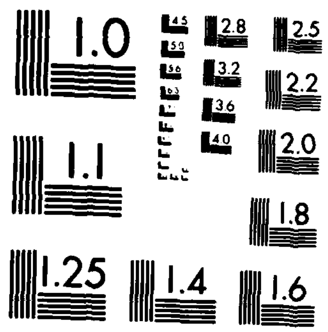
AD-A149 726 TOPICS IN OPTICAL MATERIALS AND DEVICE RESEARCH - III 1/2
VOLUME 1(U) PARKE MATHEMATICAL LABS INC CARLISLE MASS
T B BARRETT ET AL. AUG 84 RADC-TR-84-73-VOL-1
UNCLASSIFIED F19628-81-C-0052 F/G 17/8 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

(12)

RADC-TR-84-73, Vol I (of two)
Final Technical Report
August 1984

TOPICS IN OPTICAL MATERIALS AND DEVICE RESEARCH - III

Parke Mathematical Laboratories, Inc.

T. B. Barrett
R. Marshall
C. Warde
J. Caulfield
M. M. Salour

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-84-73, Volume I (of two) has been reviewed and is approved for publication.

APPROVED:

CARL A. PITHA
Project Engineer

APPROVED:

HAROLD ROTH, Director
Solid State Sciences Division

FOR THE COMMANDER:

JOHN A. RITZ
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (ESO) Hanscom AFB MA 01731. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be returned.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS N/A
2a. SECURITY CLASSIFICATION AUTHORITY N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) N/A	5. MONITORING ORGANIZATION REPORT NUMBER(S) RADC-TR-84-73, Volume I (of two)
6a. NAME OF PERFORMING ORGANIZATION Parke Mathematical Laboratories, Inc. 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Rome Air Development Center (ESO)
6c. ADDRESS (City, State and ZIP Code) 1 River Road Carlisle MA 01741	7b. ADDRESS (City, State and ZIP Code) Hanscom AFB MA 01731
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Rome Air Development Center 8b. OFFICE SYMBOL (If applicable) (ESO)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F19628-81-C-0052
8c. ADDRESS (City, State and ZIP Code) Hanscom AFB MA 01731	10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
62702F	4600	19	36

11. TITLE (Include Security Classification)
TOPICS IN OPTICAL MATERIALS AND DEVICE RESEARCH - III

12. PERSONAL AUTHOR(S)
T. B. Barrett, R. Marshall, C. Warde, J. Caulfield, M. M. Salour

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Final	FROM Dec 80 TO Dec 83	August 1984	130

16. SUPPLEMENTARY NOTATION
N/A

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	Indium phosphide, Infrared device, Electro-optic substrate, Optical signal processing, Bismuth silicon oxide, Photorefractive effect
20	02, 06		
	12, 14		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

1. The effect of various crystal growth procedures on the semi-conducting properties of indium phosphide is summarized.

2. The physical basis for the photorefractive effect in bismuth silicon oxide is investigated.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code)	22c. OFFICE SYMBOL
Carl Pitha	(617) 861-3488	RADC (ESO)

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

FOREWORD

This report is the first part of a two-part final report for Contract F19628-81-C-0052. Part II is an addendum to RADC-TR-81-372, Vol. II describing a GPIB (IEEE - 488) interface for a Nicolet 1080 Computer.

The following individuals have contributed to results reported on in these reports:

Robert C. Marshall	Consultant to PML
Cardinal Warde	Consultant to PML
John Caulfield	Consultant to PML
Michael M. Salour	Consultant to PML
Theodore B. Barrett	Parke Mathematical Laboratories

Table of Contents

PREFACE (to Volume I)

This report, composed of 6 chapters, provides a partial summary of work accomplished in four diverse areas associated with optical materials and optical signal processing. Much of the work accomplished under this contract has been reported on elsewhere in the form of Technical Memoranda, conference proceedings and journal articles.

Chapter 1 describes experimental techniques used for the evaluation of a substrate (polycrystalline indium phosphide) which is used in optical signal processing devices. Results of the experiments are also reported.

Chapter 2 describes attempts to experimentally isolate the origin of the photorefractive effect in the important photorefrative material, $Bi_{12}SiO_{20}$.

Chapter 3 describes an optical technique for unidirectional light switching. Other research in this field which has been reported elsewhere and partially supported by this contract can be found as follows:

"Ultrafast Picosecond Chronography", J.S. Fujimoto and M.M. Salour, Proc. of SPIE, Vol. 322 Picosecond Lasers and Applications, 137, (1982)

"Optically Pumped Semiconductor Platelet Lasers", C.B. Roxlo, R.S. Putnam, M.M. Salour, Proc. of SPIE, Vol. 322, Picosecond Lasers and Applications, 31, (1982)

"Photon-Counting Statistics of Pulsed Light Sources", R.S. Bonderant, P. Kumar, J.H. Shapiro, M.M. Salour, Opt. Lett. 7, 529, (1982)

"Picosecond Laser-Spectroscopy Measurement of Hydroxyl Fluorescence Lifetime in Flames", U.S. Bergano, P.A. Jaanimogi, M.M. Salour, J.H. Bechtel, Opt. Lett. e, 443, (1983) "Optically Pumped Semiconductor Platelet Lasers", C.B. Roxlo, R.S. Putnam, M.M. Salour, IEEE J. Quantum Electron. QE-13, 101, (1982)

"Poloriton-Induced Compensation of Pulse Broadening in Optical Fibers", G.W. Fehrenbach, M.M. Salour, Appl. Phys. Lett. 41, 4, (1982)

"Tunable cw Bulk Semiconductor Platelet Laser", C.B. Roxlo, D. Balelaar, M.M. Salour, Appl. Phys. Lett. 38, 507, (1981)

"Optically Pumped cw Semiconductor Ring Laser", A. Fuehs, D. Bebelaar, M.M. Salour, Appl. Phys. Lett. 43, 32, (1983)

"Broadly Tunable Mode-Locked HgCdTe Lasers", R.S. Putnam, M.M. Salour, T.C. Harman, Appl. Phys. Lett. 43, 408, (1983)

"Synchronously Pumped Mode-Locked CdS Platelet Laser", C.B. Roxlo, M.M. Salour, Appl. Phys. Lett. 38, 738, (1981)

"Picosecond Spectroscopy of Bound Excitors in CuCl using a Synchronously Operating Streak Camwera", J.S. Fujimoto, T.X. Yee, M.M. Salour, Appl. Phys. Lett. 39, 12, (1981)

"Dewar Design for Optically Pumped Semiconductor Ring Laser", A. Fuchs, M.M. Salour, Rev. Sci. Instrum. 54, 1143, (1983)

"Dewar Design for Optically Pumped Semiconductor Lasers", C.B. Roxlo, M.M. Salour, Rev. Sci, Instrum. 53, 458, (1982)

Chapters 4 and 5 describe two applications of optical processing to numerical problems - polynomial evaluations and polynomial root searching.

Chapters 6 and 7 describe procedures which were used to down load data, gathered by instruments used in spectroscopy, for processing by a large central computer.

Chapter 1

PREPARATION AND EVALUATION OF POLYCRYSTALLINE INDIUM PHOSPHIDE FOR USE AS A SUBSTRATE FOR INFRARED DEVICE STRUCTURES

Introduction

Air Force interest in indium phosphide (InP) stems from a requirement for a lattice-matched electro-optic substrate material for 1.1 to 1.6 um fiber optic sources and detectors. Indium phosphide is also considered a promising substrate material for optical signal processing devices such as mode-locked lasers, integrated lasers/modulators and optoelectronic switches. Single crystals grown for these applications by the liquid encapsulated Czochralski (LEC) technique require polycrystalline starting materials of the highest purity. Reduction of residual donor impurities in the polycrystalline material is essential for the growth of semi-insulating crystals or P-type materials with low carrier concentrations.

In order to determine which parameters or combination of parameters provided polycrystalline material with the highest purity and lowest silicon contamination, ingots of indium phosphide were synthesized using the direct reaction technique under various temperatures, pressures, boat-tube materials and in-situ vacuum baking of the elemental indium. A range of synthesis temperature profiles were investigated to determine the effect on mobility, carrier concentration, grain size, homogeneity and stoichiometry. Experiments were conducted using quartz and pyrolytic boron nitride boats interchangeably with boron nitride and aluminum oxide tubes to determine if silicon was leaching from the quartz (SiO_2) boats and ampoule during the synthesis process.

The experimental program conducted at ESM consisted of four major experimental procedures; High Pressure Synthesis, Low Pressure - Low Indium Temperature, Low Pressure - High Indium Temperature, Low Pressure with Heat Pipe, and Low Pressure with Heat Pipe with Vacuum Baking. The highest purity material was obtained using a low phosphorus pressure with heat pipe and vacuum baking the elemental indium.

Experimental Procedure

A typical loaded quartz ampoule used in the direct reaction synthesis is shown in Figure 1. The diameter is 41 mm with length of 69 cm. All quartzware are soaked and washed in deionized water with 2% Deconex for 1 to 2 hours, rinsed with de-ionized water and air dried. The boat and plug are loaded in an ampoule, placed in a 3-zone furnace, and heated for one hour at 300°C. The ampoule is then connected to the vacuum system, heated to 1000°C and baked for 3 hours at 10^{-7} Torr. After cooling, the ampoule is disconnected from the vacuum and loaded at the closed end with the red phosphorous. A long stemmed funnel is used to prevent phosphorous from sticking to the sides of the ampoule. The boat loaded with indium is placed in a specific position in the ampoule, as determined by the temperature profile. The quartz plug is placed in the ampoule and sealed under vacuum. The indium is used directly from the sealed package.

A new method of sealing was developed which utilized a special quartz plug to fit inside the quartz ampoule. A stainless steel adapter was fabricated to couple the ampoule and vacuum system. A sketch of the vacuum system is shown in Figure 2. While under vacuum, the ampoule wall around the plug is heated until it collapses on the plug and seals. This completely eliminates any contamination from the flame or surrounding ambient. Heating during synthesis is accomplished by two Lindberg furnaces with triac type power supplies. The temperature control liners (Figure 3), are used to maintain long flat zones over specific zones in the furnace system. The isothermal liners purchased from Dynatherm Corporation are heat pipes filled with either sodium or potassium for maintaining desired temperatures over the entire pipe length. The heat pipe for the indium zone of the furnace is charged with sodium and has an operating range of 500° to 1100°C. The phosphorous zone heat pipe is charged with potassium and it has a range of 400° to 1000°C. The heat pipes are made of Inconel 600 with very good oxidation resistance (see Table 1).

The whole furnace system is mounted inside a hood on a motor driven table. The furnaces are moved at a specific rate in relation to the ampoule which is held stationary. The ampoule is firmly held in place by a stainless steel tube attached to an eye on the end of the ampoule. A thermocouple is inserted through the tube for the purpose of monitoring the red phosphorous temperature. Another thermocouple is placed against the opposite end of the ampoule to monitor the indium temperature. The indium and phosphorous used in these experiments were 6 N's purity purchased from Metal Specialties, Inc.

The over-all program to synthesize high-purity indium phosphide is comprised of four separate approaches: (1) synthesis at a high pressure of 27.5 atmospheres;[1,2,3] (2) a low indium temperature, low pressure system;[4] (3) a low pressure high indium temperature system; (4) a low pressure, standard temperature profile using heat pipes; and in-situ vacuum baking[5].

HIGH PRESSURE SYNTHESIS

A sketch of the temperature profile, ampoule and furnace arrangement for the high pressure experiments is shown in Figure 4. The indium temperature ranged from 1070° to 1150°C and the phosphorous temperature was maintained at 546°C. The solid line in the temperature profile shows the initial furnace configuration. An additional winding was added to the furnace to decrease the temperature drop at each end, extending the melt zone (dotted line). Temperature fluctuations were effectively reduced with an end cap in the indium end and use fluctuations were effectively reduced with an end cap in the indium end and use of Fiberfrax in the phosphorous end. The time period for these experiments was 2 to 3 days. Typical values for mobility and carrier concentration at 77K were 25,200 cm^2/V sec and 3.5×10^{15} carriers/cm^3. The possibility of explosion at these high pressures and our inability to obtain the material purity desired precluded further experiments in favor of low pressure synthesis techniques.

LOW PRESSURE, LOW INDIUM TEMPERATURE

The temperature profile used during the initial low pressure experiments, together with a loaded ampoule in its approximate starting position, is shown in Figure 5. The circled numbers on the abscissa indicate the various controls for the heater windings. The front of the boat containing the indium is placed at the start of the down slope. The red phosphorous zone must be at least as long as the boat because of the traveling profile. Ampoule pressure is maintained through accurate control of the phosphorous temperature.

The indium and phosphorous temperatures were varied from 945° to 1055°C and 412° to 520°C, respectively. The traveling rate of the furnace was 12 mm per day. The quart boats were 150 mm long and 25 mm wide. The standard ampoule charge was 150 g of 6 N's indium and 45 g phosphorous. The highest purity material synthesized using quartz boats had a carrier concentration of 3.16×10^{15} carriers/cm^3 and a liquid nitrogen mobility of 38,912 cm^2/V-sec.

LOW PRESSURE, HIGH INDIUM TEMPERATURE

A number of indium phosphide synthesis experiments were conducted using a temperature profile (Figure 6) with a very narrow high temperature zone. The over-all temperature of the quartz ampoule and boat were kept at relatively low temperatures and a sharp temperature spike 12 mm wide was programmed in an attempt to increase the purity of the indium phosphide by reducing silicon contamination. The front of the boat was placed at the peak of the high temperature zone at the start of the experiments. The indium zone of the furnace was varied from 1054° to 1080°C. The phosphorous zone ranged from 430° to 469°C. With this temperature spike profile, only a small portion of the quartz ampoule and boat would be exposed to the high temperature at any period of time. Furnace travel was maintained at 12 mm per day. The highest purity material synthesized using this temperature profile was obtained using an indium temperature of 1060°C and a phosphorous temperature of 434°C.

This material had a carrier concentration of 3.13 X 10^{15} carriers/cm^3 and a liquid nitrogen mobility of 31,631 cm^2/V-sec. However, since the average carrier concentration and liquid nitrogen mobility for a series of eight experiments was 6.38 X 10^{15} carriers/cm^3 and only 14,014 cm^2/V-sec, respectively, it was decided to discontinue using this method.

LOW PRESSURE WITH HEAT PIPE

A new synthesis system was designed using 2 Lindberg furnaces and 2 heat pipes. The temperature profile was programmed as shown in Figure 7. Energy to the system is provided by two triac type power supplies with associated Eurotherm controllers and ramp generators. These units, together with the heat pipes, give 2 long flat heat zones with a desired sharp temperature slope between the zones.

During the initial start-up period, it is always important to maintain the indium at a higher temperature than the phosphorous. The indium furnace is programmed up at 100°C per hour. When a temperature of 700°C is reached, the phosphorous furnace is turned on and programmed at the same rate. This rate of increase is continued until the indium temperature reached 1003°C and the phosphorous maintained at 430°C to 465°C. The furnaces are allowed to equilibrate overnight before furnace travel is initiated.

Experiments were designed around various quartz ampoule sizes and shapes and different boat materials. Boats were fabricated of quartz and pyrolytic boron nitride. In some experiments, the boats were inserted into boron nitride or alumina tubes to isolate them from the quartz ampoule. Travel rate of the furnaces for these experiments was 12 mm per day. The length of time for each experiment was 15 days. A typical polycrystalline indium phosphide ingot resulting from these experiments is shown with bottom, side, and top view in Figure 8. The right side is the first to freeze. The weight of these polycrystalline ingots ranged from 150 g to 400 g. An etched slice from one of these ingots showing typical single crystal grains used for the van der Pauw measurements is shown in Fugure 9. The highest purity material produced using these procedures was with an indium temperature of 1003°C, phosphorous temperature of 465°C using a quartz boat without a shielding tube. The resulting material had a carrier concentration of 1.8×10^{15} carriers/cm^3 and a mobility of 48,235 cm^2/V-sec at 77K.

LOW PRESSURE WITH HEAT PIPE AND VACUUM BAKING

Vacuum baking of raw indium was investigated to determine which temperature would provide materials with the highest purity and lowest residual carrier concentration. Baking procedures have been reported by Yamamoto et al[6] for the purification of In for InP synthesis and by Holms and Kamath[7], and Groves and Plonko[8], for purifying In melts for liquid phase epitaxy.

Experiments were conducted using Amersil quartz ampoules and boats. The indium and red phosphorous used was six nines purity and purchased from Metal Specialities, Inc. (515 Kings Highway, Fairfield, CT 06430, USA). Red phosphorous packaged in screw top vials rather than in sealed ampoule was preferred for ease of handling.

A Trans Temp (Trans Temp Company, 155 Sixth Street, Chelsea, MA 02150, USA) see-through furnace is used for all vacuum baking experiments. Power is applied to the Trans Temp furnace after a vacuum of 1×10^{-4} Torr is obtained. Only that part of the ampoule which contains the boat loaded with the indium is in the furnace. The end of the ampoule that contains the red phosphorous extends beyond the furnace heat zone and is cooled to prevent vaporization of the red phosphorous. The furnace temperature is raised to the desired baking temperature in one-half hour. For the first one to two hours of baking, the ampoule is evacuated to 1×10^{-4} Torr using cryogenic sorption pumps. After pumping for one to two hours, the vac ion pump is opened to the ampoule. Baking continues for a total of 6 h at which point a vacuum of 5×10^{-6} Torr is achieved. After this length of time, the power to the furnace is turned off and the ampoule is left to cool while still connected to the vac ion pump. While still under vacuum, the ampoule is sealed (1 X 10^{-7} Torr) by heating the ampoule wall around the internal sealing plug until it collapses on the plug. This completely eliminates any contamination from the flame or surrounding ambient.

Electrical Measurements

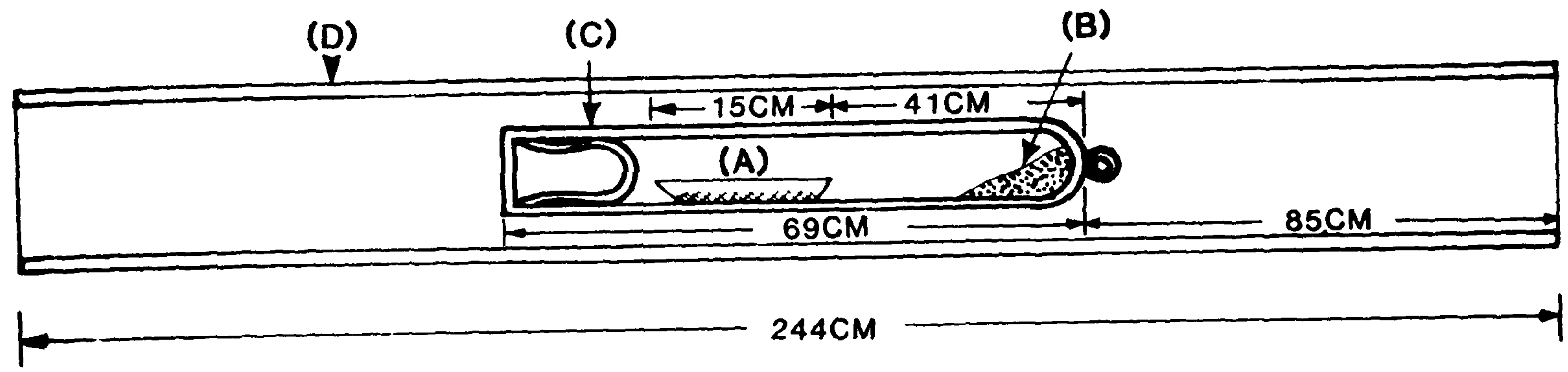
Slices 20 mils thick were cut from the ingot. 77 K van der Pauw measurements of the carrier concentration and mobility, made on 100-150 mil square samples cut from large single crystal grains in these slices, were used as an indication of the purity of the synthesized ingots. A tabulation of the baking conditions used and the electrical properties of the best sample from each ingot is given in Table 1. Figure 10 is a plot of 77 K mobility versus distance along the ingot from the first-to-freeze end and also shows a top view of the ingot. The vertical lines on the ingot show the position from which the slices (which yielded the van der Pauw samples) were cut. Each line in Figure 10 corresponds to one of the baking conditions shown in Table 2.

Conclusions of This Experimental Series

The highest purity polycrystalline indium phosphide material synthesized in all the experimental techniques resulted from using a standard temperature profile (Figure 7), utilizing heat pipes, quartz boats and in-situ vacuum baking of the indium. An indium temperature of 1003°C and a phosphorous temperature of 465°C was used.

Results clearly indicate that in-situ vacuum baking of the In prior to synthesis, at a temperature of 850°C and above, significantly improves the electrical properties of the resulting InP. Mass spectrometric analysis on these samples is inconclusive due to the low concentrations of the impurities involved. The carrier concentration and mobility measurements reported are, however, within the limits of published InP electrical and compensation ratio data[9]. There is a possibility that the high mobilities reported here could be "anomolously high" due to indium inclusions[10]. Van der Pauw measurements at high magnetic fields and photoluminescence measurements are now in progress to check this possibility.

The use of other boat materials and shielding tubes for elimination of silicon contamination from the quartz did not provide any measurable increase in purity as indicated from the van der Pauw measurements.



