTO 200
0047         GOTO 5
        C =======================================
        C  SODIUM ALIGNMENT RUN
        C =======================================
0048     100 IF(OPT.NE.'Y') RETURN
0050         CALL BAKSCN
0051         RUNFLG=3
        C
        C  PARAMETER CALCULATIONS
        C
0052         DENOM=SIN(REANG)*RG
0053         B30=IFIX((30.+(RG/2)-BSAT)/DENOM+.5)
0054         B60=IFIX((60.+(RG/2)-BSAT)/DENOM+.5)
0055         S20=IFIX((20.+(RG/2))/DENOM+.5)
0056         MSHOTS=SHOTS4
0057         REPNUM=IFIX((SQRT((1.02046*1.E6/REPR4-57251.)*4./15.+
            >(5.4*5.4))-5.4)/2.+.5)
        C
        C  INIT FLAGS & START PROFILE RUN
        C
0058     120 DFLAG=0
0059         ERRFLG=0
0060         CALL SNDCOM
        C
        C  GET DATE AND TIME
        C
0061         CALL TIME(ITIME)
0062         CALL DATE(IDATE)
        C
        C  WAIT FOR PROFILE
        C
0063     130 ICHAR=ITTINR()
0064         JCHAR=ICHAR
0065         IF(JCHAR.EQ.'S') GOTO 160
0067         IF(DFLAG.NE.1)GOTO 130
0069         IF(ERRFLG.EQ.1)GOTO 160
0071         CALL DISCAL
        C
```

```
FORTRAN IV      V02.5-5    Tue 31-May-83 15:03:26        PAGE 003
0072     135 TYPE 140
0073     140 FORMAT(' ',//,T19,'Do you want to run the Routine
         > again ? ',$)
0074         READ(5,145,ERR=135)MON
0075     145 FORMAT(A1)
0076         IF(MON.EQ.'Y')GOTO 120
      C
0078     160 RETURN
      C
      C----------------------------------------
      C  ERROR MESSAGES
      C---------  -----------------------------
0079     200 TYPE 210
0080     210 FORMAT(' ',T7,'?? PARAMETER RANGE ERROR')
0081         GOTO 81
      C
0082         STOP
0083         END
```

```
FORTRAN IV      Storage Map for Program Unit ALNRTN

Local Variables, .PSECT $DATA, Size = 000060 (   24. words)

Name   Type   Offset      Name   Type   Offset      Name   Type   Offset
DENOM  R*4    000006      I      I*2    000022      ICHAR  I*2    000026
INTSIZ I*2    000012      J      I*2    000024      JCHAR  L*1    000017
MON    L*1    000016      OPT    L*1    000020      RECSIZ I*2    000014

COMMON Block /PTERS /, Size = 000006 (    3. words)

Name   Type   Offset      Name   Type   Offset      Name   Type   Offset
B30    I*2    000000      B60    I*2    000002      S20    I*2    000004

COMMON Block /LREPS /, Size = 000042 (   17. words)

Name   Type   Offset      Name   Type   Offset      Name   Type   Offset
LREP   R*4    000000      LRI    R*4    000004      HRI    R*4    000010
LBI    I*2    000014      HBI    I*2    000016      BSAT   R*4    000020
BSATF  R*4    000024      EANG   R*4    000030      REANG  R*4    000034
PDEL   I*2    000040

COMMON Block /DATTIM/, Size = 000021 (    9. words)

Name   Type   Offset      Name   Type   Offset      Name   Type   Offset
IDATE  L*1    000000      ITIME  L*1    000011

COMMON Block /ALN   /, Size = 000006 (    3. words)

Name   Type   Offset      Name   Type   Offset      Name   Type   Offset
REPR4  R*4    000000      SHOTS4 I*2    000004

COMMON Block /FLAGS /, Size = 000003 (    2. words)

Name   Type   Offset      Name   Type   Offset      Name   Type   Offset
RUNFLG L*1    000000      DFLAG  L*1    000001      ERRFLG L*1    000002

COMMON Block /SEND  /, Size = 000004 (    2. words)

Name   Type   Offset      Name   Type   Offset      Name   Type   Offset
REPNUM I*2    000000      MSHOTS I*2    000002

COMMON Block /CONST /, Size = 000020 (    8. words)

Name   Type   Offset      Name   Type   Offset      Name   Type   Offset
PI     R*4    000000      C      R*4    000004      GATTIM R*4    000010
RG     R*4    000014

Local and COMMON Arrays:

Name      Type    Section Offset   ------Size----- Dimensions
IDATE  L*1        DATTIM  000000   000011 (   5.)  (9)
ITIME  L*1        DATTIM  000011   000010 (   4.)  (8)
```

FORTRAN IV   Storage Map for Program Unit ALNRTN

Subroutines, Functions, Statement and Processor-Defined Functions:

Name	Type	Name	Type	Name	Type	Name	Type	Name	Type
BAKSCN	R*4	CLRSCN	R*4	DATE	R*4	DISCAL	R*4	FORSCN	R*4
IFIX	I*2	ITTINR	I*2	SIN	R*4	SNDCOM	R*4	SQRT	R*4
TIME	R*4								

ORIGINAL PAGE IS
OF POOR QUALITY

III.7 DATRUN

FORTRAN IV   V02.5-5   Tue 31-May-83 15:06:58   PAGE 001

```
      C *******************************************************
      C
      C    SUBROUTINE DATRUN - DATA RUN
      C              D.VOELZ
      C
      C *******************************************************
      C
      C   THIS SUBROUTINE INITIATES A DATA RUN (WHICH CONSISTS
      C   OF A SET OF PROFILES).
      C
0001        SUBROUTINE DATRUN
0002        LOGICAL*1 IIDATE(9),ITIME(8)
0003        REAL LREP,LRI,HRI,BSAT,EANG,C,GATTIM,RG,PI,REANG,DENOM,BSATF
0004        INTEGER SETS,PROFS,SHOTSP,CSET,CPROF,LSET,LPROF,HBI,LBI
           >,INTSIZ,RECSIZ,B30,B60,S20,REPNUM,MSHOTS,MN,DAY,YR,HDRB
           >,BINS,LDATA(2100),PDEL
0005        INTEGER*4 TIM2,TIML,TIMH,STIM,DTIM
0006        BYTE OPT,MON,RUNFLG,FNAME(15),EXT(3),DFLAG,ERRFLG
           >,JCHAR
      C
0007        COMMON /DATTIM/ IIDATE,ITIME
0008        COMMON /DISK/ INTSIZ,RECSIZ,FNAME
0009        COMMON /PTERS/ B30,B60,S20
0010        COMMON /LREPS/ LREP,LRI,HRI,LBI,HBI,BSAT,BSATF,EANG,
           >REANG,PDEL
0011        COMMON /SETS/ SETS,PROFS,SHOTSP
0012        COMMON /FLAGS/ RUNFLG,DFLAG,ERRFLG
0013        COMMON /SEND/ REPNUM,MSHOTS
0014        COMMON /STATS/ CSET,CPROF,LSET,LPROF
0015        COMMON/CONST/ PI,C,GATTIM,RG
0016        COMMON/BUFF/LDATA
      C
0017        HDRB=9
0018        BINS=HBI-LBI+1
0019        RECSIZ=(BINS+HDRB+1)/2
0020        INTSIZ=IFIX(FLOAT(PROFS)*(BINS+HDRB)/256.+1.1)
      C
      C  TYPE DATA RUN SCREEN
      C
0021      5 CALL CLRSCN
0022        CALL DATE(IIDATE)
0023        CALL TIME(ITIME)
0024        TYPE 20,(IIDATE(I),I=1,9),(ITIME(J),J=1,8)
0025        TYPE 25,RECSIZ,INTSIZ
0026        TYPE 30
0027     20 FORMAT(' ',/,T35,'DATA RUN',/,/T35,9A1,/T35,8A1,//)
0028     25 FORMAT(' ',/,T5,'Disk Parameters:',/T8,'Recordsize    = ',I4
           >,/T8,'Set Blocksize = ',I4,/)
0029     30 FORMAT(' ',//,T5,'To begin a Data Run press "Y" then RETURN  '
           >,$)
      C ==========================================
      C  GET & CHECK RESPONSE
      C ==========================================
0030        READ(5,80,ERR=5) OPT
```

```
FORTRAN IV      VO2.5-5    Tue 31-May-83 15:06:58          PAGE 002

0031      80 FORMAT(A1)
0032         IF(OPT.NE.'Y') RETURN
      C ========================================
      C  DATA RUN
      C ========================================
0034         RUNFLG=1
      C
      C  UPDATE SET FILE NAME
      C
0035         ENCODE(3,100,EXT)CSET
0036     100 FORMAT(I3)
0037         DO 105 I=1,3
0038         FNAME(7+I)='0'
0039     105 IF(EXT(I).NE.' ')FNAME(7+I)=EXT(I)
      C
      C  PARAMETER CALCULATIONS
      C
0041         CALL JICVT(PDEL,DTIM)
0042         DTIM=60*DTIM
0043         DENOM=SIN(REANG)*RG
0044         B30=IFIX((30.+(RG/2.)-BSAT)/DENOM+.5)
0045         B60=IFIX((60.+(RG/2.)-BSAT)/DENOM+.5)
0046         S20=IFIX((20.+(RG/2.))/DENOM+.5)
0047         MSHOTS=SHOTSP
0048         REPNUM=IFIX((SQRT((1.02046*1.0E6/LREP-57251.)*4./15.+
             >(5.4*5.4))-5.4)/2.+.5)
      C -------------------------------------------
      C  START DATA RUN
      C -------------------------------------------
0049         LSET=CSET
0050         PRINT 115,CSET,(IIDATE(I),I=1,9),(ITIME(I),I=1,8)
0051     115 FORMAT(' ',T14,'SET # ',I4,11X,9A1,11X,8A1)
0052         REWIND 6
0053         OPEN(UNIT=3,TYPE='NEW',INITIALSIZE=INTSIZ,RECORDSIZE=RECSIZ
             >,NAME=FNAME,ACCESS='DIRECT',FORM='UNFORMATTED')
      C
      C  INIT FLAGS & START PROFILE RUN
      C
0054     120 DFLAG=0
0055         EPRFLG=0
0056         CALL SNDCOM
      C
      C  GET TIME AND DATE
      C
0057     125 CALL GTIM(TIM2)
0058         CALL IDATE(MN,DAY,YR)
0059         CALL TIMASC(TIM2,ITIME)
0060         CALL DATE(IIDATE)
      C
      C  WAIT FOR PROFILE
      C
0061     130 ICHAR=ITTINR()
0062         JCHAR=ICHAR
0063         IF(JCHAR.EQ.'S') GOTO 195
```

FORTRAN IV   V02.5-5   Tue 31-May-83 15:06:58   PAGE 003