(A) INDIUM

(B) PHOSPHORUS

(C) QUARTZ AMPOULE *

(D) 244CM QUARTZ SUPPORT TUBE

* AMPOULE PLACEMENT DEPENDENT ON TEMPERATURE PROFILE

FIGURE 1 AMPOULE PLACEMENT

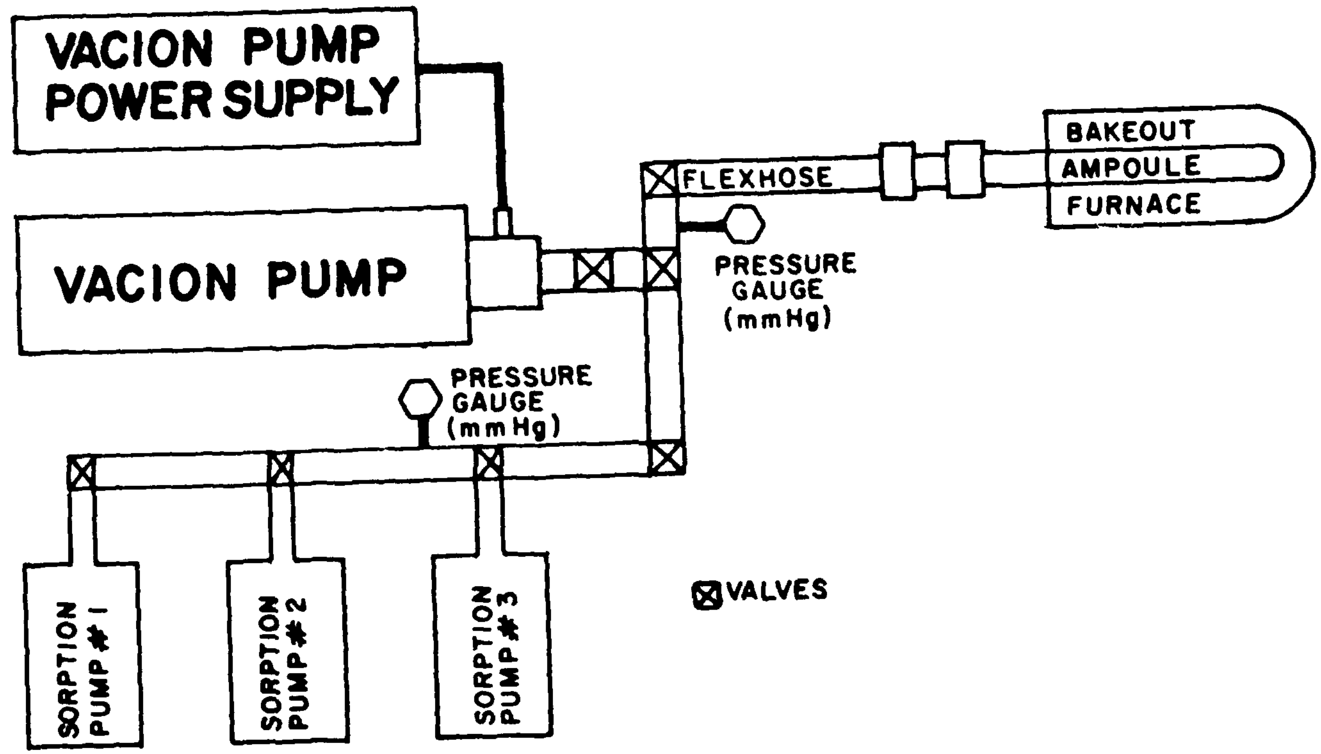
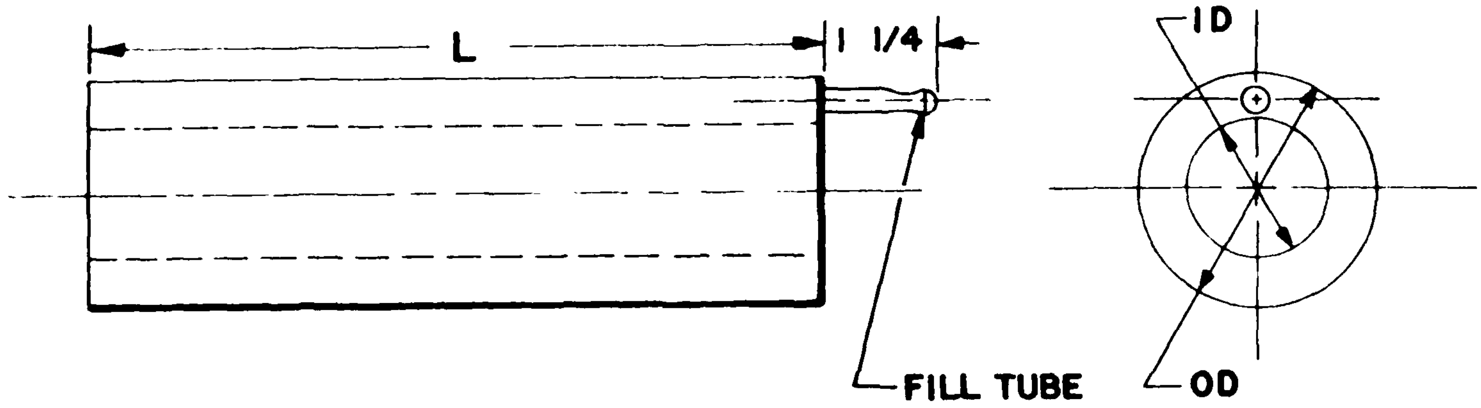


FIGURE 2 SKETCH OF VACUUM SYSTEM

ISOTHERMAL FURNACE LINER



Provides uniform zone with single heater
Temperature adjustment one step process
Frequent profile measurements not necessary

A flat profile is inherent to the liner
Temperature uniformity is within 1/2°C over liner length

Usable reaction zone length becomes equal to or larger than the heater length
Provides absolute temperature uniformity over the entire length and circumference of the tube furnace wall

* U.S. Patent No. 3,6777,329

FIGURE 3 ISOTHERMAL FURNACE LINER

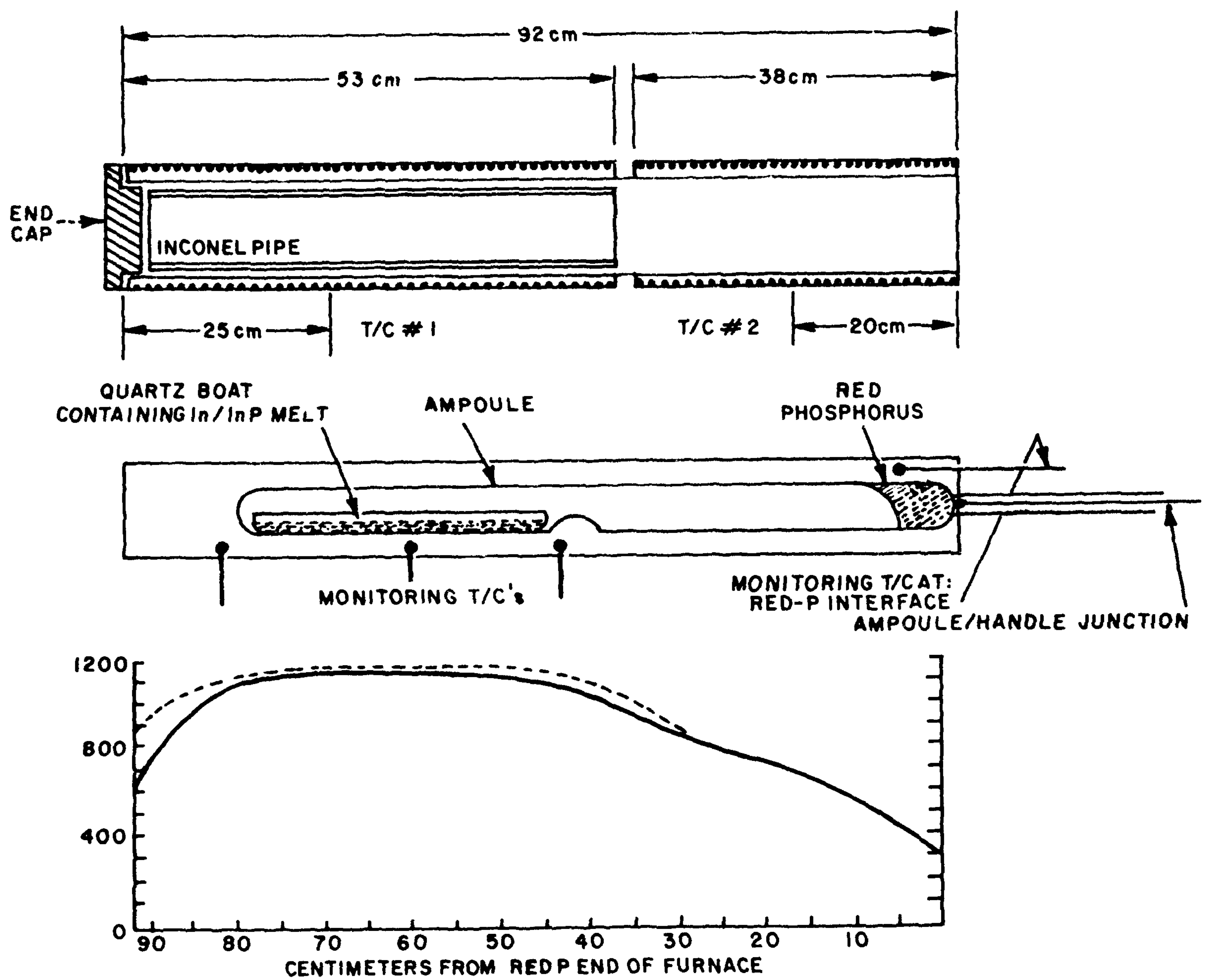


FIGURE 4 SKETCH OF HIGH TEMPERATURE SYNTHESIS FURNACE

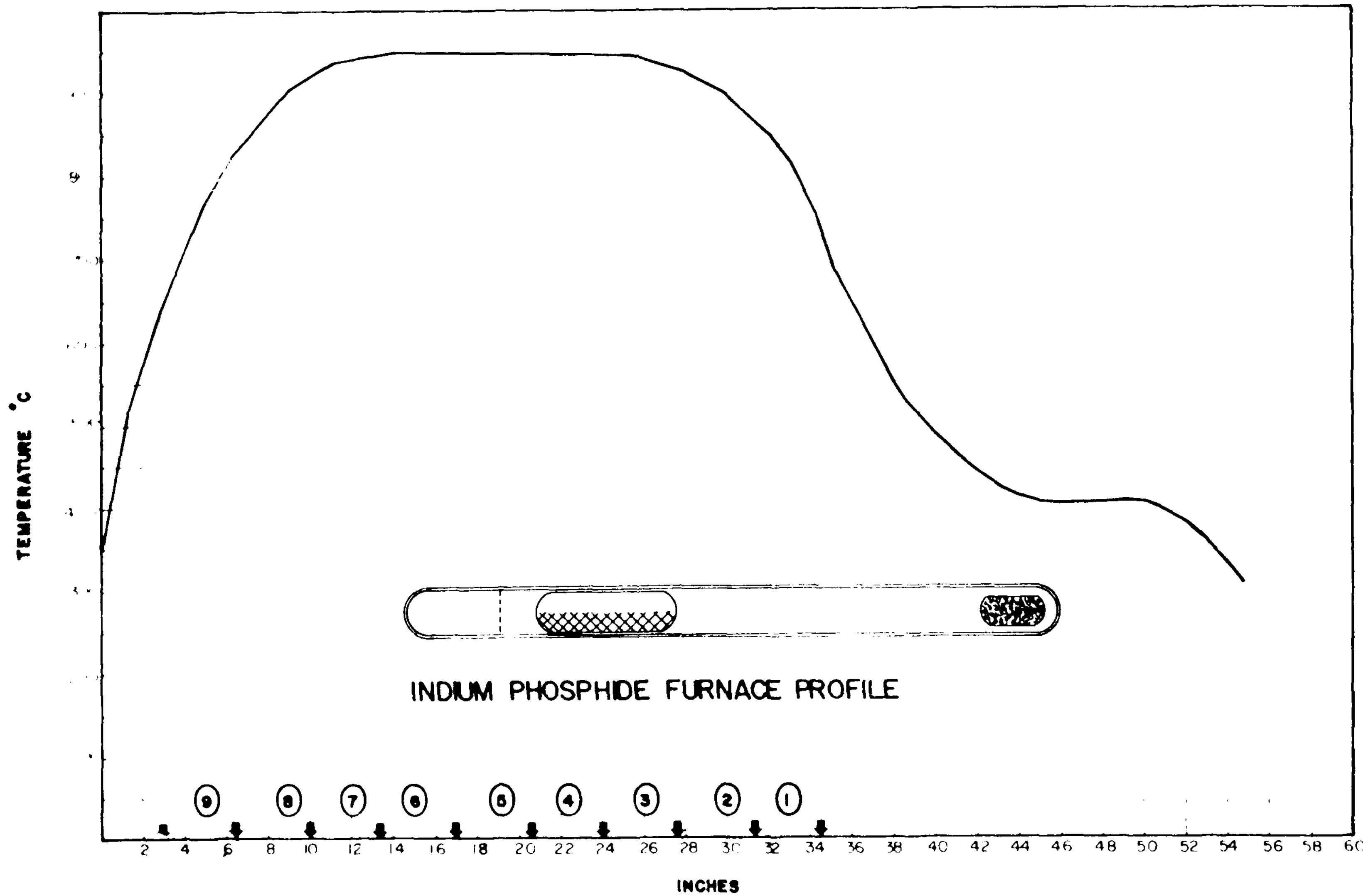


FIGURE 5 INDIUM PHOSPHIDE FURNACE PROFILE

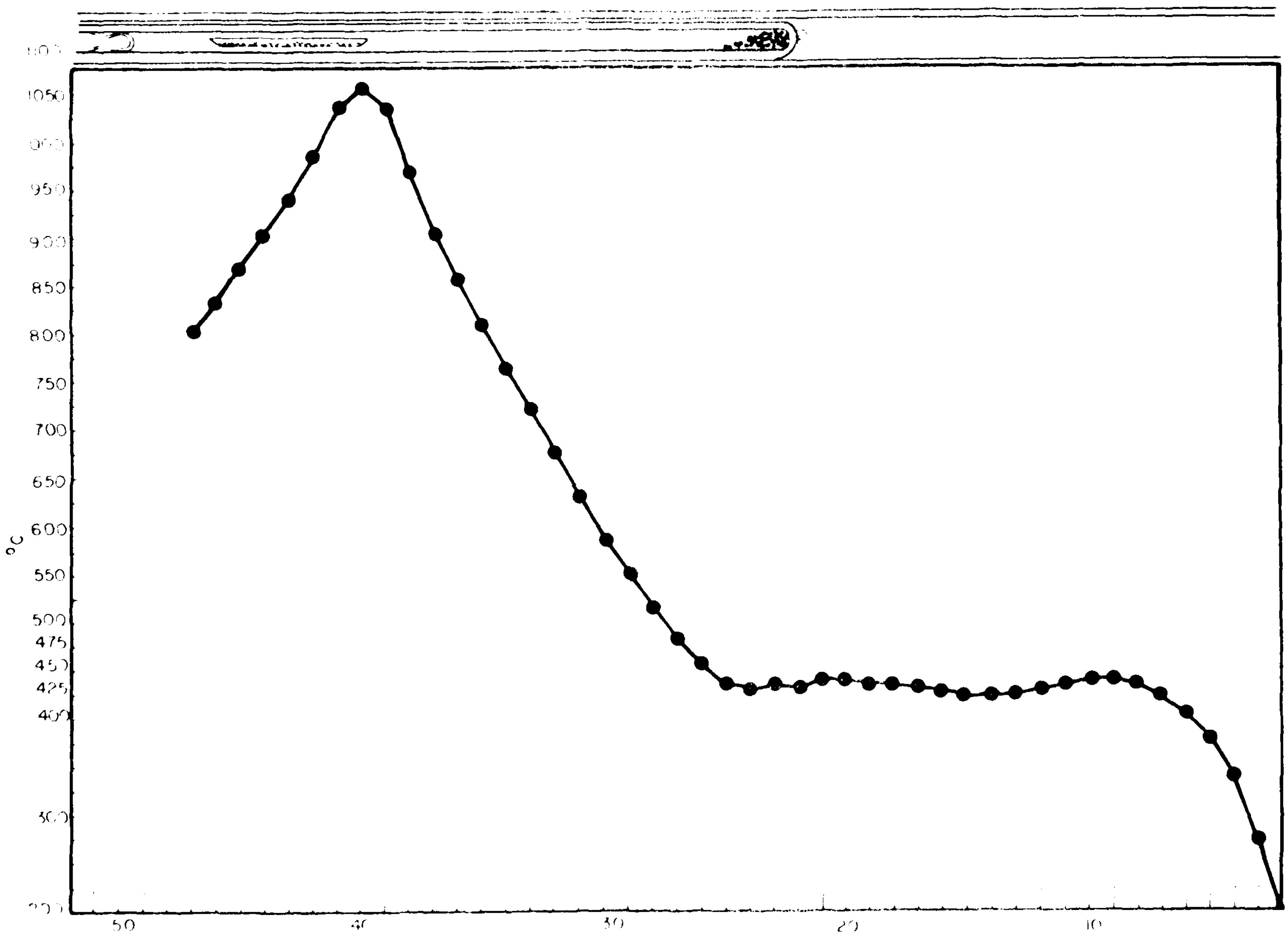


FIGURE 6 SPIKE TEMPERATURE PROFILE

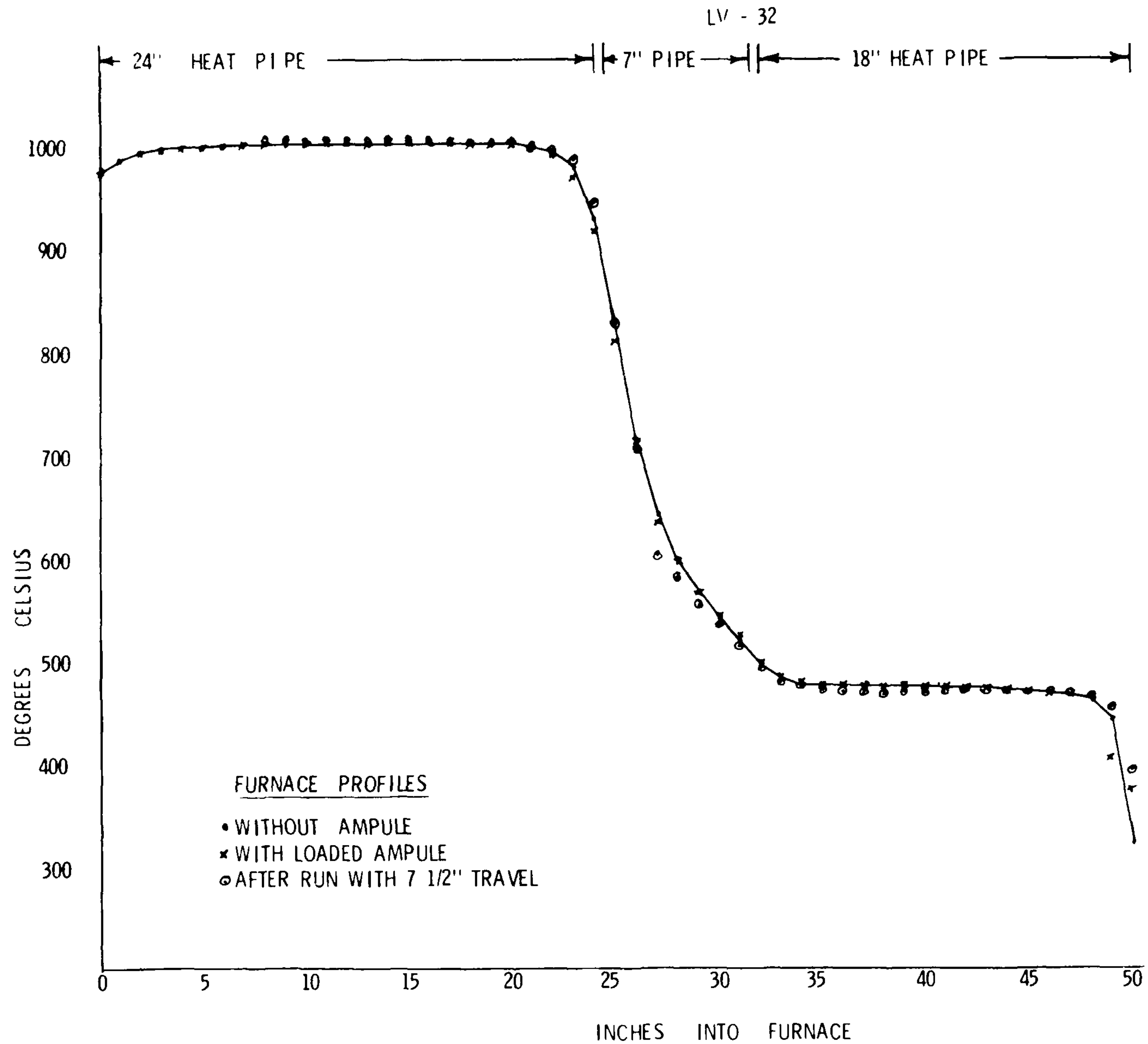


FIGURE 7 TEMPERATURE PROFILE (HEAT PIPES)

SYNTHESIZED INDIUM PHOSPHIDE

FIGURE 8 POLYCRYSTALLINE InP INGOT (BOTTOM-SIDE-TOP)