```
0065          IF(DFLAG.NE.1)GOTO 130
0067          IF(ERRFLG.EQ.1)GOTO 200
0069          CALL DISCAL
0070          WRITE(3'CPROF) BINS,LBI,CSET,CPROF,MN,DAY,YR,TIM2
             >,(LDATA(K),K=LBI,HBI)
      C
      C   UPDATE & CHECK COUNTERS
      C
0071          LPROF=CPROF
0072          CPROF=CPROF+1
0073          IF(CPROF.GT.PROFS)GOTO 140
      C
      C   INTER-PROFILE DELAY
      C
0075          CALL GTIM(TIML)
0076          CALL JJCVT(TIML)
0077      135 CALL GTIM(TIMH)
0078          CALL JJCVT(TIMH)
0079          CALL JSUB(TIMH,TIML,STIM)
0080          IF(STIM.LT.0)TIM2=0
0082          IF(DTIM.GT.STIM)GOTO 135
      C
0084          DFLAG=0
0085          ERRFLG=0
0086          CALL SNDDTR
0087          GOTO 125
      C ------------------------------------------
      C   SET COMPLETED
      C ------------------------------------------
0088      140 CLOSE(UNIT=3)
0089          CSET=CSET+1
0090          CPROF=1
0091          CALL FORSCN
0092          TYPE 150,LSET
0093          CALL BAKSCN
0094          PAUSE 'Press  RETURN to continue'
0095      145 RETURN
0096      150 FORMAT(' ',//,' SET # ',I4,' -- COMPLETED')
      C =======================================
      C   PROFILE ERROR
      C =======================================
      C   CLEAR 'S' CHARACTER FROM KEYBOARD
0097      195 READ(5,80) OPT
      C   ERROR ROUTINE
0098      200 CALL CLRSCN
0099          TYPE 210
0100      210 FORMAT(' ',//////,T11,'Do you want to close out the current
             > Data Set on disk ? ',$)
0101          READ(5,80) OPT
0102          IF(OPT.EQ.'Y') GOTO 140
0104          TYPE 220
0105      220 FORMAT(' ',//,T27,'Press RETURN to try again ',$)
0106          READ(5,80) OPT
0107          GOTO 120
0108          END
```

```
FORTRAN IV      Storage Map for Program Unit DATRUN

Local Variables, .PSECT $DATA, Size = 000136 (   47. words)

Name   Type  Offset      Name   Type  Offset      Name   Type  Offset
BINS   I*2   000034      DAY    I*2   000026      DENOM  R*4   000020
DTIM   I*4   000056      HDRB   I*2   000032      I      I*2   000066
ICHAR  I*2   000072      J      I*2   000070      JCHAR  L*1   000064
K      I*2   000074      MN     I*2   000024      MON    L*1   000063
OPT    L*1   000062      STIM   I*4   000052      TIMH   I*4   000046
TIML   I*4   000042      TIM2   I*4   000036      YR     I*2   000030

COMMON Block /DATTIM/, Size = 000021 (   9. words)

Name   Type  Offset      Name   Type  Offset      Name   Type  Offset
IIDATE L*1   000000      ITIME  L*1   000011

COMMON Block /DISK  /, Size = 000023 (  10. words)

Name   Type  Offset      Name   Type  Offset      Name   Type  Offset
INTSIZ I*2   000000      RECSIZ I*2   000002      FNAME  L*1   000004

COMMON Block /PTERS /, Size = 000006 (   3. words)

Name   Type  Offset      Name   Type  Offset      Name   Type  Offse.
B30    I*2   000000      B60    I*2   000002      S20    I*2   000004

COMMON Block /LREPS /, Size = 000042 (  17. words)

Name   Type  Offset      Name   Type  Offset      Name   Type  Offset
LREP   R*4   000000      LRI    R*4   000004      HRI    R*4   000010
LBI    I*2   000014      HBI    I*2   000016      BSAT   R*4   000020
BSATF  R*4   000024      EANG   R*4   000030      REANG  R*4   000034
PDEL   I*2   000040

COMMON Block /SETS  /, Size = 000006 (   3. words)

Name   Type  Offset      Name   Type  Offset      Name   Type  Offset
SETS   I*2   000000      PROFS  I*2   000002      SHOTSP I*2   000004

COMMON Block /FLAGS /, Size = 000003 (   2. words)

Name   Type  Offset      Name   Type  Offset      Name   Type  Offset
RUNFLG L*1   000000      DFLAG  L*1   000001      ERRFLG L*1   000002

COMMON Block /SEND  /, Size = 000004 (   2. words)

Name   Type  Offset      Name   Type  Offset      Name   Type  Offset
REPNUM I*2   000000      MSHOTS I*2   000002

COMMON Block /STATS /, Size = 000010 (   4. words)

Name   Type  Offset      Name   Type  Offset      Name   Type  Offset
CSET   I*2   000000      CPROF  I*2   000002      LSET   I*2   000004
LPROF  I*2   000006
```

FORTRAN IV   Storage Map for Program Unit DATRUN

COMMON Block /CONST /, Size = 000020 (8. words)

Name	Type	Offset	Name	Type	Offset	Name	Type	Offset
PI	R*4	000000	C	R*4	000004	GATTIM	R*4	000010
RG	R*4	000014						

COMMON Block /BUFF /, Size = 010150 (2100. words)

Name	Type	Offset	Name	Type	Offset	Name	Type	Offset
LDATA	I*2	000000						

Local and COMMON Arrays:

Name	Type	Section	Offset	------Size-----	Dimensions
EXT	L*1	$DATA	000000	000003 (2.)	(3)
FNAME	L*1	DISK	000004	000017 (8.)	(15)
IIDATE	L*1	DATTIM	000000	000011 (5.)	(9)
ITIME	L*1	DATTIM	000011	000010 (4.)	(8)
LDATA	I*2	BUFF	000000	010150 (2100.)	(2100)

Subroutines, Functions, Statement and Processor-Defined Functions:

Name	Type	Name	Type	Name	Type	Name	Type	Name	Type
BAKSCN	R*4	CLRSCN	R*4	DATE	R*4	DISCAL	R*4	FLOAT	R*4
FORSCN	R*4	GTIM	R*4	IDATE	I*2	IFIX	I*2	ITTINR	I*2
JICVT	I*2	JJCVT	I*2	JSUB	I*2	SIN	R*4	SNDCOM	R*4
SNUDTR	R*4	SQRT	R*4	TIMASC	R*4	TIME	R*4		

III.8 EXMPRF

FORTRAN IV   V02.5-5   Tue 31-May-83 15:18:47   PAGE 001

```
      C *****************************************
      C
      C  SUBROUTINE EXMPRF - EXAMINE PROFILE
      C              D.VOELZ
      C
      C *****************************************
      C
      C  THIS SUBROUTINE DISPLAYS A SELECTED PROFILE
      C  FROM THE DATA DISK ON THE TERMINAL AND PRINTER.
      C
0001        SUBROUTINE EXMPRF
0002        INTEGER DSET,DPROF,BINS,LBI,CSET,CPROF,MN,DAY,YR
           >,LDATA(2100),HDRB,NXTL,LSTL,ICNT,BCNT,BNMBR,INMBR
           >,LNMBR,INTSIZ,RECSIZ
0003        INTEGER*4 TIM2
0004        LOGICAL*1 ITIME(8)
0005        BYTE FNAME(15),EXT(3),MON
      C
0006        COMMON/BUFF/ LDATA
0007        COMMON/DISK/ INTSIZ,RECSIZ,FNAME
      C
0008        HDRB=9
      C===============================
      C  RESPONSES
      C===============================
0009        CALL CLRSCN
0010        TYPE 15
0011        TYPE 20
0012        READ(5,*)DSET
0013        IF(DSET.EQ.0)RETURN
0015        TYPE 30
0016        READ(5,*)DPROF
0017        IF(DPROF.EQ.0)RETURN
0019     20 FORMAT(' ',T5,'SET #       ',$)
0020     15 FORMAT(' ',T32,'EXAMINE PROFILE',////)
0021     30 FORMAT(' ',T5,'PROFILE #  ',$)
      C
      C  SET FILE NAME
      C
0022        ENCODE(3,50,EXT)DSET
0023     50 FORMAT(I3)
0024        DO 60 I=1,3
0025        FNAME(7+I)='0'
0026     60 IF(EXT(I).NE.' ')FNAME(7+I)=EXT(I)
      C
      C  DETERMINE RECORD SIZE
      C
0028        OPEN(UNIT=3,TYPE='OLD',RECORDSIZE='2',NAME=FNAME,ACCESS=
           >'DIRECT',FORM='UNFORMATTED',ERR=200)
0029        READ(3'1,ERR=210)BINS
0030        CLOSE(UNIT=3)
0031        RECSIZ=(BINS+HDRB+1)/2
      C
      C  GET DATA
```

```
        C
0032          OPEN(UNIT=3,TYPE='OLD',RECORDSIZE=RECSIZ,NAME=FNAME,ACCESS=
             >'DIRECT',FORM='UNFORMATTED',ERR=200)
0033          READ(3'DPROF,ERR=210)BINS,LBI,CSET,CPROF,MN,DAY,YR,TIM2
             ,(LDATA(K),K=1,BINS)
0034          CLOSE(UNIT=3)
        C===============================
        C  DISPLAY
        C===============================
        C
        C  CONVERT TIME
        C
0035          CALL TIMASC(TIM2,ITIME)
0036          K=0
        C--------------------------------
        C  DISPLAY ON TERMINAL
        C--------------------------------
        C
        C  SET UP PAGE
        C
0037       70 TYPE 75,CSET,MN,DAY,YR,K+1
0038       75 FORMAT(' ',' SET #       ',I4,4X,I2,'-',I2,'-',I2,10X,I2)
0039          TYPE 77,CPROF,ITIME
0040       77 FORMAT(' ',' PROFILE # ',I4,4X,8A1)
0041          NXTL=20
0042          ICNT=1+200*K
0043          BCNT=LBI+ICNT-1
0044          IF(ICNT+199.GE.BINS)NXTL=(BINS-ICNT)/10
0046          IF(NXTL.EQ.0)GOTO 120
        C
        C  TYPE FULL LINES
        C
0048          DO 80 J=1,NXTL
0049          BNMBR=BCNT+10*(J-1)
0050          INMBR=ICNT+10*(J-1)
0051          LNMBR=ICNT+10*J-1
0052       80 TYPE 90,BNMBR,(LDATA(I),I=INMBR,LNMBR)
0053       90 FORMAT(' ',I5,'-',10I6)
0054          IF(NXTL.NE.20)GOTO 120
0056          K=K+1
0057          PAUSE
0058          GOTO 70
        C
        C  TYPE FINAL LINE
        C
0059      120 TYPE 130,BNMBR+10
0060      130 FORMAT(' ',I5,'-',$)
0061          DO 140 X=LNMBR+1,BINS
0062      140 TYPE 150,LDATA(K)
0063      150 FORMAT(I7,$)
0064          TYPE 160
0065      160 FORMAT(' ')
0066          PAUSE 'LAST PAGE'
        C--------------------------------
```

FORTRAN IV   V02.5-5   Tue 31-May-83 15:18:47   PAGE 003