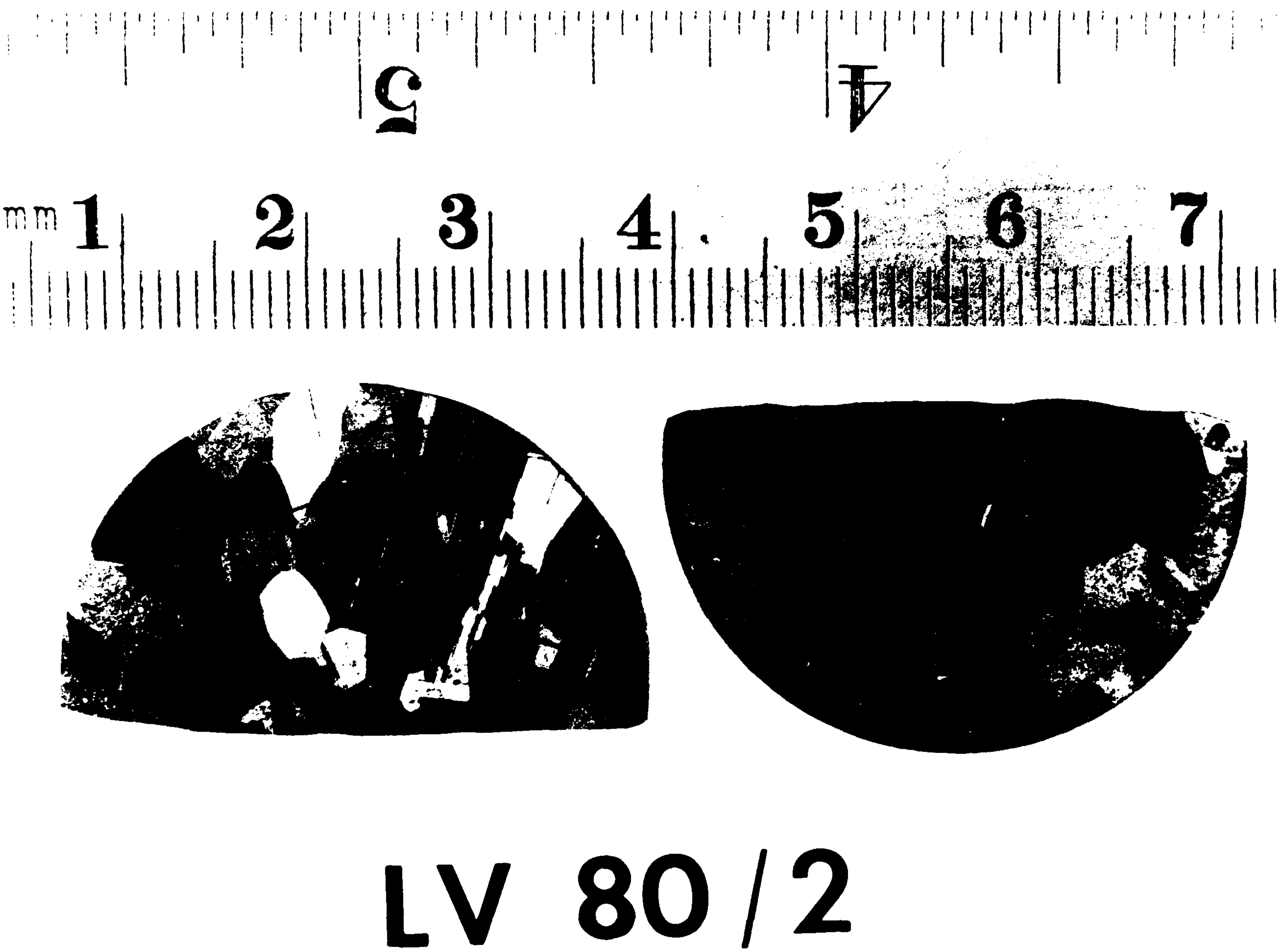


FIGURE 9 POLYCRYSTALLINE InP SLICES

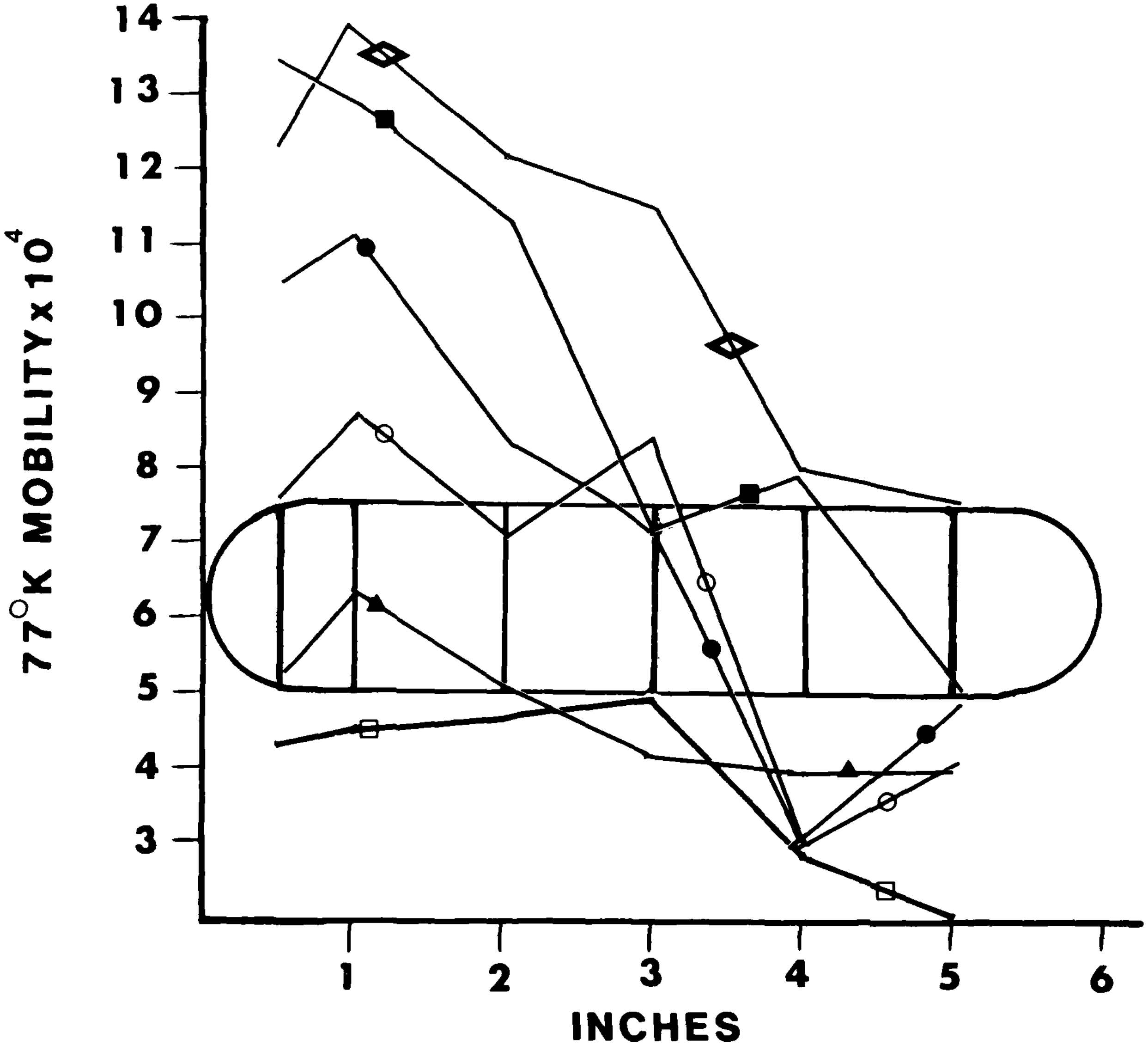


FIGURE 10 **MOBILITY vs. INGOT LENGTH**

ISOTHERMAL FURNACE LINERS**			
MODEL No. DESIGNATION	OPERATING TEMPERATURE °C		FURNACE LINER CHARGE
	MAXIMUM	MINIMUM	
3 - XX - XX	350	200	MERCURY
* 10 - 20 - 18	1000	400	POTASSIUM
δ 11 - 20 - 24	1100	500	SODIUM

* @ 970°C - MIN. 2 YR. OPERATION/@ 1000°C - MIN. 1000 HRS.

δ @ 1040°C - MIN. 2 YR. OPERATION/@ 1100°C - MIN. 1000 HRS.

** INCONEL ALLOY 600
(NICKEL-CHROMIUM-IRON)
OXIDATION RESISTANCE AT
HIGH TEMPERATURE ABOVE 1000°C

TABLE 1 ISOTHERMAL FURNACE LINER DATA

Temp. °C	/ Time(hrs.)	h	μ
▲ 800	6	1.3×10^{15}	64,000
○ 848	4	8.5×10^{14}	85,500
■ 850	6	2.0×10^{14}	134,000
◇ 900	6	3.0×10^{14}	138,500
● 950	6	5.6×10^{14}	110,000
□ not baked		2.0×10^{15}	47,000

BAKING CONDITIONS vs. 77°K ELEC. DATA

TABLE 2

REFERENCES

1. Klausutis, N., Adamski, J.A., and Sampson, J.L. (1976) Synthesis of Indium Phosphide, RADC-TR-76-305, ADA035507.

2. Fauth, T.A., and Adamski, J.A. (1979) High Pressure Synthesis of Stoichiometric Indium Phosphide, RADC-TR-79-246, ADA081875.

3. Fauth, T.A., and Adamski, J.A. (U.S. Patent No. 4, 185, 081).

4. Adamski, J.A., Low Pressure Synthesis of Indium Phosphide, (1982) RADC-TR-82-104.

5. J.A. Adamski, J. Crystal Growth 60 (1982) 141.

6. A. Yamamoto, S. Shinoyama and C. Uemura, J. Electrochem, Soc. 128 (1981) 585.

7. D.E. Holmes and G.S. Kamath, J. Crystal Growth 54 (1981) 51.

8. S.H. Groves and M.C. Plonko, J. Crystal Growth 54 (1981) 81.

9. W. Walukiewicz, J. Lagowski, L. Jastrzewbski, P. Rava, M. Liehtensteiger, C.H. Gatos and J. Appl. Phys. 51 (1980) 2659.

10. C.M. Wolfe, G.E. Stillman, D.L. Spears, D.E. Hill and F.V. Williams, J. Appl. Phys. 44 (1973) 732.

Chapter 2
CHARACTERIZATION OF DEFECTS IN $Bi_{12}SiO_{20}$

Abstract

Although significant advances have been made in the development and application of photorefractive devices employing bismuth silicon oxide ($Bi_{12}SiO_{20}$) the origin of the photorefractive effect in this material has received very little attention and, consequently, is poorly understood. The goal of this portion of the research program has been to identify and characterize the origin of the photorefractive effect in $Bi_{12}SiO_{20}$. Our preliminary results indicate that oxygen vacancies are not the primary defects giving rise to the photorefractive effect. We postulate that either silicon vacancies or impurity-ion ionization or both is the source of free carrier generation in this material.

Introduction

A material is said to exhibit the photorefractive effect if its refractive index changes upon exposure to optical radiation. Photorefractive materials are essentially electro-optic photoconductors operating at photon energies below the band gap of the material.

In the model for the photorefractive effect, subband gap light incident on the material excites electrons from traps (e.g. lattice defects) into the conduction band. Impurity-ion ionization can also be an important source of the free electrons, the result is that these electrons migrate through the crystal, by drift or diffusion, until they are eventually retrapped in the low-level intensity areas of the crystal. The space-charge electric fields set up by this non-equilibrium charge distribution then modulates the refractive index of the crystal via the electro-optic effect.

Information is generally stored holographically in a photorefractive material. It is therefore desirable that the characteristic drift or diffusion lengths, $\mu\tau E_o$ and $(D\tau)^{1/2}$ respectively, be larger than or comparable with the usual holographic fringes. Here, μ is the electron mobility, τ is the life time before trapping, D is the electron diffusion coefficient and E_o the applied electric field.

Most researchers using this material for photorefractive experiments such as wavefront phase conjugation and real-time holography, have employed coherent write beams in the green and blue regions of the spectrum. Because the purity of the feed materials used to grow $Bi_{12}SiO_{20}$ is generally fairly low, it is not clear whether the free carriers generated by this light have their origin in impurity-ion ionization, lattice defects, electron detrapping or a combination of both!

Experimental Approach

To date, only optical characterization of the defects has been carried out. The experiments include, absorption spectroscopy, cathodoluminescence at 300°K and 77°K and photoluminescence at 300°K and 77°K. These experiments were carried out on as-grown samples, as well as on samples annealed in oxygen at 700°C. Figure 1 is an illustration of the apparatus used for cathodoluminescence and photoluminescence experiments. The photoluminescence pump laser wavelength was 4579°A.

Figure 2 shows the absorption spectrum of a sample of $Bi_{12}SiO_{20}$ at room temperature. Notice that the absorption band edge is extended in the region 400 - 650 nm approximately. It is the origin of this weak but measurable absorption and its role in the photorefractive effect that is of interest here.

Figure 3 shows a low energy (2 keV) cathodoluminescence spectrum of $Bi_{12}SiO_{20}$ at 77°K. The strong emmission peak at 640 nm and the sholder at 580 nm are believed to be due to either lattice defect emmission or impurity-ion electronic transitions.

Figure 4 shows the temperature dependence of the cathodoluminescence of $Bi_{12}SiO_{20}$. Under steady electron beam excitation, the intensity of the cathodoluminesce generally increases as the temperature is decreased reaching a maximum at 173°K. The initial increase in luminescence may be due to the fact that as the temperature is lowered, fewer photons are available for the transfer of energy to nonradiative processes. The cathodoluminescence at room temperature was barely measureable with the equipment available.

Figures 5(a) and 5(b) show the room temperature photoluminescence of As-grown and O_2-annealed crystals. The spectra are seen to be identical. Annealing the crystal in an O_2 atmosphere for 5 hours would have been expected to remove all oxygen vacancies, and so we conclude that oxygen vacancies are not contributing the photoluminescence. The room temperature photoluminescence spectra exhibit two peaks centered at 560 and 635 nm respectively. Figure 6 shows that when the samples are cooled to liquid nitrogen temperature, the luminescence at 560 nm is partially quenched.

Further analysis of this data is being continued and complementary optical excitation spectra and high temperature transport measurements in the dark are being planned.

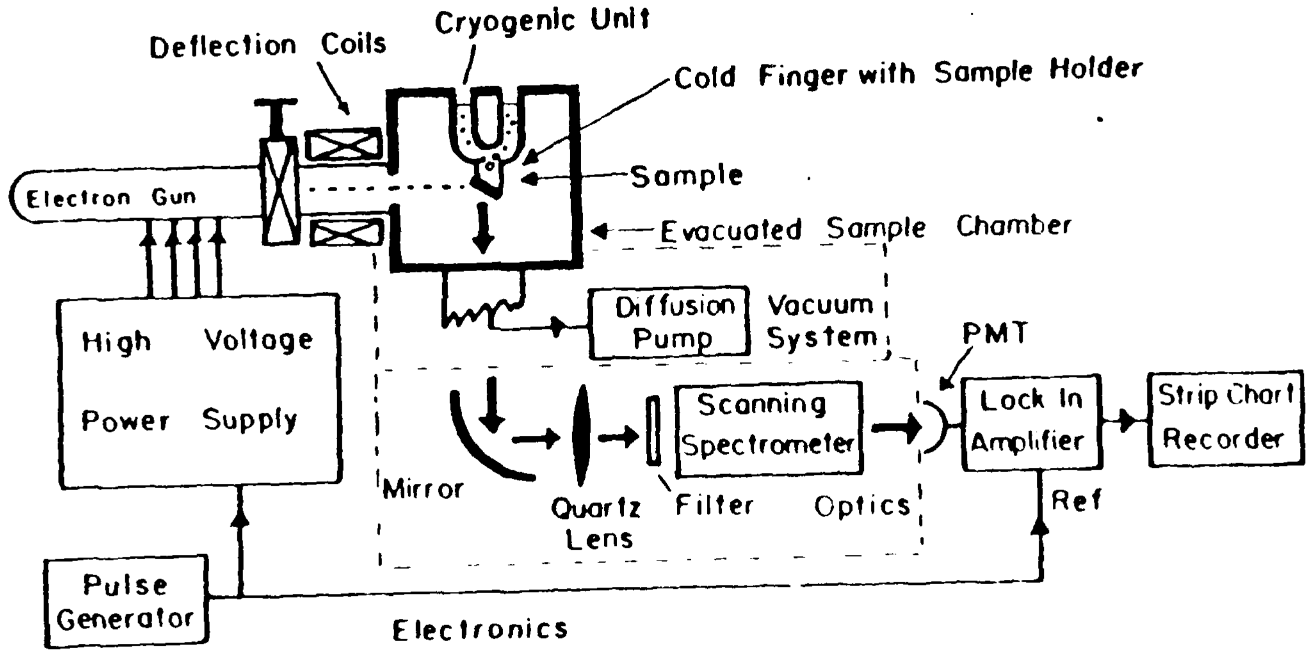


FIG 1 Block Diagram for Cathodoluminescence Experiments

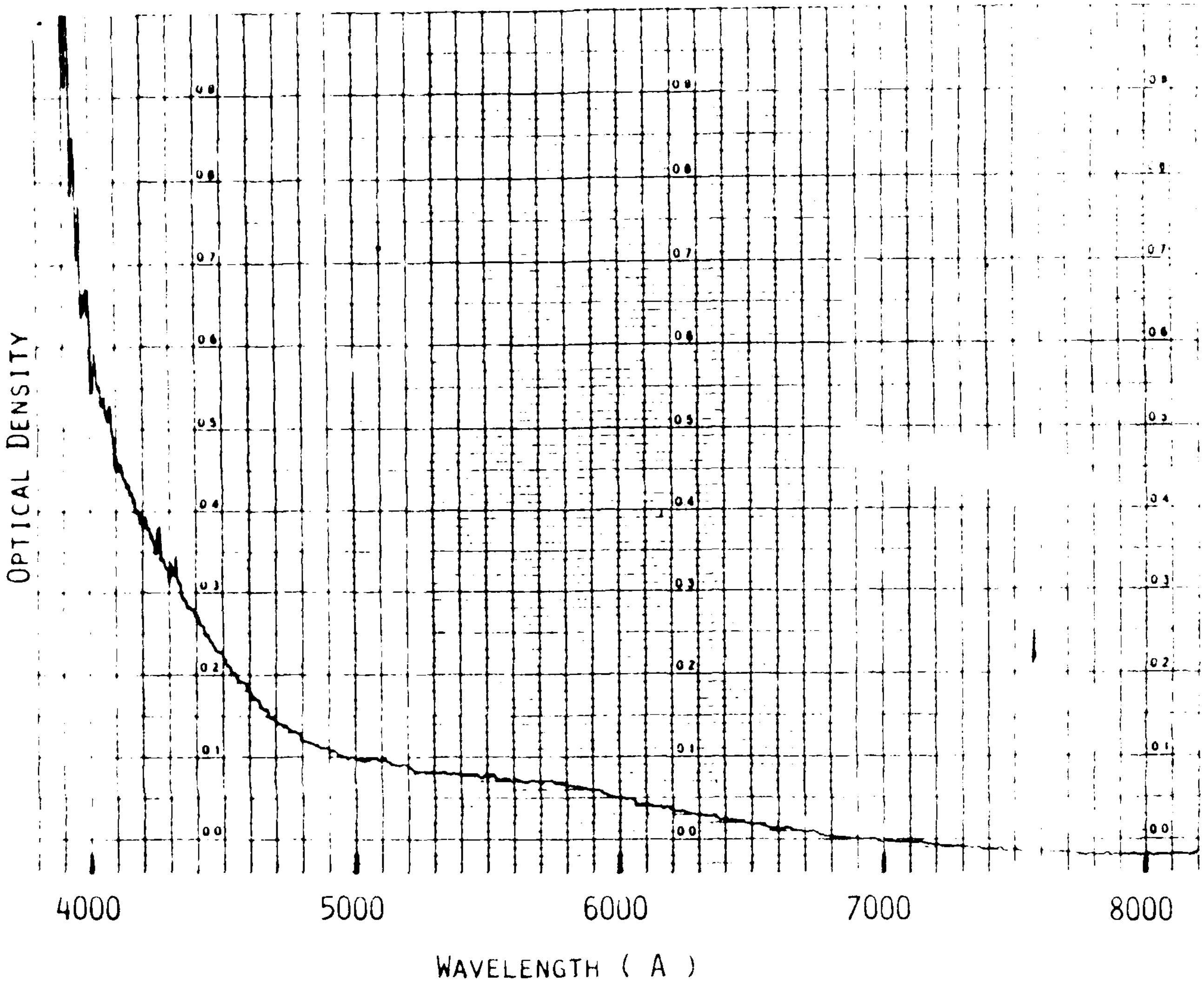


Fig. 2 ABSORPTION SPECTRUM OF $BI_{12}SIO_{20}$ AT 300 ^{0}K

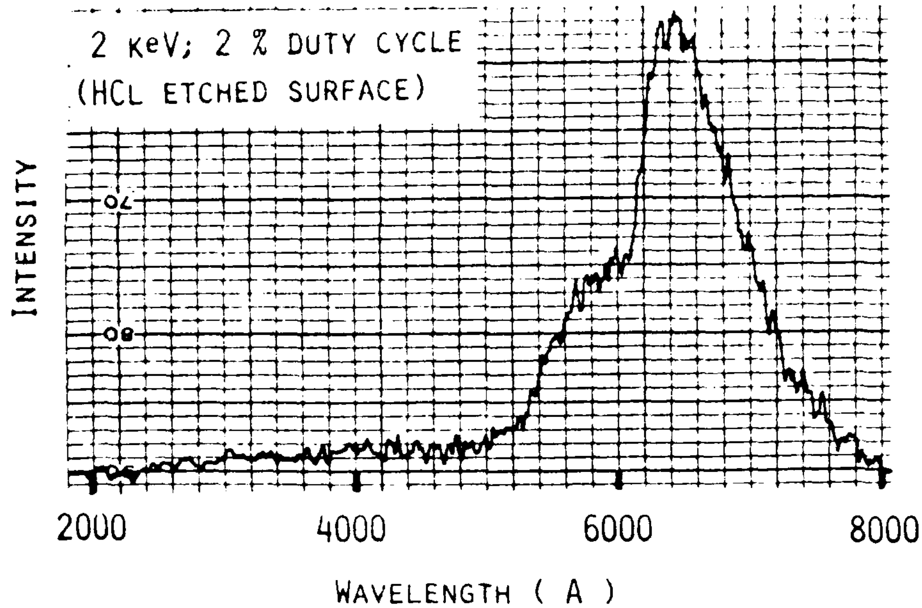
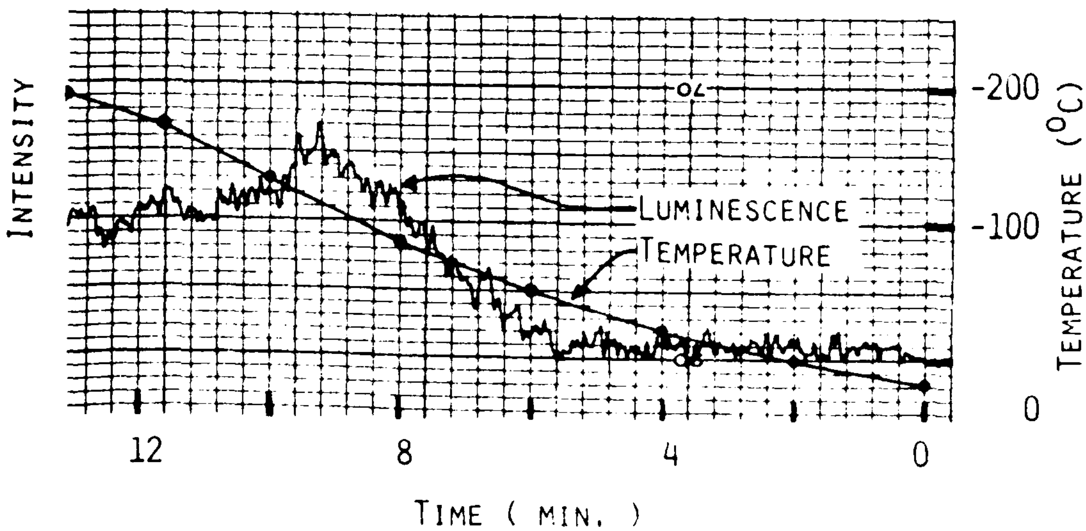
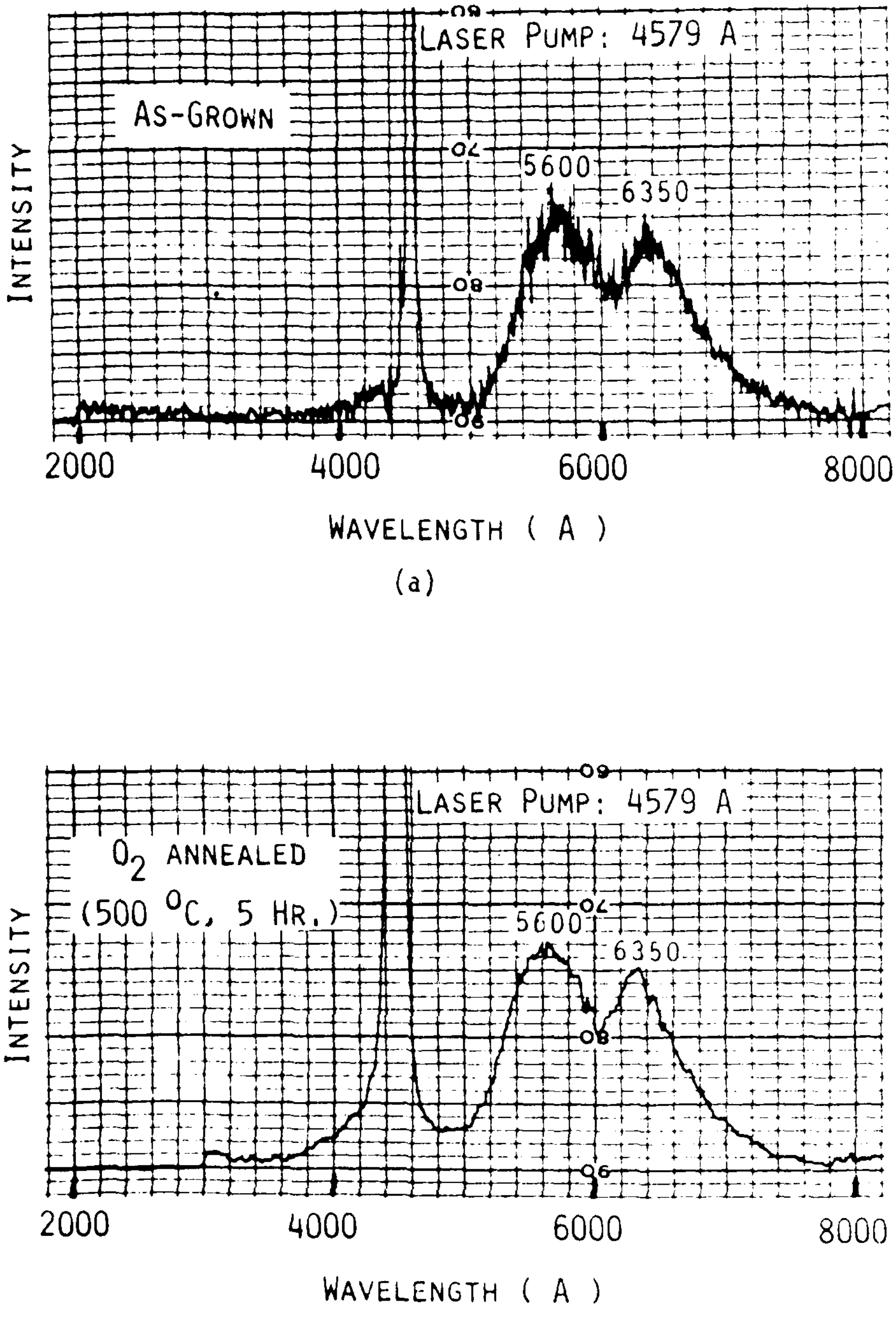


Fig. 3 CATHODOLUMINESCENCE OF $BI_{12}SIO_{20}$ AT 77 ^{o}K



$BI_{12}SIO_{20}$ CATHODOLUMINESCENCE AT 6500 A VS TIME AND TEMPERATURE

Fig. 4



(a)

(b)

Figs. 5(a),(b) PHOTOLUMINESCENCE OF $Bi_{12}SiO_{20}$ AT 300 ^{o}K

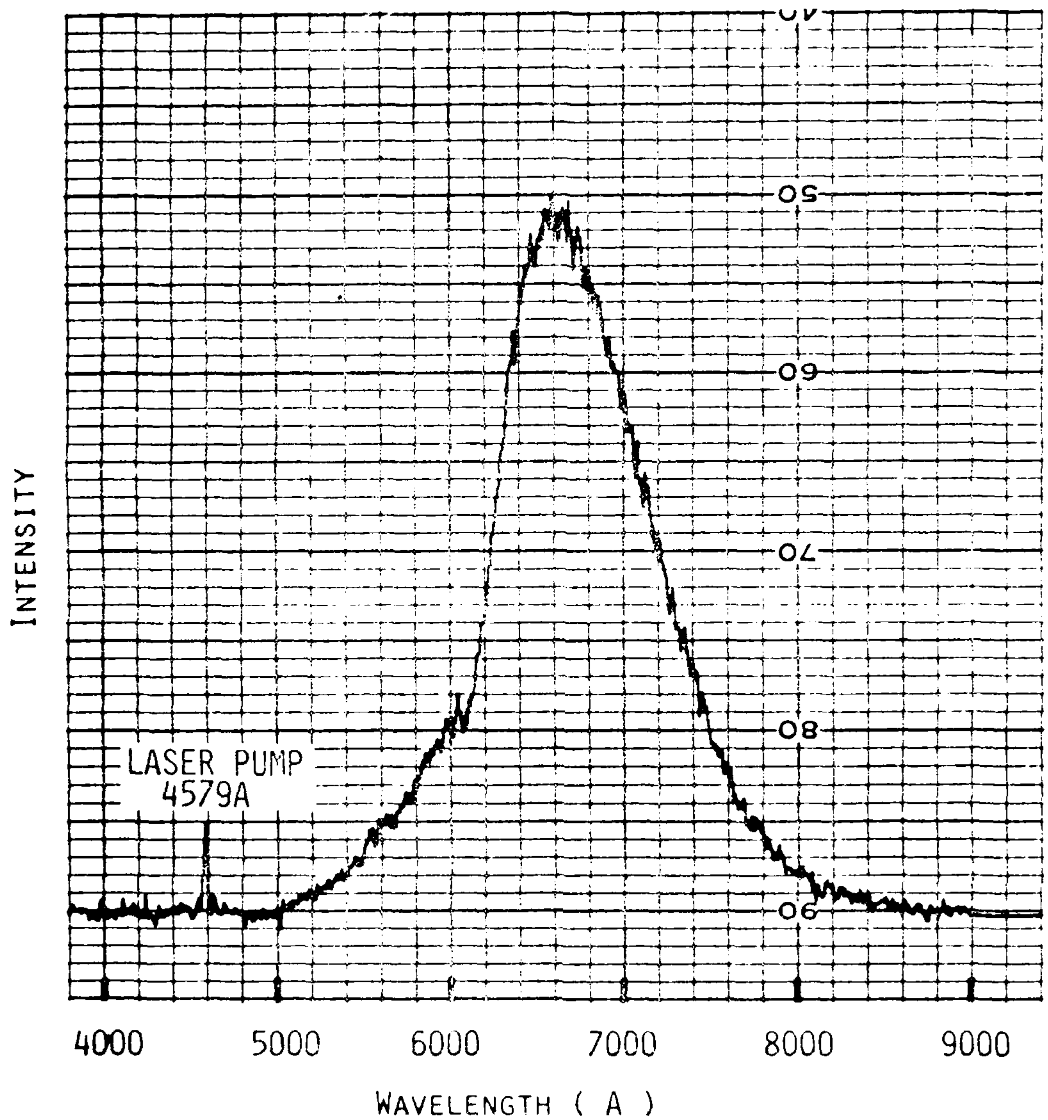


Fig. 6 PHOTOLUMINESCENCE OF $BI_{12}SIO_{20}$ AT 77 ^{O}K

(FILTER CUTOFF 5200 A)

Chapter 3

LIGHT-INDUCED UNIDIRECTIONAL LIGHT SWITCHING

When atoms are being exposed to an intense monochromatic light source, they are subject to light shifts[1,2]. Under certain conditions, these light shifts can be made to depend on the atoms' velocity, and this dependence can be used to compensate for Doppler broadening.[3,4] The anisotropy that characterizes this new Doppler-compensation technique has been used to induce an anisotropy of the superfluorescent emission of an atomic transition.[5] In this Letter we report the first experimental observation of a light-induced light switching in a unidirectionally amplifying medium.

Figure 1 shows the principle of the experiment. The final state of a $|c\rangle\rightarrow|a\rangle$ atomic transition is coupled to a doublet $|b_+\rangle, |b_-\rangle$ by a laser tuned between the $|a\rangle\rightarrow|b_+\rangle$ and $|a\rangle\rightarrow|b_-\rangle$ resonances. There is a frequency ω_{L0} around which the perturbed energy of $|a\rangle$ depends linearly on $\omega_L - \omega_{L0}$ (to third-order terms), where ω_L is the frequency seen by an atom in its rest frame. With the laser tuned to ω_{L0}, a moving atom will see a laser frequency detuned from ω_{L0} by an amount proportional to its velocity (Doppler shift). The frequency shift of the $|c\rangle\rightarrow|a\rangle$ transition will therefore be proportional to this velocity. The proportionality coefficient can be set by choosing the laser intensity so that light shifts and Doppler shifts of the $|c\rangle\rightarrow|a\rangle$ transition cancel for the emission in the direction of propagation of the laser and add in the background direction to double the Doppler broadening.

Consider an inverted medium with respect to the $|c\rangle\rightarrow|a\rangle$ transition. This medium will emit photons, and stimulated emission dominates if the gain is large enough. This type of emission, hereafter referred to as superfluorescence, is subject to a thresholdlike condition, that is, it can occur only if $g(\omega)\ell\Delta\omega > Q$, where $g(\omega)$ is the gain per unit length per unit bandwidth at a frequency ω, ℓ is the length of the medium, and $\Delta\omega$ is the bandwidth of the radiation. The quantity Q on the right-hand side of the inequality is a dimensionless number generally considered to be of the order of 20. Doppler broadening is an important limitation on the maximum amplitude of $g(\omega)$. Compensation of the Doppler effect brings the gain bandwidth from the Doppler width (~1 GHz) down to the natural width of the emitting transition (~50 MHz), thus leading to an important increase in the gain amplitude for superfluorescence. Since $g(\omega)$ is also proportional to the population inversion on the $|c\rangle \to |a\rangle$ transition, Doppler compensation will lead to a dramatic decrease in the threshold population inversion beyond which superfluorescence is allowed to develop. By using the Doppler-compensation method described above, it is possible to switch the emitting medium into a fast super-fluorescent decay state through introduction of a compensating laser pulse. This fast decay liberates a pulse of light that otherwise would not appear, so the compensating laser is turned off, the emitting medium will resume the normal fluorescent state; in other words, superfluorescence is switched off. Since one of the main features of the Doppler-compensation method by velocity dependent light shifts is its high degree of anisotropy and nonlinearity, it provides a method of light-induced light switching in a unidirectionally amplifying medium. Owing to the high nonlinearity of the system, one should expect high anisotropy of switching to be obtained even by only partial compensation for the Doppler effect.

In our experiment, $|c\rangle$ is the 3D doublet of sodium, and population inversion is achieved on the $3D \to 3P_{3/2}$ ($|c\rangle \to |a\rangle$) transition by two-photon pumping from the $3S_{1/2}$ ground state at a wavelength of 685 nm. Doppler compensation on the $3D \to 3P_{3/2}$ transition can then be achieved by coupling the $3P_{3/2}$ state to the 4D finestructure doublet ($|b_+\rangle, |b_-\rangle$) by use of a laser of 569-nm wavelength.

Figure 2 shows the experimental setup. It consists of an amplified Q-switched Nd:YAG laser whose second harmonic pumps two dye-laser oscillator-amplifier systems. The first dye laser is tuned to the 3S-3D two-photon transition of sodium at 685 nm. This laser, hereafter referred to as the pump laser, produces 3-nsec pulses at a 10-Hz repetition rate with an energy of 1 mJ/pulse after two stages of amplification. The pumping laser is tightly focused in a sodium cell, which operates at a pressure of a few miliTorr of sodium. To ensure a proper pumping of the entire Doppler width, a counterpropagating two-photon pumping scheme is utilized.[6] A second dye laser pumped by the same Nd:YAG laser is tuned to the $3P_{3/2}$-4D transition at 569 nm. This compensating laser operates in a single longitudinal cavity mode and is built following Littman's design.[7] Tuning is accomplished by rotation of a grating (coarse tuning) and translation of an end reflector mounted on a piezoelectric translator (fine tuning). Both tuning mechanisms are controlled by a MINC-11 computer. The pulse delivered by the compensating laser has a duration of 5 nsec. It is amplified in two stages and spatially filtered to obtain a smooth beam profile with power densities in the interaction region as high as 20 MW/cm^2. The pumping and compensating lasers are spatially and temporarily overlapped in the sodium cell.

Best compensation is obtained when the pump and compensating lasers are circularly polarized in opposite directions. This is achieved by having the two dye lasers linearly polarized in the perpendicular direction and passing them through a Fresnel rhomb, which acts like an achromatic $\lambda/4$ plate. Part of the compensating beam is mixed in another sodium cell with the light of a third laser operated at the sodium D_2-line wavelength. The 330-nm fluorescence emitted during the 4d-4P-3S cascade is monitored, permitting tuning of the compensating laser to $3P_{3/2}$-4D resonance. The intensity fluctuations within a single pulse are high (up to 50%) for the pump laser and rather low (less than 10%) for the compensating laser. Shot-to-shot pulse energy fluctuations are of the order of 20% for both lasers. Shot-to-shot frequency fluctuations for the compensating laser are of the order of 100 MHz, which is small compared with the 4D fine-structure splitting (1.05 GHz). The superfluorescent signals (forward and backward) are filtered by monochromators and detected with photomultipliers. The signals are then amplified, fed to analog-to-digital converters, and processed by a computer. Data are presented in the form of histograms plotting the number of shots giving rise to a given superfluorescence intensity versus the compensating laser intensity. The pump-laser intensity was set to a high value in order to observe superfluorescence in both directions without Doppler compensation. Then the compensating laser was turned on, and its intensity was adjusted so when it was tuned through the $3P_{3/2} \rightarrow 4D$ resonance we were able to observe an increase in the signal propagating in the forward direction and a decrease in the signal propagating in the backward direction. Then the compensating-laser intensity was adjusted so that the best forward-backward asymmetry was obtained. The pump-laser intensity was then decreased so that in the absence of Doppler compensation no signal was detected in either direction. Finally, once the compensation conditions were applied, a signal in the forward direction was switched on.

Figure 3 shows the experimental results. The recorded histogram with a low-intensity pump-laser source and the compensating laser turned off is shown in Fig. 3(a). It represents a statistical average over 300 laser shots. The distribution peaks at a zero signal, which means that the pumping rate of the 3D state is too low to allow superfluorescence to develop; however, owing to pump-laser fluctuations, a few laser shots gave rise to a detectable signal. Figure 3(b) shows the results with the compensating laser turned on. The distribution no longer peaks at zero and extends much farther away toward higher intensities. This proves that the presence of the compensating laser has allowed the superfluorescence to develop. If we consider the mean value of the signal computed from the two histograms of Fig.3, it indicates an increase of a factor of 4. It has been shown,[5] in the case of an intense superfluorescent signal, that the largest increase obtainable is a factor of 2, corresponding to the situation in which all the photons that were emitted in the backward direction have been forced to emit in the forward direction. Thus a further increase in our signal mean value demonstrates the on-off switching characteristic of our technique. In cases (a) and (b) of Fig.3, no signal could be detected in the backward direction, and it was verified that whenever a signal could be detected in the backward direction the application of the compensation condition always leads to a decrease of this signal.

In conclusion, we have demonstrated a new method of light-by-light switching in a unidirectionally amplifying medium based on the method of

Doppler compensation by velocity-dependent light shifts. The time response of such a switch is limited by the fact that the bandwidth of the compensating pulse has to be small compared with the Doppler width of the atomic vaper. These widths are generally of the order of gigahertz, which means that the response time cannot be shorter than a few nanosceconds (i.e., $\sim 10^{-9}$ sec.). The rise and fall times of the switching are determined by the rise and fall times of the superfluorescent pulse, which, in this experiment, was of the order of 1 nsec (i.e., $\sim 10^{-9}$ sec). Clearly, an exciting feature of this switching technique is its unidirectionality, which makes it a unique tool for such applications as optical communications, ring lasers, Doppler-free directed superadiance, and Doppler-free coherent transients.

We wish to thank C. Cohen-Tannoudji for a stimulating conversation with critical review of the manuscript and B.C.Johnson, F.Leland, W.Lange and R.Bondurant for technical assistance and help with data taking. M.M. Salour is an Alfred P. Sloan Fellow.

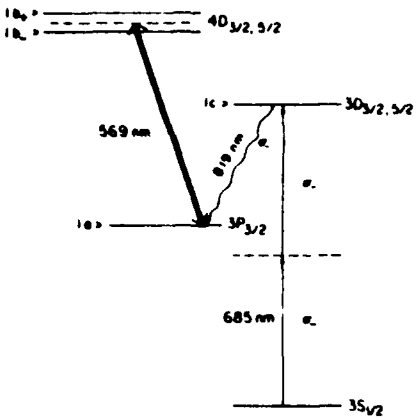


Fig. 1. Principle of Doppler compensation by velocity-dependent light shifts. Double arrow indicates the compensating laser light. Wavy arrow indicates the superfluorescent emission.

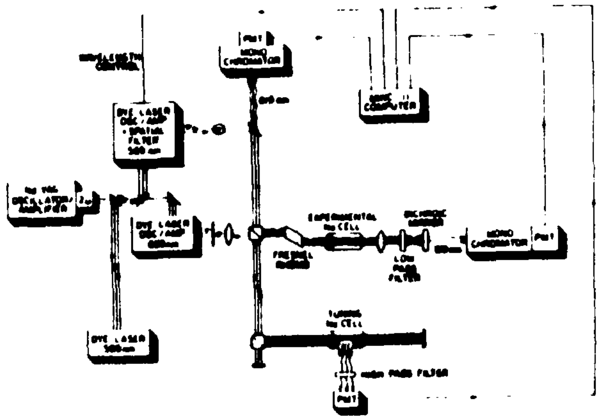


Fig. 2. Experimental setup.

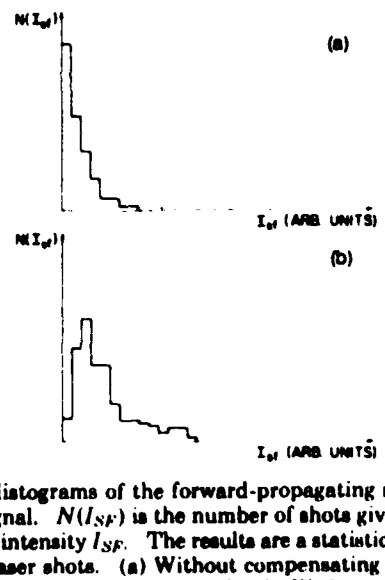


Fig. 3. Histograms of the forward-propagating superfluorescent signal. $N(I_{SF})$ is the number of shots giving rise to a signal of intensity I_{SF}. The results are a statistical average over 300 laser shots. (a) Without compensating pulse, the distribution peaks at a zero signal. (b) With compensating pulse, the distribution peaks at a signal different from zero.

References

1. J.P.Barrat and C.Cohen-Tannoudji, J.Phys.Radium 22, 39 (1961)

2. W.Happer, Progress in Quantum Electronics (Pergamon, New York, 1971) Vol.1; P.E.Toschek, in Frontiers in Laser Spectroscopy, R.Balian, S.Haroche, and S.Liberman, eds. (North-Holland, Amsterdam, 1977).

3. C.Cohen-Tannoudji, F.Hoffbeck, and S.Reynaud, Opt. Cummin. 27, 71 (1978).

4. S.Reynaud, M.Himbert, J.Dupont-Roc, H.H.Stroke, and C.Cohen-Tannoudji, Phys. Rev. Lett. 42, 756 (1979); S.Reynaud, M.Himbert, J.Dupont-Roc, and C.Cohen-Tannoudji, in Laser Spectroscopy IV, H.Walther and K.W.Rothe, eds. (Springer-Verlag, Berlin, 1979), p. 96.

5. G.Petite, B.C.Johnson, W.K.H.Lange, and M.M.Salour, Phys. Rev. Lett. 45, 1242 (1980)

6. R.Biraben, B.Cagnac, and G.Gryngerg, Phys. Rev. Lett. 32, 643 (1976); M.M.Salour, Opt. Commun. 18, 377 (1976).

7. M.G.Littman, Opt. Lett. 3, 138 (1978).

Chapter 4
DIGITAL OPTICAL POLYNOMIAL EVALUATION

Abstract

While analog optical pipelines for polynomial evaluators offer promise for root searching, function evaluation, and division, they do not have the accuracy needed for some problems. By ordering the bits properly we can achieve the needed accuracy through digital operation using the same hardware.

I. BACKGROUND

In prior papers (1,2) we showed how to make optical pipeline processors capable of evaluating the function

$$p(x) = p_n x^n + p_{n-1} x^{n-1} + \dots + p_0. \qquad (1)$$

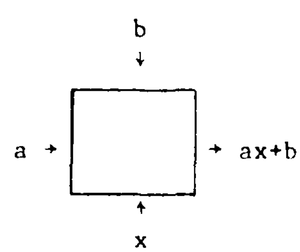
The basic scheme was Horner's rule

$$p(x) = (\dots((p_n x + p_{n-1})x + p_{n-2})x + p_0. \qquad (2)$$

While many schemes for the basic module are possible (1), they all perform an operation of the form

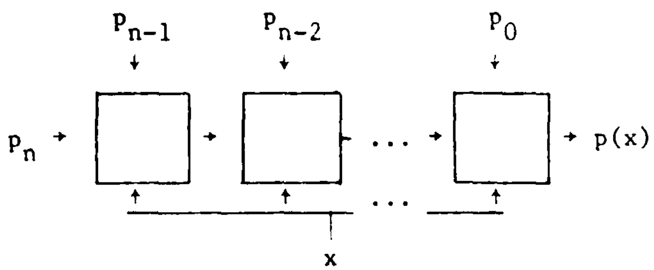
$$O(a,x,b) = ax+b.$$

Symbolically, we have



. (4)

A general polynomial evaluator for a nonnegative single variable with nonnegative coefficients is



(5)

Methods to handle real and complex numbers were shown along with ways to handle other considerations such as multiple variables (1).

II. GOAL OF THIS PAPER

The goal of this paper is to show that the same hardware used for analog pipeline processing can be used for digital processing. We will confine ourselves to achieving N bit accuracy on variables and coefficients. It requires little to go from there to floating point operation. Our emphasis will be on convenient data flow and on optimum use of hardware.

III. SINGLE MODULE OPERATIONS

To allow pipeline no detection should occur in any but the last module. For N bit accuracy, three data flow schemes suggest themselves:

(1) 2N+1 parallel channels (the most parallel and hence the fastest approach),

(2) N parallel channels with time division, and

(3) A single parallel channel with time division.

We will discuss all three.