```
        C   PRINT OUT
        C--------------------------------
0067      165 TYPE 170
0068      170 FORMAT('0',T20,'Do you want a printout ? ',$)
0069          READ(5,175,ERR=165)MON
0070      175 FORMAT(A1)
0071          IF(MON.NE.'Y')GOTO 190
        C
        C   SET UP PAGE
        C
0073          PRINT 176,CSET,MN,DAY,YR
0074      176 FORMAT(' ',' SET#        ',I4,4X,I2,'-',I2'-',I2)
0075          PRINT 77,CPROF,ITIME
0076          LSTL=IFIX(FLOAT(BINS)/10.+.99)
        C
        C   PRINT LINES
        C
0077          DO 177 J=1,LSTL
0078          BNMBR=LBI+10*(J-1)
0079          INMBR=1+10*(J-1)
0080          LNMBR=10*J
0081          PRINT 90,BNMBR,(LDATA(I),I=INMBR,LNMBR)
0082      177 CONTINUE
0083          REWIND 6
        C
0084      190 RETURN
        C-----------------------------
        C   ERROR CONDITIONS
        C-----------------------------
0085      200 PAUSE '?? FILE OPENING ERROR'
0086          RETURN
        C
0087      210 PAUSE '?? FILE READ ERROR'
0088          CLOSE(UNIT=3)
0089          RETURN
        C
0090          END
```

FORTRAN IV   Storage Map for Program Unit EXMPRF

Local Variables, .PSECT $DATA, Size = 000130 (44. words)

Name	Type	Offset	Name	Type	Offset	Name	Type	Offset
BCNT	I*2	000060	BINS	I*2	000032	BNMBR	I*2	000062
CPROF	I*2	000040	CSET	I*2	000036	DAY	I*2	000044
DPROF	I*2	000030	DSET	I*2	000026	HDRB	I*2	000050
I	I*2	000076	ICNT	I*2	000056	INMBR	I*2	000064
J	I*2	000102	K	I*2	000100	LBI	I*2	000034
LNMBR	I*2	000066	LSTL	I*2	000054	MN	I*2	000042
MON	L*1	000074	NXTL	I*2	000052	TIM2	I*4	000070
YR	I*2	000046						

COMMON Block /BUFF /, Size = 010150 (2100. words)

Name	Type	Offset	Name	Type	Offset	Name	Type	Offset
LDATA	I*2	000000						

COMMON Block /DISK /, Size = 000023 (10. words)

Name	Type	Offset	Name	Type	Offset	Name	Type	Offset
INTSIZ	I*2	000000	RECSIZ	I*2	000002	FNAME	L*1	000004

Local and COMMON Arrays:

Name	Type	Section	Offset	------Size-----	Dimensions
EXT	L*1	$DATA	000010	000003 (2.)	(3)
FNAME	L*1	DISK	000004	000017 (8.)	(15)
ITIME	L*1	$DATA	000000	000010 (4.)	(8)
LDATA	I*2	BUFF	000000	010150 (2100.)	(2100)

Subroutines, Functions, Statement and Processor-Defined Functions:

Name	Type	Name	Type	Name	Type	Name	Type	Name	Type
CLRSCN	R*4	FLOAT	R*4	IFIX	I*2	TIMASC	R*4		

III.9 DISCAL

FORTRAN IV   V02.5-5   Tue 31-May-83 15:22:06   PAGE 001

```
      C  **************************************************
      C
      C  SUBROUTINE DISCAL.FOR - DATA CALCULATIONS AND DISPLAY
      C                     D.VOELZ
      C
      C  **************************************************
      C
      C  THIS SUBROUTINE PROCESSES INCOMING SODIUM LIDAR PROFILES.
      C  IT CALCULATES AND DISPLAYS PARTICULAR VALUES OF INTEREST.
      C
0001        SUBROUTINE DISCAL
0002        INTEGER CSET,CPROF,LSET,LPROF,INTSIZ
           >,RECSIZ,B30,B60,S20,LDATA(2100),K30,K60,K80,K100
           >,SGP,TSGP,AK30,AK60,AK80,AK100,ASGP,ATSGP
           >,REPNUM,MSHOTS,LBI,HBI,PDEL
0003        REAL CLA,ACLA,RAYC,LREP,LRI,HRI,EANG,REANG,BSAT,BSATF
0004        LOGICAL*1 IDATE(9),ITIME(8)
0005        BYTE RUNFLG,DFLAG,ERRFLG
      C
0006        COMMON /DATTIM/ IDATE,ITIME
0007        COMMON /STATS/ CSET,CPROF,LSET,LPROF
0008        COMMON /CALCS/ K30,K60,K80,K100,SGP,TSGP,CLA
0009        COMMON /FLAGS/ RUNFLG,DFLAG,ERRFLG
0010        COMMON /PTERS/ B30,B60,S20
0011        COMMON /BUFF/ LDATA
0012        COMMON /SEND/ REPNUM,MSHOTS
0013        COMMON /LREPS/ LREP,LRI,HRI,LBI,HBI,BSAT,BSATF,EANG,
           >REANG,PDEL
      C
0014        CALL CLRSCN
      C
      C-----------------------------
      C  CALCULATIONS
      C-----------------------------
0015        AK30=0
0016        AK60=0
0017        AK80=0
0018        AK100=0
      C  CALCULATE K30
0019        DO 10 I=1,5
0020  10    AK30=AK30+LDATA(B30-3+I)
0021        AK30=IFIX(FLOAT(AK30)/5.+.5)
      C  CALCULATE K60,K80,100
0022        DO 20 I=1,S20
0023        AK60=AK60+LDATA(B60-1+I)
0024        AK80=AK80+LDATA(B60-1+I+S20)
0025  20    AK100=AK100+LDATA(B60-1+I+S20+S20)
      C  CALCULATE SIGNAL PHOTONS
0026        ATSGP=AK80-AK60
0027        ASGP=IFIX(FLOAT(ATSGP)/MSHOTS+.5)
      C  COLUMN ABUNDANCE RATIO
0028        RAYC=FLOAT(AK30)-FLOAT(AK60)/S20
0029        ACLA=0.
0030        IF(RAYC.LE.0.)GOTO 25
```

```
FORTRAN IV      V02.5-5    Tue 31-May-83 15:22:06          PAGE 002

0032        ACLA=FLOAT(ATSGP)/RAYC
      C--------------------------------
      C  CHECK RUN FLAG
      C--------------------------------
0033     25 IF(RUNFLG.EQ.3)GOTO 30
      C--------------------------------
      C  DATA RUN DISPLAY
      C--------------------------------
0035        K30=AK30
0036        K60=AK60
0037        K80=AK80
0038        K100=AK100
0039        SGP=ASGP
0040        TSGP=ATSGP
0041        CLA=ACLA
      C
0042        TYPE 40
0043        TYPE 50,(IDATE(I),I=1,9),(ITIME(I),I=1,8)
0044        TYPE 60,CSET,CPROF
0045        TYPE 70,MSHOTS
0046        TYPE 72,BSAT
0047        TYPE 80,K30,K60,K80,K100
0048        TYPE 90,TSGP
0049        TYPE 100,SGP
0050        TYPE 110,CLA
      C
0051        PRINT 50,(IDATE(I),I=1,9),(ITIME(I),I=1,8)
0052        PRINT 60,CSET,CPROF
0053        PRINT 70,MSHOTS
0054        PRINT 72,BSAT
0055        PRINT 80,K30,K60,K80,K100
0056        PRINT 90,TSGP
0057        PRINT 100,SGP
0058        PRINT 110,CLA
0059        PRINT 47
0060        REWIND 6
      C
0061        RETURN
      C--------------------------------
      C SODIUM ALIGNMENT RUN DISPLAY
      C--------------------------------
0062     30 TYPE 45
0063        TYPE 50,(IDATE(I),I=1,9),(ITIME(I),I=1,8)
0064        TYPE 70,MSHOTS
0065        TYPE 72,BSAT
0066        TYPE 80,AK30,AK60,AK80,AK100
0067        TYPE 90,ATSGP
0068        TYPE 100,ASGP
0069        TYPE 110,ACLA
      C
0070        RETURN
      C--------------------------------
      C  FORMAT STATEMENTS
      C--------------------------------
```

```
FORTRAN IV      V02.5-5     Tue 31-May-83 15:22:06          PAGE 003
0071     40 FORMAT(' ',/,T35,'DATA RUN',/)
0072     45 FORMAT(' ',/,T30,'SODIUM ALIGNMENT RUN',/)
0073     47 FORMAT(' ',T23,'---------------------------------')
0074      ) FORMAT(' ',T35,9A1,/T35,8A1,)
0075     60 FORMAT(' ',T33,'Set #        ',I4,/T33,'Profile # ',I4,/)
0076     70 FORMAT(' ',T23,'Laser Shots Per Profile   =',I6)
0077     72 FORMAT(' ',T23,'Base Altitude Setting     =',F6.2)
0078     80 FORMAT(' ',T23,'Detected Photons:',/T38,'at 30 Km   =',I6,
           >/T35,'60 -  80 Km   =',I6,/T35,'80 - 100 Km   =',I6,/T34,'100
           > - 120 Km   =',I6)
0079     90 FORMAT(' ',T23,'Total Signal Photons      =',I6)
0080    100 FORMAT(' ',T23,'Signal Photons Per Shot   =',I6)
0081    110 FORMAT(' ',T23,'Column Abundance Ratio    =',F6.2)
0082        END
```

FORTRAN IV   Storage Map for Program Unit DISCAL

Local Variables, .PSECT $DATA, Size = 000070 (28. words)

Name	Type	Offset	Name	Type	Offset	Name	Type	Offset
ACLA	R*4	000036	AK100	I*2	000030	AK30	I*2	000022
AK60	I*2	000024	AK80	I*2	000026	ASGP	I*2	000032
ATSGP	I*2	000034	I	I*2	000046	INTSIZ	I*2	000016
RAYC	R*4	000042	RECSIZ	I*2	000020			

COMMON Block /DATTIM/, Size = 000021 (9. words)

Name	Type	Offset	Name	Type	Offset	Name	Type	Offset
IDATE	L*1	000000	ITIME	L*1	000011			

COMMON Block /STATS /, Size = 000010 (4. words)

Name	Type	Offset	Name	Type	Offset	Name	Type	Offset
CSET	I*2	000000	CPROF	I*2	000002	LSET	I*2	000004
LPROF	I*2	000006						

COMMON Block /CALCS /, Size = 000020 (8. words)

Name	Type	Offset	Name	Type	Offset	Name	Type	Offset
K30	I*2	000000	K60	I*2	000002	K80	I*2	000004
K100	I*2	000006	SGP	I*2	000010	TSGP	I*2	000012
CLA	R*4	000014						

COMMON Block /FLAGS /, Size = 000003 (2. words)

Name	Type	Offset	Name	Type	Offset	Name	Type	Offset
RUNFLG	L*1	000000	DFLAG	L*1	000001	ERRFLG	L*1	000002

COMMON Block /PTERS /, Size = 000006 (3. words)

Name	Type	Offset	Name	Type	Offset	Name	Type	Offset
B30	I*2	000000	B60	I*2	000002	S20	I*2	000004

COMMON Block /BUFF /, Size = 010150 (2100. words)

Name	Type	Offset	Name	Type	Offset	Name	Type	Offset
LDATA	I*2	000000						

COMMON Block /SEND /, Size = 000004 (2. words)

Name	Type	Offset	Name	Type	Offset	Name	Type	Offset
REPNUM	I*2	000000	MSHOTS	I*2	000002			

COMMON Block /LREPS /, Size = 000042 (17. words)

Name	Type	Offset	Name	Type	Offset	Name	Type	Offset
LREP	R*4	000000	LRI	R*4	000004	HRI	R*4	000010
LBI	I*2	000014	HBI	I*2	000016	BSAT	R*4	000020
BSATF	R*4	000024	EANG	R*4	000030	REANG	R*4	000034
PDEL	I*2	000040						

FORTRAN IV   Storage Map for Program Unit DISCAL

Local and COMMON Arrays:

Name	Type	Section	Offset	------Size-----	Dimensions
IDATE	L*1	DATTIM	000000	000011 (5.)	(9)
ITIME	L*1	DATTIM	000011	000010 (4.)	(8)
LDATA	I*2	BUFF	000000	010150 (2100.)	(2100)

Subroutines, Functions, Statement and Processor-Defined Functions:

Name	Type	Name	Type	Name	Type	Name	Type	Name	Type
CLRSCN	R*4	FLOAT	R*4	IFIX	I*2				

RCVER.MAC   MACRO V04.00   31-MAY-83 15:40:37 PAGE 1