Let us write

$$a = a_o 2^o + a_1 2^1 + \dots + a_N 2^N , \qquad (6)$$

$$x = x_o 2^o + x_1 2^1 + \dots + x_N 2^N , \qquad (7)$$

and

$$b = b_o 2^o + b_1 2^1 + \dots + b_N 2^N . \qquad (8)$$

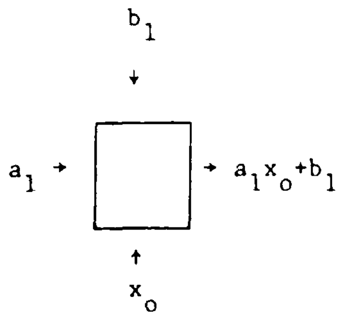
Note that the product ax contains a term $a_N x_N 2^{2N}$, so 2N+1 output coefficients for ax + b must be allowed for. Note, however, that ax+b becomes the input for the next stage, so "a" must, in general, have 2N+1 coefficients. Of course x and b will have only N coefficients.

Accordingly we must find a means capable of handling inputs and outputs up to 2^{2N+1}. We will arbitrarily terminate calculated outputs at that level and assume that the inputs are appropriately limited to prevent overflow.

FULLY PARALLEL CASE

We will use 2N+1 parallel processors. To show the data flow concretely we will limit our discussion to N=2. Generalization will be easy. We will draw charts of channel activity versus time. Channel n (corresponding to power of 2^n) will be the n+1 row since Row 1 will be used for n=0. Time flow is left to right.

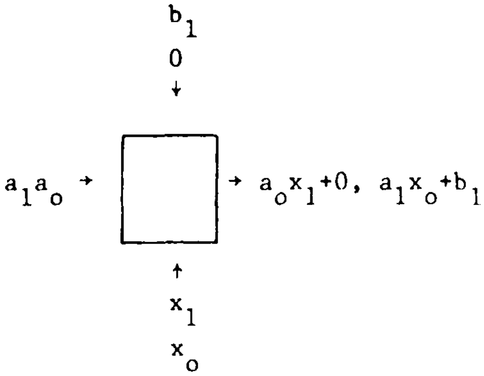
The operation



will simply be written

$$a_1x_o + b_1.$$

For instance, in the n=1 channel we could write



meaning that at time instant 1 we multiply a_o by x_1 and add 0 and time instant 2 we multiply a_1 by x_o and add b_1. We will write this

$$a_ox_1 + 0 \quad a_1s_o + b_1.$$

The data flow pattern is obvious in the two-dimensional plot (Figure 1). Only at the detector do terms like these get summed.

PARTIALLY PARALLEL CASE

Clearly if we allow a given ax+b multiplier to calculate terms for multiple powers of 2 we can get by with only N+1 multipliers. Figure 2 shows the data flow.

SINGLE CHANNEL CASE

The single channel case is the most difficult and inconvenient. For pipelining we want to calculate the coefficients d_o, d_1, ..., d_4 in order. The data flow of Figure 3 will be adequate. We go back to an earlier notation for convenience. From the point of view of memory, this is quite unsatisfactory. In the two prior cases each x_k was removed from memory only once and then moved through the space-time pattern systematically (down for 2N+1 channels or down with "wraparound" for N+1 channels). The a_k's were removed from memory only once and then applied only once in parallel. For the single channel all of these conveniences vanish and the data flow looks and is complicated.

CONCLUSIONS

Pipelined N-bit accuracy optical polynomial evaluation is quite practical at a factor of 2N+1 in speed and a factor of 2N or 2N+1 increase in the number of processors. The data flow is simple, systolic, and non Von Neumann. Going to only one processor, however, puts us back into a complicated Von Neumann situation as well as slowing us down by a factor of about $4N^2$ for large N (12 for the small N=2 case illustrated above).

REFERENCES

1. C.M. Verber, R.P. Kenan, H.J. Caulfield, J.E. Ludman and P.D. Stilwell, Jr., "Suggested Integrated Optical Implementation of Pipelined Polynomial Processors", Proc. SPIE 388, (1983).

2. H.J. Caulfield, "Application of Optical Pipelines to Root Searching and Division", Appl. Optics. (To be published).

POWER OF 2					
0	$a_0x_0 + b_0$	00 + 0	00 + 0	00 + 0	00 + 0
1	$a_0x_1 + 0$	$a_1x_0 + b_1$	00 + 0	00 + 0	00 + 0
2	$a_0x_2 + 0$	$a_1x_1 + 0$	$a_2x_0 + b_2$	00 + 0	00 + 0
3	00 + 0	$a_1x_2 + 0$	$a_2x_1 + 0$	$a_3x_0 + b_3$	00 + 0
4	00 + 0	00 + 0	$a_2x_2 + 0$	$a_3x_1 + 0$	$a_4x_0 + b_4$
5	1	2	3	4	5

CLOCK PERIOD NUMBER

Figure 1. The data flow for ax = b for 2 bit accuracy. Note all a's are addressed in parallel to the appropriate ax multiplier. The x's move "downward" one multiplier each clock period. The b's are added when all of the ax terms for the corresponding power of 2 have been calculated. The sum over five time periods gives the proper d_k values for $ax + b = d = d_4 2^4 + d_3 2^3 + \ldots + d_0$.

CHANNEL $(2^0, 2^3)$	$a_0x_0 + b_0$	$a_1x_2 + 0$	$a_2x_1 + 0$	$a_3x_0 + b_3$	$00 + 0$
CHANNEL $(2^1, 2^4)$	$a_0x_1 + 0$	$a_1x_0 + b_1$	$a_2x_2 + 0$	$a_3x_1 + 0$	$a_4x_0 + b_4$
CHANNEL (2^2)	$a_0x_2 + 0$	$a_1x_1 + 0$	$a_2x_0 + b_2$	$00 + 0$	$00 + 0$
	1	2	3	4	5

CLOCK PERIOD NUMBER

Figure 2. With N+1 multipliers we can time share the channel. Thus the first channel can calculate the 2^0 coefficient and then the 2^3 coefficient. The data flow pattern is again obvious and again one power of 2 is completed each clock time.

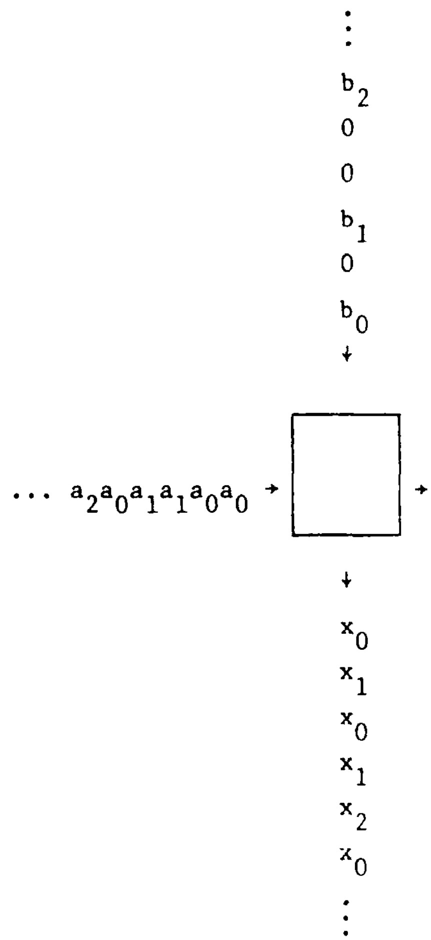


Figure 3. A possible data flow pattern for a single processor.

Chapter 5

APPLICATION OF OPTICAL PIPELINES TO ROOT SEARCHING AND TO DIVISION

Abstract

Optical pipeline polynomial evaluators have been described earlier. Here we examine in detail two applications: (1) the search for roots of one or possibly many simultaneous polynominals and (2) optical division.

Introduction

In a previous paper we introduced the concept of a pipelined optical polynomial computer implementing Horner's rule.(1) Here we wish to examine two applications: the search for roots of polynomials or, equivalently, multiple polynomials and optical division.

POLYNOMIAL ROOTS

The basic concept is to simply search through all values of the independent variable. This can be done very quickly.(1) The question we address here is: "Can it be done accurately?"

Accuracy requires careful attention to questions of the scale of optical coefficients and variables. The coefficients are optical source signal levels and have some maximum values S_{max}. The variables are transmissions with a maximum value of unity. With those basic restrictions, we want to do high accuracy evaluations over a wide range of values of the variable. As shown in (1), no generality is lost by restricting ourselves to a single positive variable x. The polynomial is

$$P(x) = \sum_{K=0}^{n} a_K \, x^K \qquad (1)$$

We will treat the scaling of a_K's there and x's in sequence.

In scaling the a_K's there is one obvious rule: keep the largest $|a_K|$ in any pipeline at S_{max}. Any lower value throws away dynamic range. If the available dynamic range is inadequate to cover $|(a_K)_{max}|/|(a_K)_{min}|$, the obvious approach is to break the problem into two or more problems which can be handled, e.g.

$$\begin{aligned} P(x) &= 2X^4 - 127X^3 + 84X^2 + 6X - 5 \\ &= \underbrace{(2X^4 + 6X - 5)}_{P_1(x)} + \underbrace{(-127X^3 + 84X^2)}_{P_2(x)} \\ &= P_1(x) + P_2(x) \end{aligned} \qquad (2)$$

$P_1(x)$ and $P_2(x)$ are separately calculated and detected. The results are then added electronically.

The problem of x values is more complex. If we know $|x|_{max} = b$, we can replace x with

$$y = x/b \quad . \qquad (3)$$

This means

$$\begin{aligned} P(x) &= \sum_{k=0}^{n} a_k b^k y^k \\ &= \sum_{k=0}^{n} W_k y^k \quad . \end{aligned} \qquad (4)$$

In some cases, x may not have a maximum. In those cases we can use Eq. (4) for $0 \leq x \leq b$ and use a different approach for $x > b$. Suppose

$$x = 1/y = b/x \quad . \tag{5}$$

Then

$$\begin{aligned} P(x) &= \sum_{k=0}^{n} a_k b^k z^{-k} \\ &= z^{-n} \left[\sum_{k=0}^{n} a_k b^k z^{n-k} \right] \\ &= z^{-n} \sum_{k=0}^{n} w_k z^{n-k} \end{aligned} \tag{6}$$

The term in brackets has only non-negative exponents of z. Thus in searching for roots we can scan

$$P_1(x) = \sum_{k=0}^{n} W_k X^k \tag{7}$$

over $0 \leq x \leq 1$ and scan

$$P_2(x) = \sum_{k=0}^{n} W_k z^{n-k} \tag{8}$$

over $0 \leq z \leq 1$. This covers all $x \geq 0$.

DIVISION

We now apply the mathematician's favorite trick of converting a problem we cannot solve into a form recognizable as a problem we have already solved. The problem optics has never solved is division. That is, given a find 1/a. Consider the function

$$f(x) = (1/x) - a \quad . \tag{9}$$

We note that the root ξ of f(x), i.e. the value of x such that $f(\xi) = 0$, has the value

$$\xi = 1/a \quad . \tag{10}$$

We also note that we can expand f(x) as a polynomial about some reference value γ_0. Dividing x into the two previously noted ranges ($0 < x \leq 1$ and $x > 1$), we see that $\gamma_0 = 0.5$ might be a reasonable choice. Of course we can write

$$F(x) = g(x) - a \tag{11}$$

and

$$g(x) = g(\gamma_0) + \frac{g'(\gamma 0)}{1!}(x - \gamma_0) + \frac{g''(\gamma 0)}{2!}(x - \gamma_0)^2 + \ldots \quad . \tag{12}$$

Since
$$\begin{aligned} g(x) &= x^{-1}, \\ g'(x) &= -x^{-2}, \\ g''(x) &= 2x^{-3}, \\ &\vdots \end{aligned}$$

we find, that for $\gamma_0 = 0.5$,

$$\frac{g^{(n)}(\gamma_0)}{n!} = (-1)^n \, 2^{n+1} \quad . \tag{13}$$

Therefore

$$f(x) = 2 - a - 4(x - 0.5) + 8(x - 0.5)^2 - 16(x - 0.5)^3$$
$$+ 32(x - 0.5)^4 - \ldots \quad . \tag{14}$$

One then must use enough terms to achieve an adequate accuracy answer.

Alternately we can use this to provide a starting estimate x_0 and then use Newton's method. That is

$$x_1 = x_0 - (1/x_0 - a)(-1/x_0^2)$$
$$= (2 - ax_0)x_0 = 2x_0 - ax_0^2 \quad ,$$
$$x_2 = 2x_1 - ax_1^2, \quad \ldots \quad . \tag{15}$$

We can even go to more complicated but better-converging iterative algorithms.[2] All of these operations fit into the category: things we already know how to do.

CONCLUSION

Polynomial root searching can be done accurately by optical methods. One application is the simultaneous solution of multiple polynomials. Thus to find x's satisfying

$$P_1(x) = P_2(x) \quad , \tag{16}$$

we seek the roots of

$$D(x) = P_1(x) - P_2(x) \quad . \tag{17}$$

To find x's satisfying

$$P_1(x) = P_2(x) = \ldots = P_N(x) = 0$$

we seek the roots of

$$C(x) = \left[P_1(x)\right]^2 + \left[P_2(x)\right]^2 + \ldots + \left[P_N(x)\right]^2 \quad . \tag{18}$$

A second application is a new one to optics: division.

REFERENCES

1. C.M. Verber, R.P. Kenan, H.J. Caulfield, J.E. Ludman, and P.D. Stilwell, Jr., Appl. Opt. (to be published).

2. E.V. Krishnamurthy, IEEE Trans. Comput., C20, 470 (1971)

Chapter 6

DATA-LOGGING USING THE HP-9825A CALCULATOR

Introduction

This chapter describes procedures and software for using the Hewlett-Packard 9825A Calculator for, what is usually called,"data-logging". More specifically, (step 1) various parameters are stored on the Calculator tape cassette; (step 2) analog data which has been digitized using an HP 3455A Digital Voltmeter (DVM) is recorded on the same cassette. In the last step (step 3) the Calculator becomes an "intelligent terminal" and transmits the data (and parameters) to a large time-sharing computer (e.g., the CDC6600 at AFGL) under computer control. The data thus gathered and transmitted to the Computer is stored as part of a large data base and is manipulated by a resident data reduction/analysis/display program.

It is of course possible, and perhaps desirable, to do the reduction/analysis/display using only the Calculator and perhaps an attached plotter. In this case the "logging" process stops when the data has been stored on the cassette or perhaps just in Calculator memory. On the other hand, it may sometimes be desirable to be able to use the Calculator only as an intelligent terminal. Using "subroutines" from this report it is possible to use the Calculator for either of these tasks which are part of the overall "data-logging" task.

Historically, the reason for using this technique for gathering data is that an extensive collection of spectral data, gathered using a Digilab FTS14 Fourier Spectrometer, is currently stored on magnetic tape at the AFGL Computer Center. Moreover, an extensive set of software modules exists at this center for the purposes of manipulating this data in various ways and eventually for plotting the results of manipulation. (See, for example, "Topics in Optical Materials and Device Research", PML, Inc., Final Technical Report RADC-TR-78-61, (1978).) While the FTS14 provides digital data which can be used directly by the Computer, there exist other, older spectrometers which provide only analog data.

Using techniques described in this chapter, it is possible to treat data gathered from these older spectrometers in exactly the same manner as the FTS14 data. Parenthetically, it should also be noted that similar work has been done for a spectrometer which produces data on punched paper tape.

This chapter is partitioned into 6 sections. In Section I an overall description of the data-logger and its operational environment is presented. In Section II a description of the data gathering procedure is given. Section III gives a description of the procedure for transferring the "logged" data to the central Computer. Section IV lists and describes the data-logging program in the 9825A, while Section V lists and describes the corresponding "data receiving" program in the remote computer (CDC6600). Section VI is a description of the pre-processing (CDC6600) software which provides the interface for the existing FTS14 software enabling it to accept the data gathered in the previous steps.

I. Overall Data-Logging System

Figure 1. shows the various pieces of hardware involved in the data-logging process and their interconnection.

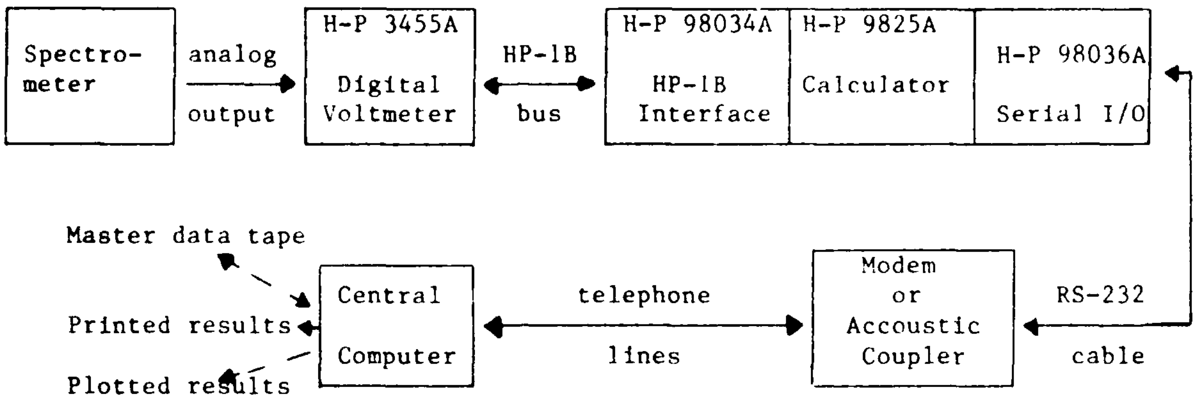


Figure 1 - Data-logging Hardware and Hardware Interconnection

The spectrometer in this diagram could be any instrument which outputs a slowly varying signal (has negligible spectral energy above 20Hz). However, in this chapter, the example chosen is a McPherson 213 spectrometer the output of which has been processed by a PAR Lock-In amplifier. This amplifier is designed to drive a galvenometric recorder with a 0-1 ma movement. Using a 1.5K ohm resistor across the output terminals, a voltage in the 0-1.5V range is obtained to be measured by the DVM. (The input impedance of the DVM is 2 Megohm which is essentially infinite for such a low source impedance.) A measurement (in this example) consists of sweeping the spectrometer across a pre-determined spectral region (e.g., 2300 Angstroms to 2800 Angstroms) -- first with a sample, then without a sample. The ratio of the two sets of measurements is the transmitivity of the sample and is the experimental result to be recorded. In addition, enough information for computation of the wave-number associated with each transmission value must be recorded. This is provided by recording the starting wavelength λ_s in Angstroms (the required wavenumber is $10^8/\lambda_s$), the sample rate (sec-1) and the scan speed (Angstroms/min). Header information, including the sample thickness (see Section IV), must also be provided by the experimenter. At the end of an experiment, or perhaps at the end of several experiments, the recorded data is transmitted to the Central Computer with the Calculator operating in the "terminal mode". This transfer of data is done under control of a program at the Central Computer after the log-in and program call sequence. After sending the data, the operator has several options, such as causing the data to be filed on a master tape and/or processed using the FTS14 system of software.

II. Data-Logging

The HP 9825A is operated in the data-logging mode using the HP 3455A digital voltmeter and HP-1B interface. The DVM is connected to the lock-in amplifier (which has a 1500 ohm precision resistor across its high/low terminals) with DVM ground connected to amplifier ground via a heavy ground wire. The following settings should be used on the DVM:

RANGE	1
FUNCTION	±V
AUTO CAL	on
HIGH RESOLUTION	off
TRIGGER	HOLD/MANUAL
SAMPLE RATE	MAXIMUM
MATH	OFF

These switches, with the exception of SAMPLE RATE, will also be set under program control. The above settings are for use during set-up of the experiment.

After the spectrometer has been properly set up (with a sample), the experimenter starts the Calculator which in turn automatically loads and starts running the data-logger program (assuming the correct tape cassette has been inserted). Track 0 of this tape holds the various subroutines (including the driver program) required while track 1 is used for storing the data. The first prompt asks L OR T to which the operator should reply L for data-logging. The next prompt is FILENAME?

to which the operator should reply with a 10 character filename. This filename uniquely identifies the experiment and the label of the data file created in the "TRANS" stage.

The experimenter next enters the header data after the A[11,91]? prompt appears in the LED display. The header, which may be up to 80 characters long, is completed when the CONTINUE key is pressed. Next the prompts, START WL, SAMPLE RATE, SCAN SPEED and END WL (see below) must be answered with appropriate values. The Calculator records this data on the printer tape and cassette, then waits in a loop for the operator to press RUN. This key should be pressed at exactly the same time as the spectrometer is started. If the end wavelength entered by the operator is greater than the start wavelength, the Calculator will automatically stop gathering data when the calculated end wavelength is reached. Otherwise, the operator must push the STOP key. (The data gathering process will also stop when the data storage array has become full.)

At the end of this phase of the experiment, the Calculator again goes into a wait loop for the RUN key to be pressed again. This should be done once the spectrometer has been set up for a transmission measurement without a sample -- no header data, etc. is inserted at this phase. Stop is automatic in this phase.

At the end of this phase, the calculator stores the computed relative transmission on the tape cassette. Program flow is such that the Calculator continues to the Data transmission phase. If data is not to be transmitted at this time, the program should be halted by pressing "reset".

III. Data Transmission

For this phase, the Calculator must be connected to an accoustic coupler (or modem) using the HP Serial I/O interface (HP-98036A) and the appropriate RS-232 cable. (The only lines required are Signal Ground (pin #7), Transmitted Data (pin #2) and Received Data (pin #3). Usually pins 2 and 3 are reversed in the cable to the modem, i.e., pin 2 at one end goes to pin 3 at the other end, etc. The Interface configuration switches should be set according to the factory settings. (Mode Word Switches are set within the program.) The Bit Rate switch should be set to 7 (300 baud) and the Interface Select Code set to 10.

The Calculator is then started and the L OR T prompt answered with T. Another prompt will then appear which asks AUTO?. If the operator reply is "yes", logging-in and remote program execution is done automatically. The communications with CDC6600 should then be established and the STORE key pressed twice. If the Calculator is in the AUTO log-in mode, no further action is required from the operator. Otherwise the operator must log-in using the standard Intercom procedures. In addition, execution of the data "gathering" procedure must be initiated by attaching the appropriate library, issuing the LIBRARY command followed by the name of the procedure to be executed. For example (procedure names and ID's may change, of course), the following is a typical sequence of commands (in response to COMMAND-)

```
ATTACH (HL,HLTLIBX3693818,ID=BARRETT)
LIBRARY (HL)
DATTRAN.
```

At the end of the "data-gathering" stage which may take several minutes, the Calculator will issue another prompt of the form: STORE OR RUN?

The operator may key the following responses:

S	=>	store the data on multifile tape
R	=>	run the data through the FTS14 system
SR	=>	S and R
X (any other key sequence)	=>	quit

If R or RS is the response, additional prompts having to do with the FTS14 system are issued. When the TRANS mode has completed all operations, the prompt "COMMAND-" will appear, indicating that the operator may now LOGOUT. (He may do other terminal type operations.) The RESET key may now be pressed to end operation of the TRANS mode (it does not terminate without operator intervention) or the operator may return to the start of the program by pressing control S.

IV. HP 9825A Data-Logging Software

The software is partitioned into 6 "logical" segments; each segment occupies a separate file on track 0 of the cassette. In addition, file 1 is used to hold the "wait table" which adjusts the DVM triggering interval according to the desired sampling rate. Each of the subroutine files is less than 1000 bytes in length; the table file is 5008 bytes long since the table uses the sampled data storage array which is 5000 bytes long. Two data files are used on track 1. The first data file holds header information, while the second file holds the sampled data (2 bytes/sample). A total of 2500 samples may be stored for any experiment.

The various segments are loaded automatically (chained) according to the flow of the overall program.

segment 1 (get experimental parameters) - file 0

This segment must always be loaded first. Among other things it declares the large string array which is used for several purposes by other segments -- including its major use as the storage place for the sampled data.

dim A$[5000]; dim D$[10]	
"" --> A$	(initialize the string array)
ent "l or t?", A$	(the log or transmit prompt)
if cap(A$[1,1])="T"; ldf 3	(the cap function converts lower case letters to upper case. If the T-mode is desired, file 3 (segment 3) is loaded and executed)
enp "filename", A$	(the file name entered by the user is stored in A$ and printed on the 9825 printer)
enp A$[11,90]	(this causes the prompt A$[11,90]? and allows header data to be directly entered into A$[11] to A$[90]. The Calculator gives a "beep" when all but 13 characters have been entered. The user can press CONTINUE at any time to terminate the header (unfilled locations are filled with blanks).
flt 3	(this sets the str conversion format to (Fortran) E10.3)

```
enp "start wl", A; str(a)->A$[91]  (the starting wavelength is
                                   converted to characters using E10.3
                                   and stored in A$[91]->A$[100])
enp "sample rate", B; str (B)->A$[101]

enp "scan speed", C; str (C)->A$[111]

enp "end wl", E; str (E)->A$[121]

trk 1; rcf 0, A$                   (record the header data -- note that
                                   the entire array must be stored
                                   although only part of it contains
                                   information)

trk 0; ldf 1                       (load segment 2)
```

<u>segment 2</u> (collect sampled data) - file 1

```
ldf 2, A$                          (load the wait table into A$)

itf (A$[2B-1])->U                  (the sample rate for the first data
                                   gathering pass, stored in B, is
                                   assumed to be an integer from 1 to
                                   20 (1 to 20 samples per second).
                                   The table consists of 2 byte inte-
                                   gers representing possible positive
                                   values form 0 to 2^15.  The itf func-
                                   tion converts this byte integer into
                                   the 8 byte floating point form used
                                   in the HP 9825A.)

itf (A$[2B+19])->V                 (a different table must be used for
                                   the second data gathering pass
                                   because of different computations
                                   which are done)
```

C/60B->D	(compute the wavelength interval between samples)
if A>E; 1e10->E	(if the end wavelength is less than the starting wavelength, an artificially large end wavelength is used)
0->K; 1->I	(K is set to 0 for key monitoring, I is used as the sample counter)
A -> G	(starting wave number)
dev "dvm", 722	(define the address of the HP-1B interface (7) and the primary address of the DVM (22))
wrt "dvm", F1R3T2T3A0H1D0"	(initialize the DVM -- see the DVM manual for the significance of the adjacent pairs of characters in the sequence)
dsp "press any key to start"	(prompt to operator)
on key "key"	(sets the key interrupt handling subroutine to "key" and puts the keyboard under program monitor)
if not K; gto +0	(wait here until a key is pressed -- K will then become non-zero)
cfg0; cfg1	(clear flag 0 and 1; flag 0 is set in the interrupt handling subroutine "key" while flag 1 is set in the "test end of sampling" subroutine "t")

"gd": trg "dvm"	(start of sample loop -- the DVM is triggered to start a voltage measurement)
red "dvm", B; 10000B->B	(the data out of the DVM is in format E13.6; the voltage must be between 0 and 1.5 volts. This number is converted to be between 15000 and 0 for use with the fti function - see below)
dsp B	(display the current sample value)
fti(B)->A$[I+2 I,I+1]	(store the number in "split integer" form. This requires 2 bytes per value. The number is rounded to the nearest integer. The first 2 bytes of A$ are used later to store the number of samples)
gsb "t"	(check for end of sampling conditions - see below)
if not flg1; gto "gd"	(flag 1 is set by "t" at end of sampling)
fti (int (I/2))->A$[1,2]	(store the number of samples)
beep	(warn the operator that he will have to take action to continue the experiment)
dsp "press any key to start"	
1->J; 0->K	(J is the counter for the second phase)

```
if not K; gto +0
"ge":trg "dvm"                 (start of second phase loop)

red "dvm", A; itf(A$[J+2->J,J+1])->B  (read a sample and convert
                               the corresponding sample from the
                               first phase to normal form)

B/A --> B                      (since A must be  = the "original"
                               sample value obtained in phase 1,
                               the final value of B is between 0
                               and 9999)

dsp B                          (display the %- transmission value
                               (x100))

wait V                         (wait for the required number of V
                               milli-seconds before collecting the
                               next sample value)

if J<I; gto "ge"               (collect the same number of samples
                               in phase 2 as in phase 1)

beep                           (warn the operator that the experi-
                               ment is ended)

trk 1; rcf 1, A$; trk 0        (record the data samples in file 1
                               of track 1 and then set the track to
                               0 for program file loading)

prt "no. of samples=", itf (A$[1,2])  (record the number of
                               samples on the print tape)

on key; ldf 3                  (return the keyboard to normal
                               operation and load segment 3)
```

```
"key": key --> K              (the key interrupt handler subrou-
                              tine puts the key value in K)

        sfg 0                 (and sets flag 0)

        kret                  (key interrupt return)

"t":  if I=5000; sfg 1        (the "t" subroutine checks for
                              various end of sampling conditions
                              and sets flag 1 if it finds one)

      if flg 0; sfg 1         (the user has stopped the experiment
                              by pressing a key)

      G+D --> G               (compute the wavelength correspon-
                              ding to the next sample)

      if G>E; sfg 1           (if the next wavelength is greater
                              than the end wavelength, the end of
                              sampling condition is set)
      wait U                  (wait the required number of milli-
                              seconds for the next sample)

      ret
```

<u>segment 3</u> (automatic login and data transfer initiation) - file 3

```
5 --> Q                       (Q holds the file number to jump to
                              from segment 4 -- the terminal mode
                              segment)

ent "auto?", A$; if cap(A$[2,2])#"Y"; ldf 4   (if automatic login
                              is not wanted, the terminal mode is
                              started)
```

wsm 10, 187	(set the Mode Word register of the Serial I/O interface (7 bits/character and even parity), etc.)
wtc 10,1 wtc 10,37 wtc 10,0	(set register 4 of the interface to the proper value -- input from the modem is handled by an interrupt subroutine while output from the keyboard is in non-interrupt mode)
oni 10, "in"	(the interrupt handling subroutine is "in")
eir 10, 132	(enable the interrupt for input)
ent "press 1 on carrier detect", A	(actually any numbered key can be pressed. The program waits here until the operator presses 1, CONTINUE or just CONTINUE)
wtb 10,13; wait 100; wtb 10,13	(carriage return is used to prompt the "login" message from the computer)
"LOGIN" --> A$	(LOGIN is the word to search for before the login message is sent by this segment)
gsb "wt"	(the "wt" subroutine causes the program to wait until the search word (here LOGIN) is detected)
wtb 10, "login,...", 13	(the user should put in his/her own user id and telephone extension in place of ....)

```
"COMMAND-" --> A$                (the next searchword is COMMAND-)
gsb "wt"

wtb 10, "attach (hl,hllibx3693818,id=bandes)", 13  (the actual
                                 sequence of "commands" sent to the
                                 remote computer depends on how the
                                 data transfer is to be done.  Here
                                 we assume a catalogued procedure
                                 (DATTRAN) in library hllib will
                                 perform the transform)

gsb "wt"                         (COMMAND- is still the search word)

wtb 10, "library(hl)", 13

gsb "wt"

wtb 10, "dattran", 13

0 --> C

if C #19; gto +0                 (a control S is sent by DATTRAN when
                                 it is ready to receive data.  This
                                 causes the "send data" segment to be
                                 loaded)

trk 0; ldf 5                     (load segment 5 to continue)

wt:  len(A$)->A                  (start of the wait for search-word
                                 subroutine -- determine the length
                                 of the word)

sfg 2                            (set flag 2 -- it is cleared in the
                                 I/O input interrupt subroutine "in")
```

```
"w0":  0->I; 0->L               (whenever the input string fails to
                                 match the search-word, go back to
                                 this point.  I points to the charac-
                                 ter in the search-word while L is
                                 the number of sequentially matching
                                 characters found in the input
                                 stream)

"w1": num(A$[I+1->I,I])->B      (convert the next search-word
                                 character for comparison purposes)

      if flg2; gto +0           (wait here until the next character
                                 arrives)

      sfg 2                     (set flag 2 for the next character)

      if B #C; gto "w0"         (no match so start over)

      L+1->L; if L=A; gto "we"  (increment the number of the
                                 matches counter and return if they
                                 equal the number of characters in
                                 the search-word)

      gto "w1"                  (get the next characters to be
                                 compared)

      "we": dsp A$[1,A]; wait 250  (display the found search-
                                 word.  If it is not found in a
                                 reasonable length of time, something
                                 is wrong)

      ret

      "in": rdb(10)-->C         (start of input interrupt handler
                                 subroutine -- put the new character
                                 in C)
```

if C=5; 0->C	(when the interrupt register is set, a value of 5 is returned. This is changed to 0 since many steps wait on C=0)
eir 10,132	(the enable interrupt must be reset)
cfg 2	(clear flag 2 to indicate that a new character has been read)
iret	(interrupt return statement)

<u>segment 4</u> (terminal mode) - file 4

wsm 10, 187 wtc 10, 1 wtb 10,37 wtc 10,0	(set the serial 1/0 registers)
0->I; ""->A$;0->C	(initialize some variables)
oni 10,"in1"	(indicate the interrupt routine for input from the modem)
eir 10, 132	(enable interrupt)
on key "key"	(enable keyboard interrupt -- with interrupt routine "key")
"loop": if C=0; gto +0	(this is the start of the main loop where input from modem is looked for. C is 0 is no input is available)

```
        if C-19; gto "ot"        (|S is the escape from terminal
                                 mode.  It may come from the modem or
                                 keyboard)

        gsb "chk"                ("chk" is the subroutine used to
                                 display characters on the LED
                                 display of the 9825A)

        gto "loop"

        "ot":  on key            (exit point -- disable keyboard
                                 interrupts)

        ldf Q                    (load file Q, the next segment to be
                                 executed)

"key":  key->K                   (start of keyboard interrupt routine
                                 -- put key value in K)

        if K=0; gto "jump"       (on initialization of "key", want to
                                 jump out immediately)

        wtb 10, asc K            (send the ASCII equivalent of the
                                 key to the modem)

        asc K->C                 (store in C for display)

        gsb "chk"                (display the new character on the
                                 LED display)
```

Note: If the remote computer echoes the transmitted character (sometimes called "duplex" operation), the statements asc K->C and gsb "chk" should be deleted.

```
"jump": kret                     (keyboard interrupt return state-
                                 ment)
```

```
"chk":  C->K;  0->C           (put C in K for display; reset C to
                              0 for main loop purposes)

if K=13; gto "clear"          (clear the display on carriage
                              return)

if K=24; gto "clear"          (clear the display on ↑X)

if K#8; gto "disp"            (if have ↑H, then reset the display
                              in the following statement)

220->K; max(I-1,1)->I; gto "disp"  (setting K to 220 causes the
                              character corresponding to K to
                              blink decrement I, then display)

"clear":  ""->A$;  0->I       (clear the display array and reset
                              I)
32->K                         (the character at A$[1] will be
                              displayed; set it to a space)

"disp":  char (K)->A$[1+1->I,I]  (start of the character display
                              section; store the character equiva-
                              lent of K as the next 1 character
                              substring in A$)

dsp A$[max(1,len(A$)-31),max(32,lenA$))]  (display the "latest"
                              32 character substring of A$)

if K=220; I-1->I              (if ↑H was pressed, I has to be
                              decremented again in order to
                              overwrite the previous character)

ret                           (return from "chk")
```

```
"inl":  rdb (10)->C          (start of modem input interrupt
                              routine)

    if C=5; 0->C              (this is done since the interrupt
                              routine is entered during register
                              setting)

    eir 10, 132               (enable interrupt on exit since it
                              is disabled on entrance to "inl")

    iret
```

<u>segment 5</u> (transmit data) - file 5

```
oni 10,"in"                   (show address of the modem input
                              interrupt routine local to this
                              segment)

eir 10,132                    (enable interrupt)

0->C                          (initialize the character variable.
                              Note that in this segment a wait for
if C=0; gto +0                ↑S is done before sending the 2 data
                              records)

trk 1; ldf 0,A$               (load the first data record)

dsp "sending", A$[1,10]       (A$[1,10] should contain the label
                              which identifies the data to be
                              sent)

for J=1 to 130                (the "header" record contains 130
                              characters)

wtb 10,A$[J,J]                (send them)
```

```
next J

wtb 10,13                     (send end-of-record carriage return)

A$[1,10]->D$                  (save the header for possible later
                              use in segment 6)

ldf 1,A$                      (load the next record)

itf (A$[1,2])->A              (the first 2 bytes contain the
                              number of samples in "split integer"
                              form)

dsp "sending",A,"samples"     (let the operator know that the data
                              samples are being transmitted)

fmt f 4.0                     (this format is used by wrt -- note
                              that values range from integer 0 to
                              9999)

wrt 10,A                      (transmit the first "record": the
                              number of samples.  Note that the
                              wrt statement causes an automatic
                              carriage return/line feed to be sent
                              after each data list sent)

1->K; 0->E                    (K is the substring pointer; E holds
                              the sum of all values to be used for
                              "checksum" purposes)

for I=1 to A

itf (A$[K+2->K,K+1])->B       (get the next 2 byte data value and
                              convert it to "normal" form)

E+B->E                        (partial checksum)
```

```
    wrt 10,B                    (transmit the next "record")

    next I

    trk 0; ldf 6                (load the next program segment at
                                end of data transmission)

    "in":  rdb(10)->C           (the "local" modem input interrupt
                                routine)

          if C=5; 0->C

          eir 10,132

          iret
```

<u>segment 6</u> ("wrap-up) - file 6

```
    dim B$[2]                   (B$ is used to store a 1 or 2
                                character user response to a query)

    oni 10,"in"                 (for another input interrupt rou-
                                tine)

    eir 10,132

    fmt f12.0                   (the checksum is sent as an integer
                                with I12 format (the largest it can
                                be is 49995000))

    wrt 10,E                    (send the checksum)
```

for J=1 to 3	(this little loop is to read 2 line feeds and a carriage return sent by the receive program before the checksum message)
0->C	
if C=0; gto +0	(wait here for the next character -- note that any nulls sent are 0)
next J	
0->I	(initialize I for the receive message loop)
"12": 0->C	(start of receive message loop)
if C=0; gto +0	
if C=13; gto "pr"	(if carriage return, break from loop)
char (C)->A$[I+1->I,I]	
gto "12"	
"pr": prt A$[1,I]	(print and display the message. Here the subroutine which receives
dsp A$[1,I]	the transmitted data returns the number of samples, the checksum value transmitted and the checksum value calculated from the data)
"COMMAND-" ->A$	(used in the search-word subroutine -- see segment 3)

```
gsb "wt"                          (wait until the "COMMAND-" prompt is
                                  received)
ent "store and/or run?",B$        (store the transmitted data or
                                  master tape and/or run the FTS24
                                  program with this data as input?)

if cap(B$[1,1]#"S" and cap (B$[2,2]#"S"; gto "r"

wtb 10, "catalog(tapel,",D$,",id=barrett)",13  (tape 1 is the
                                  transmitted data file stored on
                                  disk.  Make it a permanent file
                                  before batching the file copy job)

gsb "wt"                          (wait for the next "COMMAND-"
                                  prompt)

wtb 10, "batch,tape 3,input,here",13  (tape 3 is a file copy job
                                  produced by DATTRAN -- see Section
                                  V)

gsb "wt"

"r":  if cap(B$[1,1])="R" or cap(B$[2,2])="R"; wtb 10,"fts14",13

7->Q; trk 0; ldf 4                (load the terminal mode segment for
                                  normal Intercom interaction with
                                  FTS14 program or for logout.
                                  Segment 7 (Q-7) will be loaded from
                                  that segment if |S is keyed)
```

This segment also contains the "wt" and "in" subroutines as described for segment 3.

segment 7 (next experiment) - file 7

This segment is exactly the same as segment 1 (file 0) except that the dim statements have been removed. This segment will be loaded, and the operator can continue as above with a new experiment (see segment 1).

V. The CDC-6600 Data Logging Software

The companion software to the HP-9825A software is the simple Fortran program, DATTRAN listed below. It interfaces with segment 5 and segment 6 of the HP software. The operation of the program is as follows.

(1) Files TAPE4 and TAPE 5 are "connected" to the terminal for serial input and output. TAPE4 is for the standard 64 character display code I/O while TAPE5 is used for "ASCII-128 I/O", mainly so that control characters can be sent.

(2) Variable CS is set to control S (octal 23). Note that when ASCII-128 is used for output, that the first 12 bit "byte" of the word is used for "carriage control" which basically means the number of line feeds before the next byte is sent. Thus only the second byte of CS has significance here. (All other "characters", including carriage returns and line feeds are ignored in the corresponding part of the ›-9825A software.)

(3) The header record is read into 13 words (130 characters) of BUF.

(4) Another control S is sent to indicate readiness to receive the next segment of data.

(5) The number of sample points is read (and decoded) into NUM using the I4 format (4 characters per integer).

(6) The sampled data is read (and "decoded") into IBUF as up to 25 "records" of 130 values each.

(7) The data file, TAPE1, is written. (Note that the number of samples is added to the header record.) This is the "mass storage" file which is used to feed data to the FTS14 system of programs (see Section VI).

(8) The checksum value computed by the HP-9825A is read and decoded using I12 format.

(9) The number of samples, received checksum and computed checksum are sent as part of a message.

(10) The "remote batch job", BARCY, which may be used to copy the data file (TAPE1) to the master tape is written.

```
PROGRAM DATTRAN(TAPE1,TAPE3,TAPE5=/260,TAPE5,OUTPUT)
DIMENSION BUF(13), IBUF, IDATA(2)
DATA (CS=0023 0023 0000 0000 0000B)
CALL CONNEC(4)
CALL CONNEC(5,1)
WRITE(5,6)CS
READ(4,2) (BUF(K),K=1,13)
WRITE(5,6) CS
READ(4,1) NUM
ENCODE(10,1,NUMC) NUM
ISUM=0
ICOUNT=0
DO 100 I=1,25
DO 100 J=1,130
READ(4,1)IA
ISUM=ISUM+IA
IBUF(J,I)=IA
ICOUNT=ICOUNT+1
IF(ICOUNT .EQ. NUM) GOTO 120
```

```
100  CONTINUE
120  WRITE(1) BUF, NUMC
     DO 150 K=1,I
150  WRITE(1) (IBUF(J,K),J=1,130)
D(4,5) IA
     WRITE(4,3) NUM,IA,ISUM
     WRITE(3,4) BUF(1)
1    FORMAT(I4)
2    FORMAT(13A10)
3    FORMAT(1H+,I5" SAMPLES, "I12,1XI12)
4    FORMAT("BARCY,CM6000,T16",T50,"1244",T58,"BARRETT"/
    $"VSN(MFILE=CC3666)"/
    $"REQUEST(MFILE,MF,E,RING)"/
    $"ATTACH(TAPE1,"A10",ID=BARRETT)"/
    $"LABEL(F1,M=MFILE,W,L="A10"/
    $"COPY(TAPE1,F1)")
5    FORMAT(I12)
6    FORMAT(A10)
     END
..
```

VI. The FTS14 Interface

The major components of the FTS14 system are outlined in Appendix A. The input to this system is one or two files designated TAPE1 and TAPE2. Two files are used if transmisivity must be calculated by dividing the values of transmission with a sample by those without a sample. Here we assume that this division has already been done -- so there is only a single file input.

As mentioned in the Appendix, there are two "equipment specific" subroutines in the FTS14 system which must be tailored to match various characteristics of the equipment being used to gather spectrometer data. In the example specifically described in this chapter, the main thing to be done is to compute wavenumber from the given data on sample rate and scan speed and to transpose the data to put it in sequential order by increasing wavenumber (as oposed to wavelength). The two subroutines involved are called by their "generic" names SPTROHD and DSKGDAT. They are given the equipment specific names of DLGHEAD and DLGGDAT, respectively.

Briefly, DLGHEAD, reads the header record produced by DATTRAN, computes the wavelength difference between samples, stores this data plus starting wavelength in /DLGHDT/ for use in DLGGDAT, and outputs the textual header information in /LET/ -- one character per word.