```
  1                             ;****************************************
  2                             ;
  3                             ;  LSI-11 RECEIVER
  4                             ;      D.VOELZ
  5                             ;
  6                             ;****************************************
  7                                     .TITLE  RCVER.MAC
  8                                     .MCALL  .INTEN,.PROTECT
  9                                     .GLOBL  HDUMP,SNDACK,INITR,MOUT
 10 000000                              .CSECT  RCVER
 11                             ;
 12                             ;  I/O LOCATIONS
 13                             ;
 14         176500             RCSR=176500             ;DLV11-J CHN.0 LOCATIONS
 15         176502             RBUF=RCSR+2
 16         000300             PCVEC=300
 17         000302             STVEC=PCVEC+2
 18                             ;
 19                             ;   GENERAL PARAMETERS
 20                             ;
 21         000340             PR7=340                 ;PRIORITY LEVEL 7
 22         000003             ERRCNT=3                ;ERROR COUNT
 23         000063             PRDONE=063              ;OPCODES-PROFILE DONE
 24         000064             ACK=064                 ;        -ACKNOWLEDGE
 25         000071             DATEND=071              ;        -END OF DATA
 26         000201             CTX=201                 ;'CTX' CHAR
 27         000202             ETX=202                 ;'ETX' CHAR
 28         000203             STX=203                 ;'STX' CHAR
 29         000220             DLE=220                 ;'DLE' CHAR
 30                             ;
 31                             ;   DATA PACKET PARAMETERS
 32                             ;
 33         000120             OPCLO=120               ;LO BYTE DATA OPCODE
 34         000340             OPCHI=340               ;HI BYTE DATA OPCODE
 35         003720             PAKLEN=2000.            ;PACKET LENGTH (MAX=32767)
 36                             ;
 37                             ;----------------------------------------
 38                             ;   INITIALIZE RECEIVER
 39                             ;----------------------------------------
 40 000000                     INITR:: .PROTECT        #PAREA,#300
 41 000020 012737 000062' 000300      MOV     #IOINT,@#PCVEC ;SET UP PC VECTOR
 42 000026 012737 000340 000302       MOV     #PR7,@#STVEC   ;SET UP STATUS WORD VECTOR
 43 000034 012767 000144' 000000'     MOV     #STPAK1,STATE  ;SET INPUT BYTE VECTOR
 44 000042 000207                     RTS     PC
 45                             ;----------------------------------------
 46                             ;  HEX MEMORY DUMP OF ARRAY LDATA
 47                             ;----------------------------------------
 48 000044 012701 000000'      MOUT::  MOV     #LDATA,R1
 49 000050 012702 007640               MOV     #4000.,R2
 50 000054 004767 000000G              JSR     PC,HDUMP
 51 000060 000207                      RTS     PC
 52                             ;
 53                             ;----------------------------------------
 54                             ;  INTERRUPT POINTS
 55                             ;----------------------------------------
 56                             ;
 57                             ;  INTERRUPT ENTRY POINT
```

RCVER.MAC   MACRO V04.00   31-MAY-8: 15:40:37 PAGE 1-1

```
58                                        ;
59 000062                          IOINT:  .INTEN  7              ;ALERT RT-11 OF INTERRUPT
60 000070  010046                         MOV     R0,-(SP)        ;SAVE REGISTERS
61 000072  010146                         MOV     R1,-(SP)
62 000074  010246                         MOV     R2,-(SP)
63 000076  010346                         MOV     R3,-(SP)
64                                        ;
65 000100  016701  000000'                MOV     STATE,R1        ;MOVE INPUT BYTE VECTOR TO R1
66                                        ;
67 000104  113700  176502                 MOVB    @@RBUF,R0       ;LOAD R0 WITH INPUT BYTE
68 000110  000111                         JMP     (R1)            ;GO TAKE CARE OF INPUT BYTE
69                                        ;
70                                        ; INTERRUPT EXIT POINT
71                                        ;
72 000112  042700  177400         GETBT:  BIC     #177400,R0      ;CLEAR HI BYTE OF R0
73 000116  060067  000002'                ADD     R0,RCSUM        ;UPDATE RECD CHECKSUM
74                                        ;
75 000122  012667  000000'                MOV     (SP)+,STATE     ;SET NEW INPUT BYTE VECTOR
76                                        ;
77 000126  012603                         MOV     (SP)+,R3        ;RESTORE REGISTERS
78 000130  012602                         MOV     (SP)+,R2
79 000132  012601                         MOV     (SP)+,R1
80 000134  012600                         MOV     (SP)+,R0
81 000136  000207                         RTS     PC
82                                        ;
83                                        ;-------------------------------------
84                                        ; PACKET RECEIVING
85                                        ;-------------------------------------
86 000140  004767  177746         STPAK:  JSR     PC,GETBT
87 000144  120027  000220         STPAK1: CMPB    R0,#DLE         ;CHECK FOR 'DLE' BYTE
88 000150  001373                         BNE     STPAK
89                                        ;
90 000152  004767  177734                 JSR     PC,GETBT
91 000156  120027  000203                 CMPB    R0,#STX         ;CHECK FOR 'STX' BYTE
92 000162  001366                         BNE     STPAK
93 000164  005067  000002'        OPPAK:  CLR     RCSUM           ;CLEAR RECD CHECKSUM ACCUMULATOR
94                                        ;
95 000170  004767  177716                 JSR     PC,GETBT        ;GET OPCODE
96 000174  110067  000016'                MOVB    R0,OPCDE        ;  SAVE IT
97                                        ;
98                                        ; VERIFY CHECKSUM
99                                        ;
100 000200  004767  177706        VERCHK: JSR     PC,GETBT        ;GET 'DLE' BYTE
101 000204  004767  177702                JSR     PC,GETBT        ;GET 'ETX' BYTE
102 000210  016767  000002' 000004' VERIFY: MOV   RCSUM,SAVSUM    ;SAVE RECD CHECKSUM
103                                        ;
104 000216  004767  177670                JSR     PC,GETBT        ;GET LO BYTE XMITTED CHECKSUM
105 000222  120067  000004'               CMPB    R0,SAVSUM       ;  CHECK IT
106 000226  001344                        BNE     STPAK
107 000230  004767  177656                JSR     PC,GETBT        ;GET HI BYTE XMITTED CHECKSUM
108 000234  000367  000004'               SWAB    SAVSUM
109 000240  120067  000004'               CMPB    R0,SAVSUM       ;  CHECK IT
110 000244  001335                        BNE     STPAK
111                                        ;
112                                        ; DECIPHER OPCODE
113                                        ;
114 000246  116701  000016'               MOVB    OPCDE,R1        ;MOVE OPCODE TO R1
```

RCVER.MAC   MACRO V04.00   31-MAY-83 15:40:37 PAGE 1-2

```
115                                      ;
116 000252 122701 000064              CMPB    #ACK,R1      ; ACK ?
117 000256 001433                     BEQ     ACKN
118 000260 122701 000071              CMPB    #DATEND,R1   ; DAT.. PACKET COMPLETED ?
119 000264 001003                     BNE     PRFCHK       ;   -NO,GO ON
120 000266 004767 000000G             JSR     PC,SNDACK    ;   -YES,SEND ACK
121 000272 000722                     BR      STPAK
122 000274 122701 000063    PRFCHK:   CMPB    #PRDONE,R1   ; PROFILE COMPLETED ?
123 000300 001011                     BNE     DATCHK       ;   -NO,GO ON
124 000302 004767 000000G             JSR     PC,SNDACK    ;   -YES,SEND ACK
125 000306 112767 000001 000001'      MOVB    #1,DFLAG     ;   SET DATA RUN FLAG
126 000314 042737 000100 175500       BIC     #100,@@RCSR  ;   DISABLE DLV11-J INTERRUPT
127 000322 000706                     BR      STPAK
128 000324 042701 000017    DATCHK:   BIC     #17,R1       ;CLEAR LOWER 4 BITS OF OPCODE
129 000330 122701 000120              CMPB    #OPCLO,R1    ; LOW BYTE DATA ?
130 000334 001410                     BEQ     LBYTES
131 000336 122701 000340              CMPB    #OPCHI,R1    ; HI BYTE DATA ?
132 000342 001410                     BEQ     HBYTES
133 000344 000675                     BR      STPAK        ;BAD OPCODE
134                         ;
135                         ;-----------------------------------------
136                         ; RECEIVED ACKNOWLEDGE
137                         ;-----------------------------------------
138 000346 112767 000001 000000' ACKN: MOVB   #1,ACKFLG    ;SET ACK FLAG
139 000354 000671                     BR      STPAK
140                         ;-----------------------------------------
141                         ; RECEIVE DATA FRAME
142                         ;-----------------------------------------
143                         ;
144                         ; DETERMINE BASE ADDRESS FOR DATA FRAME
145                         ;
146 000356 012704 000000'   LBYTES:   MOV     #LDATA,R4    ;SET BASE ADDRESS FOR LO BYTES
147 000362 000402                     BR      DATFRM
148 000364 012704 000001'   HBYTES:   MOV     #<LDATA+1>,R4 ;SET BASE ADDRESS FOR HI BYTES
149                         ;
150 000370 012703 007640    DATFRM:   MOV     #<2*PAKLEN>,R3 ;MOVE DOUBLE PACKET LENGTH INTO R3
151 000374 116701 000016'             MOVB    OPCDE,R1     ;MOVE OPCODE INTO R1
152 000400 112767 000071 000016'      MOVB    #DATEND,OPCDE ;LOAD 'DATEND' OPCODE
153                         ;
154 000406 042701 177760              BIC     #177760,R1   ;CLEAR HI 12 BITS OF OPCODE
155 000412 070301                     MUL     R1,R3        ;MULT. ADJSTD. OPCODE & DBL. PACKET LENGTH
156 000414 060304                     ADD     R3,R4        ;ADD ADDRESS BASE & ADDRESS OFFSET
157 000416 010467 000006'             MOV     R4,STADD     ;SAVE DATA PACKET STARTING ADDRESS
158                         ;
159                         ; RECEIVE DATA BYTES
160                         ;
161 000422 004767 177464              JSR     PC,GETBT
162 000426 120027 000220              CMPB    R0,#DLE      ;CHECK FOR 'DLE' CHARACTER
163 000432 001242                     BNE     STPAK
164                         ;
165 000434 004767 177452              JSR     PC,GETBT
166 000440 120027 000201              CMPB    R0,#CTX      ;CHECK FOR 'CTX' CHARACTER
167 000444 001235                     BNE     STPAK
168 000446 005067 000002'             CLR     RCSUM        ;CLEAR RECD CHECKSUM ACCUMULATOR
169                         ;
170 000452 004767 177434    DATA:     JSR     PC,GETBT     ;GET DATA BYTE
171 000456 120027 000220              CMPB    R0,#DLE      ; CHECK FOR 'DLE' CHARACTER
```