DLGGDAT reads the sampled data records produced by DATTRAN, converts them to floating point percent transmission, stores them in blank common and then transposes them to wavenumber order. In addition, the wavenumber associated with each sample value is computed using data from /DLGGH/. DLGGDAT also returns the number of samples in /KIT/ and an error flag value in /SPTFLAG/.

```
      SUBROUTINE DLGHEAD(IU)
C
C REVISION -- MAY 15, 1981
C AUTHOR   -- BARRETT,TB
C PURPOSE  -- READ AND DECODE THE HEADER RECORD PRODUCED BY DATTRAN.
C
C PARAMETERS -
C  IU  -  TAPE UNIT TO READ
C     THIS IS THE "SPTROHD" SUBROUTINE FOR THE "DATA LOGGER" SYSTEM
C     I.E. THE HP-9825A CALCULATOR AND PERIPHERALS AND PROGRAM DATTRAN
C     WHICH READS THE DATA FROM THE 9825.  THE HEADER RECORD IS OF THE
C     FORM-
C     WORD 1 - FILE NAME
C     WORD 2-9 - TEXT DATA (SHOUE THICKNESS WITH UNITS)
C     WORD 10 - STARTING WAVELENGTH (ANGSTROMS)
C     WORD 11 - SAMPLE RATE (SAMPLES/SECOND)
C     WORD 12 - SCAN SPEED (ANGSTROMS/MINUTE)
C     WORD 13 - END WAVELENGTH (ANGSTROMS)
C     WORD 14 - TOTAL NO. OF SAMPLES
C
C     THE NUMERICAL VALUES IN WORDS 10-14 ARE IN CHARACTER FORM
C     WORDS 10-13 CAN BE CONVERTED TO FLOATING POINT USING E10.0
C     WHILE WORD 14 CAN BE CONVERTED USING I10.  THE END WAVELENGTH
C     MAY BE NONSENSE AND SHOULD NOT BE USED.
      COMMON /DLGHD/ NUM,DELW,FSW
      COMMON /LET/ MSG(80)
      COMMON /DLGHDT/ FILNM,HD(8),SW,SR,SP,EW,NC
      DIMENSION D(5)
      EQUIVALENCE (SW,D)
      READ(IU) DATA
      WRITE 1,FILNM,SW,EW,SP,SR,NC
      WRITE 2,HD
      DECODE(50,3,D) FSW,FSR,FSP,FEW,NUM
      DECODE(80,4,HD) MSG
      DELW=FSP/(60.*FSR)
      RETURN
```

```
1       FORMAT(*0 FILE    *A10*  START WL*A10* END WL  *A10/
       $*  SCAN SPD*A10*  SAMP. RATE*A10*  NO. SAMPLES*A10)
2       FORMAT(*0 TITLE LINES -- */H0,8A10)
3       FORMAT(4E10.0,I10)
4       FORMAT(80A1)
        END


        SUBROUTINE DLGGDAT
C
C  REVISION -- MAY 15, 1981
C  AUTHOR   -- BARRETT,TB
C  PURPOSE  -- GET DATTRAN DATA FROM FILE TO XARRAY,YARRAY
C
C THE DSKGDAT SUBROUTINE FOR THE DATA LOGGING SYSTEM
C CONSISTING OF THE HP 9825 CALCULATOR + PERIPHERALS AND
C PROGRAM DATTRAN WHICH READS THE DATA FROM THE 9825
C
C IN THIS VERSION IT IS ASSUMED THAT THERE IS ONLY ONE INPUT
C FILE (DIVISION HAS ALREADY BEEN DONE).  THE TRANSMISSION
C DATA IS STORED IN RECORDS OF LENGTH 130 WORDS WHERE EACH
C WORD IS AN INTEGER FROM 0 TO 9999 REPRESENTING TRANSMISSION
C VALUES FROM 0% TO 99.99%.  THE DATA IS IN SEQUENTIAL ORDER
C BY WAVELENGTH AND MUST BE INVERTED FOR ORDERING BY
C WAVENUMBER.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
        COMMON /KIT/ K,KSAV,IT,ITSAV
        COMMON /SPTL/ NF,DUM(2)
C THIS COMMON IS NOT USED NOW.
        COMMON /DLGHD/ NUM,DELW,SW
        COMMON /SPTFLAG/ IFLAG
        COMMON XARRAY(6000),YARRAY(6000),IDATA(130)
        IFLAG=2LOK
        K=II=0
```

```
100     READ(1) IDATA
        IF (EOF(1) .EQ. 0) GOTO 160
        IFLAG=3LERR
        WRITE 1,II
        GOTO 600
160     II=II+1
        DO 110 J=1,130
        K=K+1
        IF (K .GT. NUM) GOTO 170
        XARRAY(K)=1.E8/(SW+(K-1)*DELW)
        IF ((I .LT. 0) .0. (IDATA(J) .GT. 9999)) GOTO 500
        YARRAY(K)=IDATA(J)/100
110     CONTINUE
        GOTO 100
170     CALL REVERSE(XARRAY,K)
        CALL REVERSE(YARRAY,K)
        GOTO 600
500     WRITE 2,II,J,IDATA(J)
        IFLAG=3LERR
600     RETURN
1       FORMAT(* END OF INPUT PREMATURE AT RECORD*I5)
2       FORMAT(* BAD DATA AT RECORD*I4*,WORD*I4*,DATA=*I10)
        END
```

APPENDIX A - The FTS 14 System Outline

An outline of the FTS14 system follows with a description of each module of the system. Particular attention should be paid to the SPTROHD and DSKGDAT modules since these are the equipment specific modules which usually have to be written, or at least modified, for each spectrometer. Note also should be made of subroutine FNUMBR which locates sample thickness data in the header text.

SPTRODR

main program

- declares input, output

tape 1, tape 2	(2 data files)
scr 1, scr 2	(2 scratch files
tape 7	- job file
tape 51	- namelist save file

This main program does the following:

(1) initializes some data.
(2) calls SPTROCD to read da a cards.
(3) calls SELLMEI to load dispersion data.
(4) initializes the plotter.
(5) calls SPTROPL if simple plot is required.
(6) calls SPTRSTR for stacked plots.
(7) calls SPTDIFF for difference plots.
(8) ends plotting

SPTROCD

- reads input data "cards"

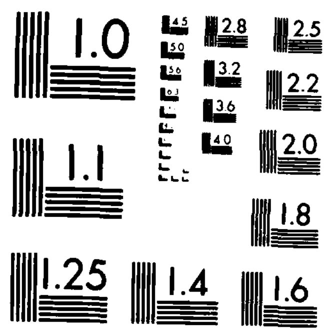
This subroutine does the following:

(1) assigns default data using a DATA statement.
(2) resets some selected data.
(3) sets system traps for erroneous data.

AD-A149 726 TOPICS IN OPTICAL MATERIALS AND DEVICE RESEARCH - III 2/2
VOLUME 1(U) PARKE MATHEMATICAL LABS INC CARLISLE MASS
T B BARRETT ET AL. AUG 84 RADC-TR-84-73-VOL-1

UNCLASSIFIED F19628-81-C-0052 F/G 17/8 NL

END
FILMED
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

SPTRODR

main program

- declares input, output

tape 1, tape 2	(2 data files)
scr 1, scr 2	(2 scratch files
tape 7	- job file
tape 51	- namelist save file

This main program does the following:

(1) initializes some data.
(2) calls SPTROCD to read data cards.
(3) calls SELLMEI to load dispersion data.
(4) initializes the plotter.
(5) calls SPTROPL if simple plot is required.
(6) calls SPTRSTR for stacked plots.
(7) calls SPTDIFF for difference plots.
(8) ends plotting

SPTROCD

- reads input data "cards"

This subroutine does the following:

(1) assign default data using a DATA statement.
(2) resets some selected data.
(3) sets system traps for erroneous data.

(4) sets data values to those of the previous run by reading TAPE51 if it exists.

(5) gets new data, if any, and rewrites TAPE51.

(6) computes various plot quantities.

(7) writes data values to output and a summary of data to TAPE7.

SPTRSTK

- does "stacked" plots

This subroutine does the following:

(1) draws and labels the "z-axis".

(2) calls SPTROPL to process data from a spectrometer "run"

(3) plots a legend if required.

(4) calls SPTROCD for the next set of data card(s).

(5) repeats 2,3,4 until there is no more data or the maximum z-value has been exceeded.

SPTDIFF

- controls plotting of difference plots

This subroutine does the following:

(1) calls SPTROPL to process data for the first experiment (A).

(2) moves the plot axis for the next set of data.

(3) calls SPTROPL to process data for the second experiment (B).

(4) computes the difference in absorption (A-B) in the wavelength overlap region.

(5) draws axes and plots absorption difference as a function of wavelength. Also prints the difference using PRINT6.

SPTROPL

- processes (and plots/prints) one set of spectroscopy data

This subroutine does the following:

(1) initializes some data for "decoding" sample thickness information.

(2) reads a set of data from multi-file tape to disk (if the files already exist as TAPE2 and TAPE3, a dummy MFTODSK subroutine can be used). Note that 2 data files are required for difference plots and for experiments where transmission must be computed by performing A/B.

(3) reads the header records by calling SPTROHD for data files. Note that SPTROHD is specific to the type of data being processed and a SUBST statement must be used to cause the correct "front end" processor to be loaded. See a specific processor, F14HEAD, for the output requirements of the header processing subroutine.

(4) decodes sample thickness data if required by a call to FNUMBR.

(5) encodes header into TITLE for later plot display.

(6) stores the header in the temporary file.

(7) repeats (2) through (6) if two data sets are to be processed.

(8) fills up the wavenumber, transmission arrays (XARRAY(6000), YARRAY(6000) in unlabeled common) using data specific subroutine DSKGDAT. Note that XARRAY must contain the wavenumber in wavenumber ascending order in units of cm^{-1} while YARRAY must contain the corresponding <u>percent</u> transmission. Instruments which output data in ascending order by Angstroms must have the data transposed and the conversion $w = 10^{-8}/\lambda$ where λ is the wavelength in Angstroms. See F14GDAT for the DSKGDAT used for the FTS14 spectrometer.

(9) processes (e.g., smooths) the transmission data, computes absorption, plots and prints either transmission or absorption or both. This is done by a call to DATPLOT. See DATPLOT for details.

(10) draws the plot axes with a call to AXESPLT.

SPTROHD(IU)

This data specific subroutine reads one or more "header" records from unit IU (either 1 or 2 for TAPE1,TAPE2) and returns a single header "message" in common area /LET/. In addition, it may print header information on a title page which introduces the results printed by the PRINT6 subroutine. For example, F14HEAD, which is the SPTROHD for the FTS14 spectrometer, decodes the header produced by the NOVA computer used by this spectrometer to obtain spectrometer parameters as well as one or two lines of information about the experiments which were given by the experimenter.

The "message" returned by SPTROHD is used for labeling plots and for labeling each page of the PRINT6 document. It should consist of 80 printable (may be blanks) display code characters, 1 character per word, left justified blank fill. In addition, the sample thickness should appear in this message in the form "x(y)" where x is a number in fixed point form (e.g., 1.664) and y is the characters "IN" for inches or "CM" for centimeters. (See FNUMBR for more explicit details.)

It may be advantageous to use SPTROHD to pass header information to DSKGDAT, the other data-specific subroutine. For exapmle PELHEAD, the Perkin-Elmer header subroutine, gets wavelength data from the header which is used by PELGDAT, the DSKGDAT for the Perkin-Elmer spectrometer.

FNUMBR(IU,F,INDX,CMATRIX,LM)

The purpose of FNUMBR is to extract a number with a dimension associated with it from a line of text and return it as a dimension in "standard" units. The following arguments are used:

IU - the standard units code -- left justified character string with up to 10 characters. ("M", for centimeters, is the standard used in SPTRODR.)

F - returned by FNUMBR -- the floating point number in standard units.

INDX - a list of length LM of units abbreviations for which a conversion is required.

CMATRIX - conversion table matrix. For example, if the standard unit is found at location IS in INDX and the text unit is found at location IT, the conversion performed is:

F = FNUM * CMATRIX(IS,IT)

where FNUM is the number found in the text with its associated dimension (INDX(IT)).

LM - length of INDX. CMATRIX is LM by LM.

The text passed to FNUMBR is in common area /LET/ with length 80, one left justified character per computer word. The dimensioned number to be decoded and perhaps converted to standard units should have the form XD_1YD_2 where

X is a string of (display code) integers with perhaps a single decimal point (the letter O is converted to 0 and letter I to 1).

D_1 is the left delimiter for the units code (either "(" or "8" -- the lower case equivalent of "(" on teletype keyboards).

Y is the units code, (e.g., "IN","CM"). The units code may be up to 10 characters in length, but should have a matching code in INDX.

D_2 is the right delimiter for the units code. This delimiter may be any non-alphabetic character.

DSKGDAT

This is the data dependent subroutine which interfaces the "FTS14 system" with the actual spectrometer data, regardless of its source. The specific requirements of this subroutine are given below.

The main function of DSKGDAT is to put wavenumber data in a segment of unlabeled common along with corresponding transmission data in another segment of unlabeled common. More specifically, the first 6000 words of unlabeled common may contain the wavenumber in units of cm^{-1} and in ascending order while the next 6000 words may contain the corresponding transmission in percent. In addition, the actual number of wavenumber (or transmission) values must be recorded in labeled common area /KIT/.

The FTS 14 system is capable of handling the situation where percent transmission must be calculated by dividing transmission with sample by transmission without sample (two separate files) in addition to the more common situation where percent transmission exists in a single file.

The following summarizes the input/output characteristics of DSKGDAT:

<u>input</u>

/SPTLBL/NUM,DUM(2)

where integer NUM is the number of input files. If NUM=1, the transmission data is assumed to exist in a single file (usually TAPE1). If NUM=2, the transmission data can be computed by dividing the contents of TAPE1 by that of TAPE2 (usually).

<u>output</u>

//XARRAY(6000),YARRAY(6000)
/KIT/K,K1,K2,K3 (K1->3 not used)

where floating point XARRAY contains the wavenumber values in ascending order (units of cm^{-1}) and floating point YARRAY contains the corresponding percent transmission values. Integer K contains the total number of wavenumber-transmission pairs. Note that XARRAY(1) should contain the first wavenumber value and similarly for YARRAY(1).

/SPTFLAG/IFLAG

IFLAG should be set to 2LOK if data is found on the files as expected but otherwise set to 3LERR.

<u>DATAPLOT</u>

The main function of this subroutine is to plot (and print) either transmission data and/or absorption data which is computed from the transmission data, reflectivity data and sample thickness. The plot (print) interval as well as scale is under user control. A certain amount of data manipulation such as smoothing can also be done by DATPLOT. The following is a summary of DATPLOT and its action under various values of user supplied data.

(1) The transmission data is converted according to y = y+.01(ADDF). Normally ADDF has the default value of 0.

(2) The transmission data is "smoothed" using a triangular weighting function with a wavenumber width given by SWMN. The default value of SWMN is zero for no smoothing.

(3) The wavenumber of maximum transmission and the corresponding maximum transmission (Y) is found.

(4) An average maximum transmission (Y) over a wavenumber interval given by AVRG about the wavenumber of maximum transmission is found. The default value of AVRG is 0 for no averaging.

(5) Transmission values are transformed ("squashed") according to values assigned to TRANSMX (default-value 92.30709) and SQUASH (default value 1) as follows:

(a) if TRANSMX>1 and SQUASH ≠ 1, then T->S * T

(b) if TRANSMX>0 and SQUASH=1, then T->T·TRANSMX/Y where Y = maximum transmission or average maximum transmission.

(c) if TRANSMX=0, then no squashing is done.

(d) if TRANSMX<0, then

$$T \rightarrow T \cdot \frac{(1-Q)}{(1+Q)\cdot .01 \cdot Y}$$

where Q = reflectivity at the wavenumber of maximum transmission.

(6) Reflectivity values for the sample are calculated according to various algorithms provided TRANSMX $\geq$ 0. These algorithms are discussed elsewhere (see, for example, RADC-TR-78-61, Parke Mathematical Laboratories).

(7) Transmission values (or modified transmission values) are plotted and printed as a function of wavenumber provided the transmission scale increment, TINC, is greater than 0.

(8) Absorption values are calculated, plotted and printed provided the user given reflectivity value, REFL, is not equal to zero. (Note that if TRANSMX is less than zero, absorption will be calculated and plotted using calculated values of reflectivity.) Absorption plots can be on a semi-logarithmic scale while transmission plots are always linear.

AXESPLT

This subroutine finishes all plots by causing the axes to be drawn including scale information and axis labels. Other plot annotation such as the printing of plot titles is also done through AXESPLT.

MFTODSK(ILABEL,IUNIT,IGOOD)

This subroutine reads a multi-file tape (i.e., a tape which requires the MF parameter on the REQUEST command) of spectrometer data and stores the required file on disk. The file is, of course, sequential and must have the following characteristics:

maximum record length = 5020 characters
record type = W

The MFTODSK arguments are used as follows:

ILABEL - label of the file to be copied (up to 10 characters, left justified, blank fill).

IUNIT - unit number of the disk file for copy.

IGOOD - if the required file is found, the value of IGOOD returned by MFTODSK is 5LFOUND. Otherwise it is 3LERR.

Chapter 7

"DIRECT" DATA-LOGGINE USING THE 9825A CALCULATOR

Introduction

In the last chapter, procedures and software for using the HP-9825A Calculator for data logging were described. In this chapter similar procedures and software are described for a system which sends binary data directly to the remote computer. In the last chapter the "front end" of the system was a spectrometer attached to a digital voltmeter with a GPIB (General Purpose Interface Bus). In this chapter the system described uses a front end consisting of a Hamamatsu C-1000 camera with a camera control unit which has a M999-04 GPIB. Thus Fig. 1 in the last chapter remains the same except that Spectrometer <-> C-1000 camera, Digital Voltmeter <-> Camera Control Unit.

This chapter is organized much like the last one. In section I, an overall description of the system and its operational environment are presented. In Section II, the data gathering and transmission procedures are described. Section III lists and annotates the HP-9325A software; section IV describes the corresponding CDC6600 software. Section V is a supplement to both the last chapter and this chapter, describing the use of special keys on the calculator while it is used as a terminal. Appendix A is a PL/I listing for a program to test the communications protocol between a micro-computer and the CDC6600.

I. THE SYSTEM

Figure 1 of the last chapter remains the same except the Spectrometer/ Digital Voltmeter front end is replaced by the Hamamatsu Camera/Camera Controller (with M999-004, GPIB Interface Unit).

The front end (referred to below as the camera) is capable of gathering an image or part of an image as a set of 1 or more "scan" lines wherein each line is sampled at 256,512 or 1024 points depending on the value of a camera parameter called "interlace" (INT for short). The x-position of each scan line is determined by the value of another camera parameter, the x-coordinate (XCO for short). A video frame (partial or complete) may be gathered by sending a starting x-coordinate to the camera and to then command the camera to send the first scan line (using the "video input" execute command, called VI). Subsequent scan lines are obtained by sending the "video input increment" (VII) command. (Note that the incrementing is done before the line is obtained.) A complete frame (starting at x-coordinate 0 and ending at x-coordinate 1023) consists of 256,512 or 1024 scan lines cooresponding to INT values of 1, 2 and 4 respectively.

The data returned by the camera can be formatted to be either ASCII or binary. In ASCII, 3 bytes are sent to represent a number between 000 and 255 along with a space to separate adjacent numbers. In the system described here, only binary data is gathered wherein each byte represents a pixel intensity value of 0 to 255.

The 9825A software, described in Section II, includes provisions for allowing the user to set the interlace and starting x-coordinates. There are also provisions for either automatic or manual "login" to the remote computer (here assumed to be a CDC6600). Automatic login includes start-up procedures for the remote software which accepts and stores the transmitted data. Under manual login, the user must login (using the 9825A in "terminal" mode) and, in addition, he must attach and start the remote software.

A video frame is transmitted as a stream of 8 bit codes in 256 byte blocks (in CDC Inntercom parlance, ASCII-256 mode is used). At the end of each block a checksum is transmitted. If the checksum value computed by the CDC program does not agree with that sent, the block of data is retransmitted. At a transmission rate of 1200 baud (120 bytes/second), it takes a minimum of 7 minutes to transmit a 256 x 256 frame, about ½ hour to transmit a 512 x 512 frame and 2 hours to transmit a 1024 x 1024 frame. The actual transmission time is about twice these values because of time required to gather the data, and "turn around" time for checksum protocol.

The data is stored on disc at the CDC6600, 5 bytes per CDC 60-bit word. A full 1024 x 1024 frame requires around 2400 physical record units (pru's) of disc storage space. The first record of this data file contains only the number of scan lines and the number of pixels per line. All other records contain a block of 256 pixels stored 5 pixels per word. Thus each such record is 52 words long with the last word zero filled.

A catalogued procedure called BINRECP is invoked by the user, or by the automatic login procedure, to receive data from the 9825A. This procedure computes the checksum, and if the checksum "checks", stores the data.

Note that in ASCII-256 mode all bytes are interpretted as data except for the byte(s) which represent carriage return (CR). The 2 carriage returns are (decimal) 13 and 141. Whenever one of these bytes is received, it is interpretted as an end of record and all data received up to that point (starting from the previous CR) is returned from the peripheral processor to the main processor (and hence the program). To circumvent this protocol, all pixels which have a value of 13 are changed to 12 and all with a value of 141 to 142.

II. DATA GATHER AND TRANSMISSION

Equipment set-up

The 9825A must be equipped with a 98034A HP-1B Interface, typically with the factory settings (see "HP-1B Interface Installation and Service Manual"). In particular the external select code switch should be set to 7. The other interface required is the 98036A Serial I/O interface with the select code switch set to 10. In addition, the bit rate switch, adjustable externally, should be set to 5 for 1200 bits per second and to 7 if 300 bits per second are used. The mode word which controls character length etc. is set by the program and need not be set internally. The line going to pin 3 of the 98036A should be connected to pin 2 of the modem and the line going to pin 2 of the 98036A to pin 3 of the modem. Depending on what modem is used, it may also be necessary to set the data set ready input to the modem.

The 98034A is connected to the Camera Interface and the interface controls set for automatic operation.

If the 9825A requires external ROM's the following ROM's must be in: STRINGS, SYSTEMS PROGRAMMING, GENERAL I/O. The MATRIX ROM must be removed.

The Camera Control address switch should be set to 2.

Operation

The "Data Logger" tape cartridge should be inserted in the 9825A. This tape cartridge is also the one used for data logging as described in the last chapter. If the 9825A is started with the tape in place, it will start the Data Logging program which must then be aborted (use RESET). The correct segment to load is file 8 (use LDF8 or LOAD-key) followed by 8. After loading file 8, press RUN.

The first prompt is "filename", to which the user should reply with an up to 10 character name of the file under which the data will be stored on the CDC6600. (Note that continue must be pressed after the filename has been entered.) The second prompt is "interlace-1, 2 or 4" to which the user should respond with the desired interlace number. The third prompt is "starting XCO" to which the user should respond with the desired starting x-coordinate in the range 0-1023. If XCO=0, a full video frame will be gathered; for INT=1,2,4 this corresponds to 256, 512, 1024 scan lines respectively. The number of scan lines, m, gathered is related to the values of XCO and INT by:

(1) m=[(1024-XCO)*INT/4].

This value is displayed for a second or so.

The next prompt is "auto?". If the user replies with "yes" or "y", the auto login section of the program is started; if "no" or "n", file 10, the "terminal" segement, is loaded and started. In either case, the user should then establish communications with the CDC6600. In auto-login mode, the prompt "press 1 on carrier detect" is displayed. In terminal mode there is no prompt. In the first case press 1 then CONTINUE; in the second press STORE (this is equivalent to CR, see Section V).

In auto mode, the user need do nothing further until all of the data has been transmitted to the CDC6600 as indicated by the message "end of program" on the 9825A display. In terminal mode the following Intercom commands must be entered -

```
LOGIN,  .............  (user name, password and telephone #)
ATTACH (WM,WMLIBX3693818,ID=BARRETT)
LIBRARY (WM)
ETL (1000)
BINRECP,FILE=filename,N=n,M=m.
```

where filename is the name of the file under which the data is to be stored.
n is the number of pixels/scan line
m is the number of scan lines

Each command should be followed by STORE (for CR) and n and m should "agree" with the data previously entered for the responses to "interlace" and "starting-xco".