```
172 000462  001407              BEQ     CHKDBL
173                     ;
174 000464  016704  000006'  OKDBL:  MOV     STADD,R4
175 000470  110024              MOVB    R0,(R4)+        ; DEPOSIT BYTE
176 000472  005204              INC     R4
177 000474  010457  000006'     MOV     R4,STADD
178 000500  000764              BR      DATA
179                     ;
180 000502  004767  177404  CHKDBL: JSR     PC,GETBT        ;THROW AWAY FIRST 'DLE'
181 000506  120027  000220     CMPB    R0,#DLE         ;   DLE ?
182 000512  001764              BEQ     OKDBL
183 000514  120027  000202  ENDIT:  CMPB    R0,#ETX         ;  ETX ? (FRAME DONE)
184 000520  001633              BEQ     VERIFY
185 000522  000167  177412     JMP     STPAK           ;FRAME ENDER ERROR
186                     ;
187                     ;
188 000000                      .PSECT  RCV,RW,D,LCL,REL,CON
189 000000              STATE:  .BLKW   1
190 000002              RCSUM:  .BLKW   1
191 000004              SAVSUM: .BLKW   1
192 000006              STADD:  .BLKW   1
193 000010              PAREA:  .BLKW   3
194 000016              OPCDE:  .BLKB   1
195                     ;
196 000000                      .PSECT  ACKFLG,RW,D,GBL,REL,OVR
197 000000              ACKFLG: .BLKB   1
198                     ;
199 000000                      .PSECT  BUFF,RW,D,GBL,REL,OVR
200 000000              LDATA:  .BLKW   2100.
201                     ;
202 000000                      .PSECT  FLAGS,RW,D,GBL,REL,OVR
203 000000              RUNFLG: .BLKB   1
204 000001              DFLAG:  .BLKB   1
205 000002              ERRFLG: .BLKB   1
206                     ;
207         000001              .END
```

```
RCVER.MAC       MACRO V04.00  31-MAY-83 15:40:37 PAGE 1-4
SYMBOL TABLE

ACK   = 000064        DLE  = 000220        LBYTES  000356R   002  RCVEC = 000300        STADD   000006R  003
ACKFLG  000000R   004 ENDIT   000514R   002 LDATA   000000R   005  PRDONE= 000003        STATE   000000R  003
ACKN    000346R   002 ERRCNT= 000003        MOUT    000044RG  002  PRFCHK  000274R   002 STPAK   000140R  002
CHKDBL  000502R   002 ERRFLG  000002R   006 OKDBL   000464R   002  PR7   = 000340        STPAK1  000144R  002
CTX   = 000201        ETX   = 000202        OPCDE   000016R   003  RBUF  = 176502        STVEC = 000302
DATA    000452R   002 GETBT   000112R   002 OPCHI = 000340        RCSR  = 176500        STX   = 000203
DATCHK  000324R   002 HBYTES  000364R   002 OPCLO = 000120        RCSUM   000062R   003 VERCHK  000200R  002
DATEND= 000071        HDUMP = ****** G      OPPAK   000164R   002  RUNFLG  000000R   006 VERIFY  000210R  002
DATFRM  000370R   002 INITR   000000RG  002 PAKLEN= 003720        SAVSUM  000004R   003 ...V1 = 000003
DFLAG   000001R   006 IOINT   000062R   002 PAREA   000010R   003  SNDACK= ****** G

. ABS.  000000   000
        000000   001
RCVER   000526   002
RCV     000017   003
ACKFLG  000001   004
BUFF    010150   005
FLAGS   000003   006
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  9216 WORDS  ( 36 PAGES)
DYNAMIC MEMORY AVAILABLE FOR  54 PAGES
RCVER,LP:=RCVER.MAC
```

III.1J   SENDER

SENDER.MAC   MACRO V04.00   31-MAY-83 15:45:58 PAGE 1

```
 1                              ;*************************************
 2
 3                              ; LSI-11 SENDER
 4                              ;    D.VOELZ
 5
 6                              ;*************************************
 7                                      .TITLE  SENDER.MAC
 8                                      .MCALL  .TTYOUT,.PRINT,.TTYIN
 9                                      .GLOBL  SNDACK,SNDDTR,SNDCOM,CLRSCN
10 000000                               .CSECT  SENDER
11                              ;
12                              ;  I/O LOCATIONS
13                              ;
14         176500               RCSR=176500             ;DLV11-J CHN.C LOCATIONS
15         176502               RBUF=RCSR+2
16         176504               XCSR=RCSR+4
17         176506               XBUF=RCSR+6
18                              ;
19                              ;  GENERAL PARAMETERS
20                              ;
21         000003               ERRCNT=3                ;ERROR COUNT
22         000061               DRUN=061                ;OPCODES-DATA RUN
23         000062               COM=062                 ;        -COMMANDS
24         000064               ACK=064                 ;        -ACKNOWLEDGE
25         000201               CTX=201                 ;'CTX' CHAR
26         000202               ETX=202                 ;'ETX' CHAR
27         000203               STX=203                 ;'STX' CHAR
28         000220               DLE=220                 ;'DLE' CHAR
29                              ;
30                              ;  MACRO DIRECTIVE - AKWAIT
31                              ;    THIS MACRO DIRECTIVE IS THE TIME-OUT LOOP FOR AN
32                              ;    ACKNOWLEDGE. IF THE ACKNOWLEDGE FLAG (ACKFLG) IS
33                              ;    NOT SET WHILE THE ROUTINE IS IN ITS LOOP, PROGRAM
34                              ;    CONTROL JUMPS TO THE ERROR ROUTINE 'ERROUT'.
35                              ;
36                                      .MACRO  AKWAIT  JMPADD,?A,?B,?C
37                                      MOV     2000.,R0
38                              A:      CMPB    ACKFLG,#1
39                                      BEQ     C
40                                      DEC     R0
41                                      BNE     A
42                              B:      DEC     R3
43                                      BNE     JMPADD
44                                      JMP     ERROUT
45                              C:      NOP
46                                      .ENDM   AKWAIT
47                              ;
48                              ;========================================
49                              ;  SEND ACKNOWLEDGE
50                              ;========================================
51 000000 112767 000064 000002' SNDACK::MOVB   #ACK,OPCDE      ;LOAD 'ACK' OPCODE
52 000006 004767 000162         JSR     PC,SNDFRM      ;SEND ACK FRAME
53 000012 000207               RTS     PC
54                              ;
55                              ;========================================
56                              ;  SEND DATA RUN FRAME
57                              ;========================================
```

SENDER.MAC   MACRO V04.00   31-MAY-83 15:45:58 PAGE 1-1

```
58 000014  052737  000100  176500  SNDDTR::BIS   #100,@#RCSR
59 000022  112767  000061  000002'         MOVB   #DRUN,OPCDE    ;LOAD 'DRUN' OPCODE
60 000030  012703  000003          MOV    #ERRCNT,R3     ;SET ERROR COUNT
61 000034  105067  000000'  SNDTR1: CLRB   ACKFLG         ;CLEAR ACK FLAG
62 000040  004767  000130          JSR    PC,SNDFRM      ;SEND DATA RUN FRAME
63 000044                          AKWAIT SNDTR1         ;WAIT FOR ACK
64 000076  000207                  RTS    PC
65                          ;
66                          ;=======================================
67                          ;   SEND COMMANDS FRAME
68                          ;=======================================
69 000100  052737  000100  176500  SNDCOM::BIS   #100,@#RCSR
70 000106  112767  000062  000002'         MOVB   #COM,OPCDE     ;LOAD 'COMMANDS' OPCODE
71 000114  012703  000003          MOV    #ERRCNT,R3     ;SET ERROR COUNT
72 000120  105067  000000'  SNCOM1: CLRB   ACKFLG         ;CLEAR ACK FLAG
73 000124  004767  000044          JSR    PC,SNDFRM      ;SEND COMMANDS FRAME PART 1
74 000130  004767  000132          JSR    PC,SNDCM2      ;SEND COMMANDS FRAME PART 2
75 000134                          AKWAIT SNCOM1
76 000166  004767  177622          JSR    PC,SNDDTR      ;SEND DATA RUN FRAME
77 000172  000207                  RTS    PC
78                          ;
79                          ;------------------------------------
80                          ;   GENERAL PACKET SENDING ROUTINE
81                          ;------------------------------------
82 000174  112700  000220  SNDFRM: MOVB   #DLE,R0
83 000200  004767  000146          JSR    PC,SNDBYT      ;SEND 'DLE' CHARACTER
84 000204  005001                  CLR    R1             ;CLEAR XMIT CHECKSUM
85                          ;
86 000206  112700  000203          MOVB   #STX,R0
87 000212  004767  000126          JSR    PC,UPDTCK      ;UPDATE XMIT CHECKSUM & SEND 'STX'
88                          ;
89 000216  116700  000002'         MOVB   OPCDE,R0
90 000222  004767  000116          JSR    PC,UPDTCK      ;UPDATE XMIT CHECKSUM & SEND OPCODE
91                          ;
92 000226  112700  000220  ENDER:  MOVB   #DLE,R0
93 000232  004767  000106          JSR    PC,UPDTCK      ;UPDATE XMIT CHECKSUM & SEND 'DLE'
94                          ;
95 000236  112700  000202          MOVB   #ETX,R0
96 000242  004767  000104          JSR    PC,SNDBYT      ;SEND 'ETX' CHARACTER
97                          ;
98 000246  110100          SNDCHK: MOVB   R1,R0          ;SEND XMIT CHECKSUM
99 000250  004767  000076          JSR    PC,SNDBYT      ;  LO BYTE
100 000254  000301                 SWAB   R1
101 000256  110100                 MOVB   R1,R0
102 000260  004767  000066         JSR    PC,SNDBYT      ;  HI BYTE
103 000264  000207                 RTS    PC
104                          ;
105                          ;---------------- ------------------
106                          ;   SEND COMMANDS FRAME PART 2
107                          ;------------------------------------
108 000266  112700  000220  SNDCM2: MOVB   #DLE,R0
109 000272  004767  000054          JSR    PC,SNDBYT      ;SEND 'DLE' CHARACTER
110 000276  005001                  CLR    R1             ;CLEAR XMIT CHECKSUM
111                          ;
112 000300  112700  000201          MOVB   #CTX,R0
113 000304  004767  000034          JSR    PC,UPDTCK      ;UPDATE XMIT CHECKSUM & SEND 'CTX'
114                          ;
```

SENDER.MAC   MACRO V04.00 31-MAY-83 15:45:58 PAGE 1-2