During the data gathering and transmission phase, the following information is displayed on the 9825A-

"get" K,N$ "pixels" (K=scan line sequence no., the no. of pixels/scan line)

"send" K, "blk", J, "chk", P (K=scan line sequence no., J= block no. (1,2,3, or 4, P= 0 if this is first time the block is transmitted. It is incremented by 1 for each succeeding blk.)

After the frame has been transmitted, the message "end of program" is displayed. Following this, file 10 (terminal program) is loaded and the terminal mode is entered. The user can then issue further Intercom commands including LOGOUT if no more data is to be gathered and transmitted.

If another frame of data is to be gathered and transmitted, the above procedure can be restarted by pressing RESET and LOAD 8 (or LDF8). The auto mode should not be used for other than the first time through the program (as long as the user remains logged-in to the CDC). In terminal mode it is only necessary to issue the Intercom command -

RETURN(DEST)
BINRECP,FILE=filename,N=n,M=m

III. HP 9825A SOFTWARE

The HP 9825A software for data transmission consists of 3 segments (files 8,9 and 10). File 10 is the terminal mode segment described in detail in the last chapter.

<u>Segment 1 (camera formatting) file 8</u>

```
dim A$[500],D$[10],F$[10],I$[2],X$[5],M$[5],N$[5],A[1100]
                                   (all arrays are dimensioned here)
dim H$[1000];buf "buff", 1000,1
                                   (buffered i/o must be used for
                                   receiving data from the remote
                                   computer at 1200 baud)
wsm 10,175                         (set 98036A mode for 8 bit word, no
                                   parity)
wsc 10,37                          (set 98036A control word-enable
                                   transmit and receive)
eir 10,132                         (set 98036A for interrupt on received
                                   data)
tfr 10, "buff"                     (enable 98036A for buffered input)
ent "filename", F$                 (the file name prompt)
fxd 0                              (format for str function)
ent"interlace-1,2or4",T;str(T)->I$ (convert to string for transmission
                                   to 3.mera Controller)
wti 0,7                            (set select code to 7 for some of the
                                   following instructions)
wtb 70204,I$[2,2]                  (send the INT message -
                                   7 = camera control primary address
                                   2 = camera control secondary address
                                       which specifies INT.  Note that 4
                                       is translated to decimal 100 by
                                       the 9825A
                                   The string I$ consists of "n" where n
                                   = 1, 2 or 5 in ASCII.)
if iof 7=0;jmp0                    (wait for completion of the above
```

```
                                     instruction by observing the i/o flag
                                     for select code 7)
wti 7,144                            (set register 7 (for select code 7 as
                                     specified by the above wto 0,7) for
                                     the uni-line message, END.  This will
                                     cause the next character transmitted
                                     to include EOI.)
wtb 731,13                           (send carriage return (decimal 13)
                                     and the END message (end or identify
                                     line, EOI, active).  Note that 731
                                     refers to select code 7, primary
                                     address 31 and is used by the 9825A
                                     to output the following data with no
                                     preceeding listen address.)
ent "starting xco",X;str(X)->X$      (get starting coordinate and convert
                                     to string for transmission to camera)
wtb 70203,X$[2,len(X$)->L]
if iof7=0;jmp 0                      (wait for completion of the above
                                     instruction)
wti 731,144
wtb 731,13
wtb 70202, "2"                       (send the INPUT FORMAT camera control
                                     message, secondary address 2.  The
                                     ASCII data "2" is sent, indicating
                                     binary data is to be sent by the
                                     camera controller.)
if iof7=0;jmp0
wti 7,144
wtb 731,13
T256->N                              (the interlace no. is multiplied by
                                     256 to give the no. of pixels/line)
str(N)->N$                           (N is converted to ASCII for later
                                     messages)
int((1024-X)T/4->M                   (compute the no. of scan lines)
str(M)->M$                           (convert to ASCII for message)
```

```
dsp "M=",M; wait 1000                (display the value of M for "manual"
                                     log-in)
9->Q                                 (file 9 is loaded in reception of ^S
                                     within "terminal")
ent"auto?",A$;if cap(A$[1,1])#"Y";ldf 10
                                     (file 10 is "terminal")
0->S;0->J
"LOGIN"->A$                          (LOGIN is the first "key" word
                                     returned by the 6600)
ent "press 1 on carrier detect", A
wait 5000;wtb 10,13
gsb "wt"                             (wait for LOGIN prompt from 6600)
wtb 10, "LOGIN,TED,ES0647,8615551,SUP",13
                                     (login)
"COMMAND-">A$                        (COMMAND-is the next "key" word)
gsb "wt"                             (wait for it)
wtb 10,"ATTACH(WM,WMLIBX3693818,ID-BARRETT)",13
dsp "ATTACH"
gsb "wt"                             (wait for next command-prompt)
wtb 10,"LIBRARY(WM)"13
dsp "LIBRARY"
gsb "wt"                             (wait for next command-prompt)
wtb 10, "ETL (1000)",13
dsp "ETL"
gsb "wt"                             (the time limit must be extended)
wtb 10,"BINRECP,FILF=",F$[1,len(F$)->L],",N=",N$[2,len(N$)->L]
wtb 10,",M=",M$[2,len(M$)->L],13     (start procedure with parameters
                                     previously entered)
dsp "BINRECP"                        (wait for ^S)
gsb "cx"
if C#19;gto-1
trk 0;ldf 9
"wt":len(A$)->A                      (subroutine "wt" finds key words)
"w0":0->L;0->I
"w1":num(A$[I+1->I,I])->B
gsb "cx"
```

```
if B#C;gto "w0"
L+1->L; if L=A;gto "we"
gto "w1"
"we":dsp A$[1,A];wait 1000
ret
"cx":J+1->J;if J>S;gto "cy"
num(H$[J,J])->C
if C>127;C-128->C
ret
"cy":0->J
bred("buff")->H$
len(H$)->S;if S=0;gto "cy"         (wait here until one or more
                                    characters are in the input buffer)
gto "cx"
```

<u>segment 2 ("terminal") file 10</u>

```
0->I;""->A$                        (initialize the "display" variables)
on key "key"                       (enable keyboard interrupt)
"lp":
bred("buff")->H$                   (transfer contents of input buffer to
                                    work array)
len(H$)->S;if S=0;gto "lp"
for J=1 to S
num(H$[J,J])->K                    (convert from string to ASCII)
if K>127;K-128->K                  (in case bit 7 is a 1)
if K=19;gto "ot"                   (leave "terminal" on ^S)
gsb "chk"                          (display characters)
next J
gto "lp"                           (empty buffer again)
"ot":on key                        (disable keyboard interrupts)
ldf Q
"key":key->K;asc K->K              (get character from keyboard)
if K=0;gto "jp"
if K=19;gto "ot"                   (can also exit "terminal" with ^S
                                    from keyboard)
```

```
wtb 10,K                              (send to remote computer)
gsb "ck=hk"
"jp": kret
if ior(K=13,K=24);gto "cl"            (clear display if CR or ^X)
"chk": if K#8;gto"dp"                 (if have ^H then reset the display)
220->K;max(I-1,1)->I;gto "dp"         (setting K to 220 causes the charac-
                                      ter corresponding to K to blink)
"cl":""->A$;0->I                      (clear the display)
32->K                                 (set "cleared" string to space)
"dp":char(K)->A$[I+1->I,I]
dsp A$[max(1,len(A$)-31),max(32,len(A$))]
                                      (display the "latest" 32 character
                                      substring of A$)
if K=220;I-1->I                       (if ^H was pressed, I has to be
                                      decremented again in order to over-
                                      write the previous character)
ret
```

<u>segment 3 (data transmission) file 9</u>

```
0->r1                                 (zero checksum - error counter)
fmt f12.0                             (checksum data is sent with this
                                      format)
on key "ky"                           (enable keyboard interrupt)
for K=1 to M                          (start of main scan loop)
dsp "get",K,",",N$,"pixels"
if K=1;wtb 70208                      (gather the first scanline using VI)
if K#1;wtb 70209                      (gather subsequent lines using VII)
if bit(7,rds(7))=0;jmp 0              (look at bit 7 of register 7 (of
                                      select code 7) for the appearance of
                                      the SRQ message from the camera
                                      controller)
if iof7=0;jmp 0                       (may not be necessary)
```

```
wti 7,132                           (set register 7 so that ATN is
                                    activated for the following messages)
wtb731,95                           (send UNT (untalk) message)
wtb731,24                           (send SPE (serial poll enable))
if iof7=0;jmp0
wti7,128                            (turn off ATN)
rds(702)->A                         (read status byte from camera
                                    controller)
wti 7,132
wtb 731,95                          (send UNT again)
wtb 731,25                          (send SPD (serial poll disable))
if iof7=0;jmp0
wti 7,128
rdb(702)->V                         (get first datum by addressing the
                                    camera)
if V=13;12->V                       (can not send CR)
if V=141;142->V                     (this is also CR)
V->A[1]
for L=2 to N                        (N is the number of pixels/scan line)
rdb(731)->V                         (get subsequent data without
                                    addressing)
if V=13;12->V
if V=141;142->V
V->A[L]
next L
for J=1 to T                        (transmit the data as INT blocks of
                                    256)
(J-1)256->X;0->P
"send":dsp"send",K,"blk",J,"chk",P  (start of block send loop)
0->S                                (set checksum to 0)
bred ("buff")->A$                   (clear the buffer of any left over
                                    characters prior to getting control-S
                                    later on)
""->A$
for L=1 to 256
A[X+L->Y]->B
```

```
B+S->S                                   (compute incremental checksum)
wtb 10,B                                 (send the pixel in binary)
next L
wtb 10,1,1,1,1,1,1,1                     (send ones to pad the record)
wtb 10,13                                (terminate the record with CR)
P+1->P                                   (increment retry counter)
wtb 10,0                                 (seems to be necessary for timing)
wrt 10,S                                 (send the checksum using above
                                         format)
"fd":bred("buff")->A$                    (wait here for a character in the
                                         buffer)
len(A$)->F;if F=0;gto "fd"
for L=1 to F
num(A$[L,L])->C; if C>127;C-128->C
if C=6;wait 100;gto "entp"               (wait for ACK or NAK; if ACK go to
                                         next J)
                                         (NAK means retransmit)
if C=21; wait 100;wrt 16, K, S;r1 + 1->r1; gto "send"
next L
gto "fd"
"entp":
next J
next K
wait 1000;wtb 10,13,13                   (send a couple of carriage returns to
                                         end the procedure)
dsp "end of program"
on key                                   (load file 8 at the end of the
                                         terminal program see note)
prt "chk error=",r1                      (go back to "terminal")
wrt 16,"file=",F$[1,len(F$)]
wrt 16,"xco=",X$[2,len(X$)]
wrt 16,"M,m=",N$[2,len(N$)],",",M$[2,len(M$)]
stp
"ky":key->r2
wtb 10,13                                (send a CR if transmission of data
                                         from the 6600 is stopped)
kret
```

IV. THE CDC6600 SOFTWARE

This is a description of the catalogued procedure which is called, either in auto mode, or by the user in manual mode, to receive data from the HP-9825A.

(1) As noted in the procedure listing (on following pages), BINRCEP has 4 parameters: N = no. of pixels/scan line; M = no. of scan-lines; FILE = name under which the received data is to be filed; CHK = True or False, if True (default) then checksums are checked, if False, they are not. CHK = False is normally used only for program checkout.

(2) In the program statement, the source file is set to receive a maximum of 205 packed words (1024/5 = 204 + rem.). This is necessary only if the data is not sent in 256-byte blocks. The program may be modified at a later date to send the data in non-block form.

- file DEST is the disk file in which the received data is stored in packed form. In block transmission mode, each record after the first contains 52 words, the first 51 of which contain 5 bytes of data and the last of which contains 1 byte plus zero fill. The first record contains N, M (unformatted).

- file SOURCE is the binary input file, i.e. it is "connected" in mode 2 to receive "ASCII-256" data.

- file CONTRL is the "control code" output file. It is connected in mode 2 (ASCII-256) to output ^S, ACK and NAK.

- file TAPE1 is used for diagnostics. It can be examined using the COPY command to get a dump of what was received. This file may be eliminated in later versions of this program.

- file INPUT is connected in mode 0 (ASCII-64) to receive the checksum as a "normal" formatted record.

(3) The first output from the "control port" is ^S. This tells the sending program that the "source port" is ready to receive data. In the 9825, the ^S causes segment 3 (file 9) to be loaded and executed.

(4) The DO 200 loop is the main loop for receiving the M scan lines.

(5) The DO 100 loop is to receive 1, 2, or 4 blocks of data for each scan line.

(6) The 110 READ (SOURCE,3) ... causes binary data read into the PP (peripheral processor) to be transferred to the main computer memory in packed format. The CR sent by the 9825A at the end of a data block causes "completion" of this statement.

(7) The second output from the control port is ACK if the received checksum (READ (INPUT,4) ICHECK) agrees with the computed checksum, and NAK if it does not (unless CHK=False, in which case ACK is always sent).

Note that ACK's or NAK's are sent as pairs of characters in case one character gets garbled or lost in transmission.

```
"PROC,BINRECP,N=0,M=0,FILE=TEMP,CHK=TRUE.
.*
.* PML (NONE ASSIGNED)
.* REVISION -- JUNE 23,1982
.* AUTHOR   -- BARRETT,TB
.* PURPOSE  -- RECEIVE CAMERA DATA IN BINARY FORM. M SCAN LINES
.*                  OF N PIXELS EACH ARE RECEIVED. A CHECKSUM IS ALSO
.*         RECEIVED AFTER EACH SCAN LINE. IF THERE IS NO CHECK, THE SCAN
.*        LINE MUST BE RETRANSMITTED.
.*         FILE IS THE NAME OF THE PERMANENT FILE UNDER WHICH THE DATA IS
.*         STORED. NOTE THAT THE DATA IS IN PACKED FORM-- 5 BYTES/WORD.
.*       THE FIRST RECORD OF FILE CONTAINS N,M IN UNPACKED FORM,
```

```
.*      ALL OTHER RECORDS (EXCEPT POSSIBLY THE LAST) CONTAIN
.*       52 WORDS OF 5 BYTES EACH WITH THE LAST WORD IN EACH
.*        RECORD CONTAINING "EMPTY" BYTES.
.*       CHK IS USED TO TURN CHECK-SUMMING ON (T) OR OFF (F)
.*
.**********************************************************************
.*
.* * * * * * * * * * E X A M P L E * * * * * * * * * * * * * * * * * * * * *
.* ATTACH(WM,WMLWMLIBX3693818,ID=BARRETT)
.* LIBRARY(WM)
.* BINRECP,N=512,M=512,FILE=FRAME1.
.*
FTN5(I=STEPL,L=0,DB=PMD)
REQUEST(DEST,*PE)
RFL(75000)
LGO.
CATALOG(DEST,FILE,ID=BARRETT)
.DATA,STEP1
      PROGRAM BINREC(INPUT,DEST,SOURCE=/2050,CONTRL,OUTPUT,TAPE1)
      DIMENSION RECORD(300)
      INTEGER DEST,SOURCE,CONTRL
      DATA CS/O/0023002300000000000"/
      DATA IACK/O"0006000600000000000"/
      DATA INACK/O"0025002500000000000"/
      DATA INPUT,DEST,SOURCE/L"INPUT".L"DEST",L"SOURCE"/
      DATA CONTRL/L"CONTRL"/
C INPUT=REMOTE AUXILIARY DATA INPUT FILE (MODE 0)
C DEST= DISK DESTINATION FILE FOR PACKED DATA (5 BYTES/WORD)
C SOURCE=BINARY INPUT DATA PORT (MODE2)
C CNTRL=OUTPUT CONTRL PORT (MODE 1)
C OUTPUT=FTN MESSAGE FILE
      CALL CONNEC(SOURCE,2)
      CALL CONNEC(INPUT,0)
      CALL CONNEC(CONTRL,1)
C SIGNAL READY TO ACCEPT DATA
C GET ARRAY SIZE
```

```
      NWORD=52
C USE NWORD=N/256+1 FOR NON-BLOCK MODE OF TRANSMISSION
      WRITE(1,12) N,M
 12   FORMAT(1X,I10)
      WRITE(DEST) N,M
C= NO. OF WORDS PACKED IN EACH RECORD OF THE DEST FILE
      INT=N/256
C USE INT=1 FOR NON-BLOCK MODE OF TRANSMISSION
      WRITE(CONTRL,1) CS
      DO 200 II=1,M
      DO 100 I=1,INT
 110  READ(SOURCE,3) (RECORD(J),J=1,NWORD)
      WRITE(1,10) (RECORD(J),J=1,NWORD)
 10   FORMAT(4(1X,O20))
      READ(INPUT,4) ICHECK
      ICHAR=INACK
      IF (ICHECK .EQ. JCHECK9256,NWORD,RECORD,ICHECK)) ICHAR=IACK
C CHANGE 256 TO N FOR NON-BLOCK MODE OF TRANSMISSION
      IF (.NOT. .CHK.) ICHAR=IACK
      WRITE(CONTRL,1) ICHAR
      WRITE(CONTRL,1) ICHAR
      IF (ICHAR .Q. INACK) GOTO 110
      WRITE(DEST) (RECORD(J),J=1,NWORD)
 100  CONTINUE
      IF (II .EQ. M) CALL PMDDUMP
 200  CONTINUE
 1    FORMAT(A10)
 2    FORMAT(I4)
 3    FORMAT(205A10)
 4    FORMAT(I12)
      END
      FUNCTION JCHECK(NN,NWORD,RECORD,ICHECK)
      DIMENSION RECORD(300)
      NBYTE=0
      ISUM=0
      DO 100 I=1,NWORD
```

```
      DO 200 J=1,5
      IBYTE=SHIFT(RECORD(I),12*J) .AND. .NOT.MASK(48)
      ISUM=ISUM+IBYTE
      NBYTE=NBYTE+1
      IF (NBYTE .EQ. NN) GOTO 1000
200   CONTINUE
100   CONTINUE
1000  JCHECK=ISUM
      WRITE(1,11) ICHECK,ISUM
11    FORMAT(iX,020,1X,020)
      RETURN
      END
```

V. USE OF HP-9825 KEYS

1. Prompts

All user entered data in reply to 9825A prompts must be followed by CONT.

2. Intercom

When the terminal program is running, and communications have been established with the remote computer through Intercom (for example), all user commands must be followed by STORE. (STORE is equivalent to CR). The following table is a list of "control keys" which may be useful while in Intercom. In particular, it may be useful while BINRECP is running.

HEX	ASCII	CONTROL	9825 KEY	NOTE
01	SCH	A	STOP	
02	STX	B	REWIND	
07	BELL	G	RESULT	
08	SB	H	INSERT	back-space
09	HT	I	DELETE (line)	
0A	LF	J	EXEC	line feed
0B	VT	K	RECALL	
0C	FF	L	RUN	
0D	CR	M	STORE	carriage return
0E	SO	N	<-	
0F	SI	O	->	
10	DLE	P	↓	
11	DC1	Q	↑	
12	DC2	R	CLEAR	
13	DC3	S	PRT ALL	control-S
14	DC4	T	BACK	
15	NAK	U	FWD	NAK
16	SYN	V	INS/RPL	
17	ETB	W	DELETE (char)	
18	CAN	X	STEP	cancel line
19	EM	Y	CONT	
1B	ESC	[	LIST	
1C	FS	\	FETCH	
1D	GS	]	ERASE	
1E	RS	~	RECORD	
1F	US	^	LOAD	

Note that some control codes are not available at the keyboard. In particular, control-F (ACK) is not available. All in-coming codes having a corresponding display are displayed on the 9825A. See the "Systems Programming" manual, pages 50 - 51, for a complete list of display characters with their corresponding 8-bit code.

APPENDIX A - PL/I Test Program for Data Transmission

```
0000 binsend;
0006      proc options (main);
0006  * revision -- February 4, 1982
0006   author -- Barrett, TB
0006   purpose -- Test "binary" (ASCI-256) communications with the
0006      CDC6600.  The companion CDC receive program is BINREC.*
0006 % replace
0006      cr by 'OD'b4,
0006      ack by ' F',
0006      nak by ' U';
0006      dcl
0006           term entry
0006           bpunch entry (bit(8)),
0006           (rec-data,ch) char (1)
0006           (i,j,k,n,num) fixed(15)
0006           Num-array (7) fixed (15) static initial
0006              (0,50,100,150,200,250,255),
0006           (p,pc) pointer,
0006           checksum fixed(15)
0006           checkhex bit(16) based(pc),
0006           1 hilo based(p),
0006              2 lonum bit(8),
0006              2 hinum bit(8),
0006           (reader,dest) file;
0006      open file(reader) title('$rdr') stream input line size(0);
0022      open file(dest) title('$pun') stream output line size(0);
003E      call term;
0041      rec-data = 'A';
004D      put skip edit ('enter starting num') (a);
006F      do while (rec-data ='^S');
007E        get file (reader) edit (rec-data (a(1));
00A1      end;
00C3      get list (num);
00DB      p=addr (num);
```

```
00E1      pc=addr (checksum);
00E7      do while (num < 256);
00F3        if num  =13 & num  =141 then
0110           do;
0110           rec-data = 'A';
011C           do while (rec-data ^ = ack);
012B              put skip edit ('sending', num)  (a,f,(4));
0158              put file (dest) edit ('^@') (a);
0178              do j=1 to 256;
018A                  call bpunch (lonum);
01A0              end;
01A0              put file (dest) edit ('^@^@^@^@') (a);
01C0              call bpunch (cr);
01C6              checksum = num *256;
01D2              put file (dest) edit ('^@',checkhex) (a,b4(4));
0201              call bpunch (cr);
0207              rec-data=fetch-ack();
0215           end;
0215        end;
0215           num=num+1;
021F   end;
021F   put skip edit ('end of transmission') (a);
0241   call term;
0244 fetch-ack;
0244   proc returns (char(1));
0244   rec-data='A';
0253   do until ( ((rec-data=ack)|(rec-data=nak)));
0277      get file (reader) edit (rec-data) (a(1));
029A      end;
029A   return (rec-data);
02A5 end fetch-ack;
02A5 end binsend;

CODE SIZE = 02A8
DATA AREA = 00AF
FREE SYMS = 4089
END COMPILATION
```

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, solid state sciences, electromagnetics and electronic reliability, maintainability and compatibility.

Printed by
United States Air Force
Hanscom AFB, Mass. 01731

END

FILMED

3-85

DTIC