```
115 000310  116700  000000'              MOVB    REPNUM,R0
116 000314  004767  000024          ;    JSR     PC,UPDTCK        ;SEND REP NUMBER
117                                  ;
118 000320  116700  000002'              MOVB    MSHOTS,R0        ;SEND MSHOTS
119 000324  004767  000014               JSR     PC,UPDTCK        ;  LO BYTE
120 000330  016700  000002'              MOV     MSHOTS,R0
121 000334  000300                        SWAB    R0
122 000336  004767  000002               JSR     PC,UPDTCK        ;  HI BYTE
123                                  ;
124 000342  000731                        BR      ENDER           ;SEND PACKET ENDER
125
126                                  ;
127                                  ;------------------------------------
128                                  ;   UPDATE CHECKSUM & SEND BYTE
129                                  ;           ROUTINES
130                                  ;------------------------------------
131 000344  042700  177400          UPDTCK: BIC    #177400,R0      ;CLEAR HI BYTE OF R0
132 000350  060001                        ADD     R0,R1            ;UPDATE CHECKSUM
133                                  ;
134 000352  032737  000200  176504  SNDBYT: BIT    #200,@#XCSR      ;CHECK STATUS REG
135 000360  001774                        BEQ     SNDBYT
136 000362  110037  176506               MOVB    R0,@#XBUF        ;OUTPUT R0
137 000366  000207                        RTS     PC
138                                  ;
139                                  ;------------------------------------
140                                  ;   ERROR MESSAGE
141                                  ;------------------------------------
142 000370  004767  000000G         ERROUT: JSR    PC,CLRSCN
143 000374                                .PRINT  #ERRMSG
144 000402                                .TTYIN                   ;GET RESPONSE
145 000406  110001                        MOVB    R0,R1            ;  STORE IT
146 000410                                .TTYIN                   ;PICK UP LF
147 000414                                .TTYIN                   ;PICK UP CR
148 000420  120127  000131               CMPB    R1,#'Y           ;CHECK RESPONSE
149 000424  001002                        BNE     EDONE
150 000426  000167  177446               JMP     SNDCOM
151 000432  112767  000001  000001' EDONE: MOVB    #1,DFLAG        ;SET DONE FLAG
152 000440  112767  000001  000002'       MOVB    #1,ERRFLG       ;SET ERROR FLAG
153 000446  042737  000100  176500        BIC     #100,@#RCSR      ;DISABLE DLV11-J INTERRUPT
154 000454  000207                        RTS     PC
155                                  ;
156 000000                                .PSECT  MSGR,RW,D,LCL,REL,CON
157 000000     015     012     012  ERRMSG: .BYTE  15,12,12,12,12,12,12,12,12
    000003     012     012     012
    000006     012     012     012
158 000011     040     040     040         .ASCII  /                           COMMAND TRANSMISSION ERROR/
    000014     040     040     040
    000017     040     040     040
    000022     040     040     040
    000025     040     040     040
    000030     040     040     040
    000033     040     040     040
    000036     040     040     040
    000041     040     040     040
    000044     103     117     115
    000047     115     101     116
    000052     104     040     124
```

```
        000055    122   101   116
        000060    123   115   111
        000063    123   123   111
        000066    117   116   040
        000071    105   122   122
        000074    117   122
159 000076    015   012   012           .BYTE   15,12,12
160 000101    040   040   040           .ASCII  /                         /
    000104    040   040   040
    000107    040   040   040
    000112    040   040   040
    000115    040   040   040
    000120    040   040   040
    000123    040   040
161 000125    040   040   040           .ASCIZ  /              TRY AGAIN ? /<200>
    000130    040   040   040
    000133    040   040   040
    000136    040   040   040
    000141    040   040   124
    000144    122   131   040
    000147    101   107   101
    000152    111   116   040
    000155    077   040   200
    000160    000
162                               ;
163 000000                               .PSECT  CCM,RW,D,LCL,REL,CON
164 000000                        XCSUM:  .BLKW   1
165 000002                        OPCDE:  .BLKB   1
166                               ;
167 000000                               .PSECT  ACKFLG,RW,D,GBL,REL,OVR
168 000000                        ACKFLG: .BLKB   1
169                               ;
170 000000                               .PSECT  FLAGS,RW,D,GBL,REL,OVR
171 000000                        RUNFLG: .BLKB   1
172 000001                        DFLAG:  .BLKB   1
173 000002                        ERRFLG: .BLKB   1
174                               ;
175 000000                               .PSECT  SEND,RW,D,GBL,REL,OVR
176 000000                        REPNUM: .BLKW   1
177 000002                        MSHOTS: .BLKW   1
178                               ;
179         000001                       .END
```

```
SENDER.MAC     MACRO V04.00  31-MAY-83 15:45:58 PAGE 1-4
SYMBOL TABLE

ACK   = 000064         DRUN  = 000061         ETX   = 000202        SNCOM1 000120R  002 SNDFRM 000174R 002
ACKFLG  000000R    005 EDONE   000432R    002 MSHCTS 000002R    007 SNDACK 000000RG 002 SNDTR1 000034R 002
CLRSCN= ****** G       ENDER   000226R    002 OPCDE  000002R    004 SNDBYT 000352P  002 STX  = 000203
COM   = 000062         ERRCNT= 000003         RBUF  = 176502        SNDCHK 000246R  002 UPDTCK 000344R 002
CTX   = 000201         ERRFLG  000002R    006 RCSR  = 176500        SNDCM2 000266R  002 XBUF = 176506
DFLAG   000001R    006 ERRMSG  000000R    003 REPNUM 000000R    007 SNDCOM 000100RG 002 XCSR = 176504
DLE   = 000220         ERROUT  000370R    002 RUNFLG 000000R    006 SNDDTR 000014RG 002 XCSUM  000000R 004


. ABS.  000000    000
        000000    001
SENDER  000456    002
MSGR    000161    003
CCM     000003    004
ACKFLG  000001    005
FLAGS   000003    006
SEND    000004    007
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  8704 WORDS  ( 34 PAGES)
DYNAMIC MEMORY AVAILABLE FOR  54 PAGES
SENDER,LP:=SENDER.MAC
```

III.12 DISPLY

DISPLY.MAC   MACRO V04.00   31-MAY-83 16:02:57 PAGE 1

```
 1                                    ;***********************************************
 2
 3                                    ; DISPLY.MAC - VT100 TERMINAL DISPLAY ROUTINES
 4                                    ;              D.VOELZ
 5
 6                                    ;***********************************************
 7
 8                                    ; THIS SUBPROGRAM SETS UP THE DISPLAY OF THE VT100
 9                                    ; FAMILY OF TERMINALS. THE SUBROUTINES ARE AS FOLLOWS:
10                                    ;     CLRSCN - CLEAR SCREEN, CURSOR TO HOME
11                                    ;     FORSCN - INVERSE VIDEO
12                                    ;     BAKSCN - NORMAL VIDEO
13                                    ;     BELSCN - RING THE BELL
14                                    ;
15                                           .TITLE DISPLY.MAC
16                                           .MCALL .TTYOUT
17 000000                                    .CSECT  DISPLY
18                                    ;
19        000033                      ESC=033
20        000155                      LCM=155
21                                    ;
22                                    ;========================================
23                                    ; VT100 DISPLAY SUBROUTINES
24                                    ;========================================
25 000000  012701  000054'           CLRSCN::MOV     #CLRSTR,R1
26 000004  000413                             BR      OUT
27 000006  012701  000064'           FORSCN::MOV     #FORSTR,R1
28 000012  000410                             BR      OUT
29 000014  012701  000071'           BAKSCN::MOV     #BAKSTR,R1
30 000020  000405                             BR      OUT
31 000022                            BELSCN::.TTYOUT BELL
32 000032  000207                             RTS     PC
33                                    ;--------------------------------------
34                                    ; ASCII STRING OUTPUT SECTION
35                                    ;--------------------------------------
36 000034  111100                    OUT:    MOVB    (R1),R0
37 000036                                     .TTYOUT
38 000042  005201                             INC     R1
39 000044  121127  000000                     CMPB    (R1),#0
40 000050  001371                             BNE     OUT
41 000052  000207                             RTS     PC
42                                    ;--------------------------------------
43                                    ; ASCII STRINGS
44                                    ;--------------------------------------
45 000054     033    133    062       CLRSTR: .BYTE   ESC,'[,'2,'J,ESC,'[,'H,0
   000057     112    033    133
   000062     110    000
46 000064     033    133    067       FORSTR: .BYTE   ESC,'[,'7,LCM,0
   000067     155    000
47 000071     033    133    060       BAKSTR: .BYTE   ESC,'[,'0,LCM,0
   000074     155    000
48 000076     007                     BELL:   .BYTE   007
49                                    ;
50        000001                              .END
```

```
DISPLY.MAC     MACRO V04.00  31-MAY-83 16:02:57 PAGE 1-1
SYMBOL TABLE

BAKSCN  000014RG    002 BELSCN  000022RG    002 CLRSTR  000054R    002 FORSCN  000006RG  002 LCM   = 000155
BAKSTR  000071R     002 CLRSCN  000000RG    002 ESC   = 000033        FORSTR  000064R    002 OUT     000034R  002
BELL    000076R     002

. ABS.  000000      000
        000000      001
DISPLY  000077      002
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  8192 WORDS  ( 32 PAGES)
DYNAMIC MEMORY AVAILABLE FOR  54 PAGES
DISPLY,LP:=DISPLY
```

III.13   HDUMP

HDUMP.MAC   MACRO V04.00   31-MAY-83 16:04:36 PAGE 1

```
 1                                  ;*******************************************
 2                                  ;
 3                                  ;   HEX MEMORY DUMP PROGRAM
 4                                  ;           D.VOELZ
 5                                  ;
 6                                  ;*******************************************
 7                                  ;
 8                                  ;   THIS SUBPROGRAM DUMPS A SECTION OF THE LSI-11
 9                                  ;   MEMORY IN HEXADECIMAL FORM.
10                                  ;   --- LOAD R1 WITH STARTING MEMORY ADDRESS
11                                  ;   --- LOAD R2 WITH BYTE COUNT
12                                  ;
13                                          .TITLE HDUMP.MAC
14                                          .MCALL  .TTYOUT
15                                  ;
16                                  ;
17 000000 111103              HDUMP:: MOVB    (R1),R3         ;LOAD R3 WITH DATA BYTE
18 000002 004767  000020              JSR     PC,HEXOUT       ;OUTPUT BYTE IN HEX
19 000006                             .TTYOUT SPACE           ;OUTPUT SPACE
20 000016 005201                      INC     R1
21 000020 005302                      DEC     R2
22 000022 001366                      BNE     HDUMP           ;LAST BYTE ?
23 000024 000207                      RTS     PC
24                                  ;
25 000026 110346              HEXOUT: MOVB    R3,-(SP)        ;SAVE BYTE
26 000030 072327  177774              ASH     #-4,R3          ;SELECT HI 4 BITS
27 000034 004767  000010              JSR     PC,SEND         ;SEND IT
28 000040 112603                      MOVB    (SP)+,R3        ;RETRIEVE BYTE
29 000042 004767  000002              JSR     PC,SEND         ;SEND LO 4 BITS
30 000046 000207                      RTS     PC
31                                  ;
32 000050 142703  000360      SEND:   BICB    #360,R3         ;CLEAR HI 4 BITS (OF BYTE)
33 000054 120327  000012              CMPB    R3,#10.         ;VALUE LESS THAN 10 ?
34 000060 100403                      BMI     LT10            ;  -YES,GOTO LT10
35 000062 062703  000067              ADD     #<'A-10.>,R3    ;ADD BASE ALPHANUMERIC OFFSET
36 000066 000402                      BR      OUT
37 000070 062703  000060      LT10:   ADD     #'0,R3          ;ADD BASE NUMBER OFFSET
38 000074 110300              OUT:    MOVB    R3,R0
39 000076                             .TTYOUT                 ;OUTPUT CHARACTER
40 000102 000207                      RTS     PC
41                                  ;
42 000104    040              SPACE:  .ASCII  / /             ;ASCII SPACE
43                                  ;
44         000001                      .END
```

```
HDUMP.MAC      MACRO V04.00  31-MAY-83 16:04:36 PAGE 1-1
SYMBOL TABLE

HDUMP   000000RG       LT10    000070R      OUT    000074R      SEND   000050R      SPACE   000104R
HEXOUT  000026R

. ABS.  000000    000
        000105    001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  8192 WORDS  ( 32 PAGES)
DYNAMIC MEMORY AVAILABLE FOR  54 PAGES
HDUMP,LP:=HDUMP
```

III.14   CONVRT

FORTRAN IV   V02.5-5   Wed 03-Aug-83 00:06:50   PAGE 001

```
      C  ***************************************
      C
      C
      C  PROGRAM CONVRT.FOR - DATA CONVERSION
      C               6-APR-83
      C               D.VOELZ
      C
      C  ***********************************************
      C
      C  THIS PROGRAM CONVERTS BINARY FILES OF LIDAR
      C  DATA TO ASCII FILES. THIS IS NECESSARY FOR
      C  THE TRANSFER OF DATA TO THE CYBER.
      C  THE PROGRAM ALSO ALLOWS THE GROUND SPEED (KTS),
      C  ALTITUDE (FT), LATITUDE (DEG.-MIN), AND
      C  LONGITUDE (DEG.-MIN) TO BE ADDED TO THE ASCII
      C  FILE.
      C
      C  THE ASCII FILE FORMAT IS AS FOLLOWS:
      C
      C      WRITE(4,120)BINS,LBI,CSET,CPROF,MN,DAY,YR,HRS,MINS,SECS
      C  120 FORMAT(4I5,6I3)
      C
      C      WRITE(4,125)GSPD,ALT,LAT,LONG
      C  125 FORMAT(I5,F7.3,2F7.1)
      C
      C      WRITE(4,128)(LDATA(I),I=1,BINS)
      C  128 FORMAT(12(I5))
      C
      C  *** NOTE *** IN THE PROGRAM ALL THE FORMAT STATEMENTS ABOVE
      C               BEGIN WITH A '1X'. THIS IS SIMPLY THE CONTROL
      C               CHARACTER.
      C
0001          PROGRAM CONVRT
0002          INTEGER DSET,NPROF,GSPD(99),MN,DAY,YR,LDATA(2100)
0003          INTEGER HDRB,BINS,INTSIZ,LBI,RECSIZ,CSET,CPROF
0004          INTEGER HRS,MINS,SECS
0005          REAL ALT(99),LAT(99),LONG(99)
0006          LOGICAL*1 DEVF(4),DEVT(4),EXT(3),NAMEF(15),NAMET(15)
0007          LOGICAL*1 ITIME(8)
0008          INTEGER*4 JTIM
      C  ==============================
      C  INPUTS
      C  ==============================
      C
      C  INITIAL VALUES
      C
0009          HDRB=9
0010          DATA NAMEF/0,0,0,0,'S','E','T',0,0,0,'.','D','A','T',0/
0011          DATA NAMET/0,0,0,0,'S','E','T',0,0,0,'.','A','S','C',0/
      C
      C  GET INITIAL RESPONSES (DEVICE,SET #,# OF PROFILES)
      C
0012          TYPE 10
0013          READ(5,15) (DEVF(I),I=1,4)
0014          TYPE 20
```

```
FORTRAN IV      V02.5-5    Wed 03-Aug-83 00:06:50         PAGE 002
0015         READ(5,15) (DEVT(I),I=1,4)
0016      27 TYPE 30
0017         READ(5,*) DSET
0018         TYPE 40
0019         READ(5,*) NPROF
      C
0020      10 FORMAT(' ',T32,'DATA CONVERSION',//,/T5,'DEVICE:',/T5
             >,' From ? ',$)
0021      15 FORMAT(4A1)
0022      20 FORMAT(' ',T5,' To ? ',$)
0023      30 FORMAT('0',/T5,'SET # ? ',$)
0024      40 FORMAT(' ',T5,'TOTAL # OF PROFILES IN SET ? ',$)
      C
      C   INPUT GROUND SPEED,ALTITUDE,LATITUDE,LONGITUDE
      C
0025         DO 90 K=1,NPROF
0026         TYPE 50,DSET,K
0027         TYPE 52
0028         READ(5,*) GSPD(K)
0029         TYPE 56
0030         READ(5,*) ALT(K)
0031         TYPE 60
0032         READ(5,*) LAT(K)
0033         TYPE 64
0034         READ(5,*) LONG(K)
0035      90 CONTINUE
      C
0036      50 FORMAT('0',/T5,'SET #      ',I3,/T5,'PROFILE # ',I3)
0037      52 FORMAT(' ',T5,'Ground Speed (KTS) ?      ',$)
0038      56 FORMAT(' ',T5,'Altitude (FT) ?      ',$)
0039      60 FORMAT(' ',T5,'Latitude (DEG.-MIN.) ?  ',$)
0040      64 FORMAT(' ',T5,'Longitude (DEG.-MIN.) ? ',$)
      C ===============================
      C   CONVERT FILES
      C ===============================
      C
      C   SET UP FILE NAMES (SETXXX.DAT --> SETXXX.ASC)
      C
0041         DO 70 I=1,4
0042         NAMEF(I)=DEVF(I)
0043         NAMET(I)=DEVT(I)
0044      70 CONTINUE
      C
0045         ENCODE(3,80,EXT)DSET
0046      80 FORMAT(I3)
0047         DO 85 I=1,3
0048         NAMEF(7+I)='0'
0049         NAMET(7+I)='0'
0050         IF(EXT(I).EQ.' ') GOTO 85
0052         NAMEF(7+I)=EXT(I)
0053         NAMET(7+I)=EXT(I)
0054      85 CONTINUE
      C
      C CALCULATE RECORDSIZE FOR OLD FILE AND INITIAL BLOCK SIZE
```

```
FORTRAN IV      V02.5-5    Wed 03-Aug-83 00:06:50            PAGE 003

       C  FOR NEW FILE
       C
0055       OPEN(UNIT=3,TYPE='OLD',RECORDSIZE='2',NAME=NAMEF,ACCESS=
          >'DIRECT',FORM='UNFORMATTED')
0056       READ(3'1)BINS
0057       CLOSE(UNIT=3)
0058       RECSIZ=(BINS+HDRB+1)/2
0059       INTSIZ=IFIX((FLOAT(NPROF)*(BINS+15)/256.+1.1)*6.)
       C
       C  OPEN OLD & NEW FILES
       C
0060       OPEN(UNIT=3,TYPE='OLD',RECORDSIZE=RECSIZ,NAME=NAMEF,ACCESS=
          >'DIRECT',FORM='UNFORMATTED')
0061       OPEN(UNIT=4,TYPE='NEW',INITIALSIZE=INTSIZ,NAME=NAMET)
       C ---------------------------------
       C  CONVERT DATA BY PROFILES (K=PROFILE #)
       C ---------------------------------
0062       DO 100 K=1,NPROF
0063       READ(3'K)BINS,LBI,CSET,CPROF,MN,DAY,YR,JTIM,
          >(LDATA(I),I=1,BINS)
       C
       C  CONVERT ALTITUDE FROM FEET TO KILOMETERS
       C
0064       ALT(K)=ALT(K)*.0003048
       C
       C  CONVERT TIME
       C
0065       CALL TIMASC(JTIM,ITIME)
0066       ITIME(3)=ITIME(4)
0067       ITIME(4)=ITIME(5)
0068       ITIME(5)=ITIME(7)
0069       ITIME(6)=ITIME(8)
0070       DECODE(6,115,ITIME)HRS,MINS,SECS
0071   115 FORMAT(3I2)
       C
       C  WRITE ASCII FILE
       C
0072       WRITE(4,120)BINS,LBI,CSET,CPROF,MN,DAY,YR,HRS,MINS,SECS
0073   120 FORMAT(1X,4I5,6I3)
0074       WRITE(4,125)GSPD(K),ALT(K),LAT(K),LONG(K)
0075   125 FORMAT(1X,I5,F7.3,2F7.1)
0076       WRITE(4,128)(LDATA(I),I=1,BINS)
0077   128 FORMAT(1X,12(I5))
       C
0078   100 CONTINUE
       C ---------------------------------
       C  CLOSE OUT FILES
       C ---------------------------------
0079       CLOSE(UNIT=3)
0080       CLOSE(UNIT=4)
0081       TYPE 130,CSET
0082   130 FORMAT('0',T5,'SET # ',I3,' CONVERTED')
       C
       C  RETURN FOR NEXT SET
       C
0083       GOTO 27
       C
0084       END
```

FORTRAN IV   Storage Map for Program Unit CONVRT

Local Variables, .PSECT $DATA, Size = 013112 (2853. words)

Name	Type	Offset	Name	Type	Offset	Name	Type	Offset
BINS	I*2	013052	CPROF	I*2	013064	CSET	I*2	013062
DAY	I*2	013044	DSET	I*2	013036	HDRB	I*2	013050
HRS	I*2	013066	I	I*2	013100	INTSIZ	I*2	013054
JTIM	I*4	013074	K	I*2	013102	LBI	I*2	013056
MINS	I*2	013070	MN	I*2	013042	NPROF	I*2	013040
RECSIZ	I*2	013060	SECS	I*2	013072	YR	I*2	013046

Local and COMMON Arrays:

Name	Type	Section	Offset	Size		Dimensions
ALT	R*4	$DATA	010456	000614 (	198.)	(99)
DEVF	L*1	$DATA	012722	000004 (	2.)	(4)
DEVT	L*1	$DATA	012726	000004 (	2.)	(4)
EXT	L*1	$DATA	012732	000003 (	2.)	(3)
GSPD	I*2	$DATA	000000	000306 (	99.)	(99)
ITIME	L*1	$DATA	012773	000010 (	4.)	(8)
LAT	R*4	$DATA	011272	000614 (	198.)	(99)
LDATA	I*2	$DATA	000306	010150 (	2100.)	(2100)
LONG	R*4	$DATA	012106	000614 (	198.)	(99)
NAMEF	L*1	$DATA	012735	000017 (	8.)	(15)
NAMET	L*1	$DATA	012754	000017 (	8.)	(15)

Subroutines, Functions, Statement and Processor-Defined Functions:

Name	Type	Name	Type	Name	Type	Name	Type	Name	Type
FLOAT	R*4	IFIX	I*2	TIMASC	R*4				

APPENDIX IV   PREPROCESSING SYSTEM OPERATION PROCEDURE

IV.1   Data Collection

The following is the procedure for assembling and operating the lidar preprocessing system for sodium profile data collection.

1. Connect the components of the receiving system as shown in Figure IV.1. Note the following items:

   (a) Three cards are required in the Apple peripheral slots: the CCS serial card in slot 2, the DMA card in slot 4, and the disk controller card in slot 6.

   (b) Be sure to use the cable designated "Apple-LSI-11" between the Apple CCS card and the LSI-11 DLV11-J card channel 0.

   (c) Use the shielded ribbon cable between the Apple and the SI IPP unit.

2. Connect the monitor output of the Discriminator High Voltage Supply to a digital voltmeter.

3. Check the settings of the following components:

   Laser Control Unit - Trigger mode external

   Discriminator - single mode, $V = 1$, ΔV = don't care, threshold multiplier = X10

   PMT Timing Controller - blanking on, μP gate don't care, triggering external, internal rate don't care.

4. Check the setting of the Discriminator High Voltage Supply by first disconnecting the HV cable on the HV Output and then turning on the supply. The voltage should be adjusted until the monitor voltmeter reads about 2 volts (2000 volts HVS output). Shut off the supply and reconnect the HV cable.
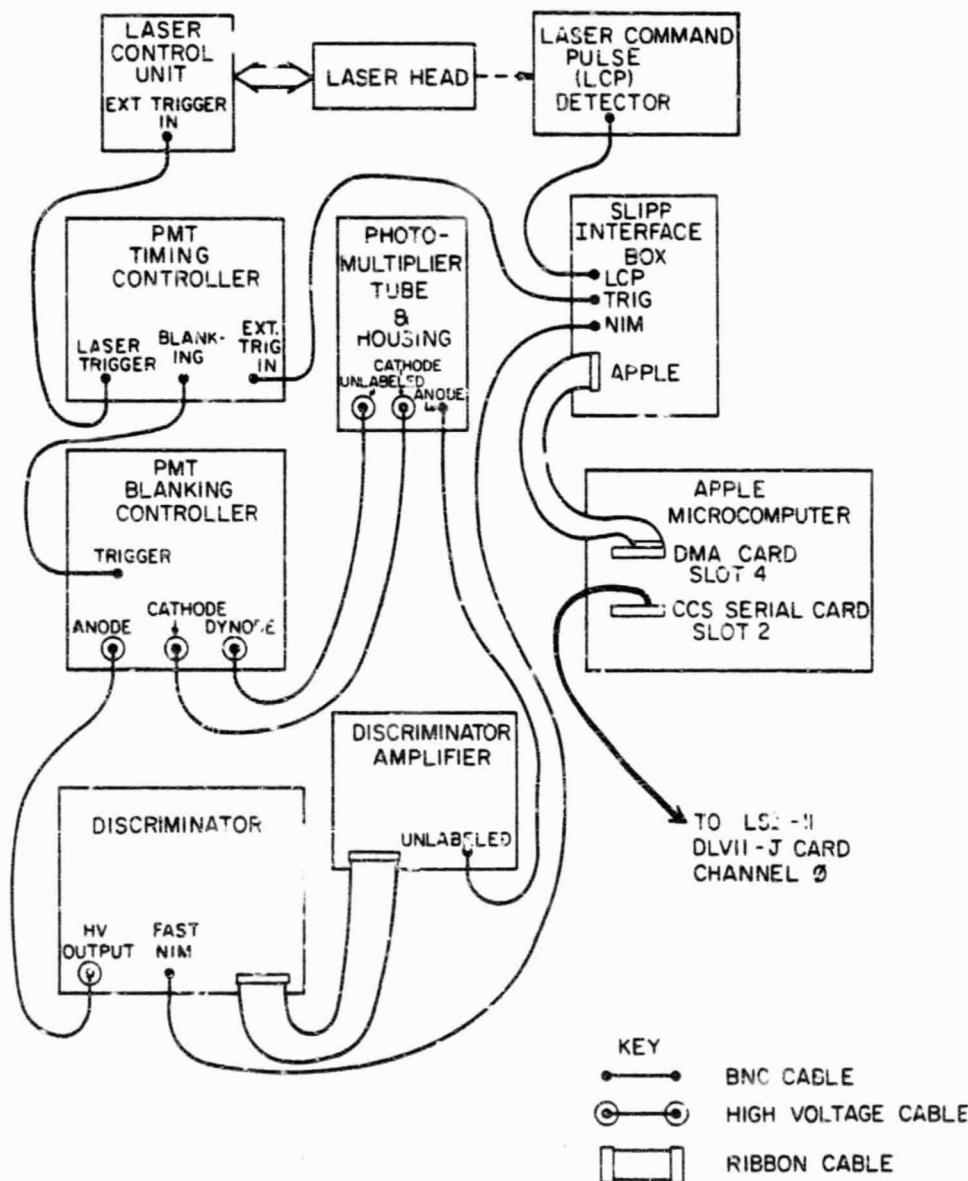
Figure IV.1 Lidar receiver connection map.

5. Power up all of the components except the Apple and LSI-11 computers. The order in which the components are switched on is not critical.

6. Load one of the Apple lidar software floppy disks into the Apple drive and switch on the Apple. Wait for the computer to boot up. Run the collection program SLPAPP.OBJ (BRUN SLPAPP.OBJ).

7. Load one of the LSI-11 lidar software floppy disks into the LSI-11 system drive and switch on the LSI-11. Wait for the computer to boot up. Load a LSI-11 data floppy disk into the storage drive. (Normally the DX1: or DY1: drive is used as the system drive and DX0: or DY0: is used as the storage drive. For more details see Section 5.4 on Data Collection Software Options.) Run the collection program FZZ.SAV (RUN FZZ).

8. The preprocessing system should now be operational. The laser-telescope alignment and data collection routines can be run by following the prompts on the Apple and LSI-11 monitors.

Notes:

(a) Remember to power up the PMT cooling housing unit a few hours before the data collection to allow the PMT to cool sufficiently.

(b) In order for the preprocessing system to trigger the laser the trigger mode selector switch must be set to "external." However, the trigger mode switch can be set to "internal" for laser tuning or alignment of optics.

(c) Two pulse delay circuits are provided on the laser control unit. The circuits provide a delay between an input trigger pulse and an output pulse with the duration of the delay set by an adjustment knob on the control unit. Normally, delay circuit #2 is incorporated between the input laser trigger pulse and the final

trigger pulse sent to the laser head. Thus, there is a delay
between the preprocessing system laser trigger pulse and the
actual laser firing. If the delay is too long, it is possible
for the Apple in the preprocessing system to send the laser
trigger pulse and then test to see if the collection is complete
before the laser ever fires. This results in a laser trigger
error message. If the laser is firing at approximately the same
time as the Apple is testing for collection completion, some
peculiar behaviors can result (such as the Apple "hanging").
These problems can be alleviated by reducing the delay with the
adjustment knob (normally, #2 ADJ).

IV.2 Testing

The following is a procedure for assembling and testing the prepro-
cessing system without the rest of the lidar system components.

1. Connect the preprocessing system components (Apple, LSI-11, and SLIPP
    unit) as shown in Figure IV.1. However, do not connect the three SLIPP
    unit lines LCP, TRIG, and NIM to the receiving components. (Also,
    note the appropriate items in Section IV.1.)

2. Connect the TRIG BNC output on the SLIPP unit to the LCP BNC input.
    This loop simulates the laser in the system.

3. Connect a source of simulated photon pulses to the NIM BNC on the SLIPP
    unit. The pulses can be supplied by a couple of methods:

    (a) Use the PMT and the Discriminator-Amplifier as shown in Figure
        IV.1 and simply leak a small amount of light into the PMT.

    (b) Use a pulse generator with output pulses of about -1 V and
        1-20 MHz.

4. Follow the steps outlined in Section IV.1 (steps 6-8) for loading the
   Apple and LSI-11 disks and running the collection programs. However, a
   slightly altered version of the Apple collection program must be used
   because of some timing difficulties due to the TRIG-LCP loop. This
   program is labeled SLPTST.OBJ and should also be provided on the Apple
   lidar software disks.

REFERENCES

Apple II Reference Manual [1979], Apple Computer Inc., Cupertino, CA.

Bowman, M. R., A. J. Gibson, M. C. W. Sanford [1969], Atmospheric sodium measured by a tuned laser, Nature, 221, 456-457.

California Computer Systems Apple II Asynchronous Serial Interface Model 7710A Owner's Manual [1979], California Computer Systems, Sunnyvale, CA.

DEC Microcomputer Interfaces Handbook [1980], Digital Equipment Corp., Maynard, MA.

Kinte, T. M. [1977], A Laser Radar System Interface, MS Thesis, Univ. Ill., Urbana-Champaign.

Newman, M. and D. Smith, The MC6809 in DMA mode on the IEEE-488 Bus, IEEE Micro, Nov. 1981, 56-70.

Richter, E. S. and C. F. Sechrist, Jr. [1978], Theoretical and experimental studies of the atmospheric sodium layer, Aeron. Rep. No. 79, Aeron. Lab., Dept. Elec. Eng., Univ. Ill., Urbana-Champaign.

Richter, E. S. and C. F. Sechrist, Jr. [1979], A meteor ablation-cluster ion atmospheric sodium theory, Geophys. Res. Lett., 6, 183-186.

Rowlett, J. R. and C. S. Gardner [1979], Signal processing of sodium lidar photocount data, R. R. L. Publication No. 504, Radio Res. Lab., Dept. Elec. Eng., Univ. Ill., Urbana-Champaign.

Scalon, L. J. [1980], 6502 Software Design, Howard W. Sams and Co., Inc., Indianapolis, IN.

Shelton, J. D. and C. S. Gardner [1981], Theoretical and lidar studies of the density response of the mesospheric sodium layer to gravity wave perturbations, Aeron. Rep. No. 99, Aeron. Lab., Dept. Elec. Eng., Univ. Ill., Urbana-Champaign.

Teitelbaum, K. and C. F. Sechrist, Jr. [1979], A microcomputer control system for the Urbana sodium lidar, _Aeron. Rep. No. 88_, Aeron. Lab., Dept. Elec. Eng., Univ. Ill., Urbana-Champaign.

Vogel, F. M. [1982], _A Lidar Microprocessor-Controlled Receiver/Data Collection System_, MS Thesis, Dept. Elec. Eng., Univ. Ill., Urbana-Champaign.

Wingfield, M. [1981], Build an intercomputer data link, _Byte_, April 1981, 252-288.

Zendt, F. T. and S. A. Bowhill [1982], A preprocessor for the Urbana coherent-scatter radar, _Aeron. Rep. No. 102_, Aeron. Lab., Dept. Elec. Eng., Univ. Ill., Urbana-Champaign.