THESIS

LIBRARY
Michigan State
University

This is to certify that the

thesis entitled

THE DESIGN AND APPLICATION OF A
VIDICON RAPID SCANNING SPECTROPHOTOMETER
TO STOPPED-FLOW KINETICS

presented by

Erik Mattis Carlson

has been accepted towards fulfillment
of the requirements for

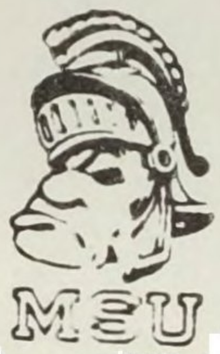
Ph.D. degree in Chemistry



Major professor

Date September 5, 1978

O-7639


MSU

THE DESIGN AND APPLICATION OF A VIDICON RAPID SCANNING SPECTROPHOTOMETER TO STOPPED-FLOW KINETICS

By

Erik Mattis Carlson

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Chemistry

1978

G108993

ABSTRACT

THE DESIGN AND APPLICATION OF A VIDICON RAPID SCANNING SPECTROPHOTOMETER TO STOPPED-FLOW KINETICS

By

Erik Mattis Carlson

A rapid scanning stopped-flow spectrophotometer has been developed. The use of a silicon vidicon array detector enables reactions to be simultaneously monitored over a spectral region of approximately 280 nm. Under computer control, this spectral region can be divided into between 16 and 4096 wavelength channels, which can be scanned at frequencies from 4 kHz to 100 kHz. This allows a spectrum with 100 channels to be acquired in as little as 1 msec.

The slow response time of the silicon vidicon detector was found to limit its usefulness in the monitoring of fast reactions with large changes in absorbance. Further experiments showed that the response time of the detector can be improved by decreasing the frequency at which the wavelength channels are scanned, increasing the number of channels per spectrum and/or adjusting experimental conditions such that the signal level from the detector is about 50% of saturation.

The instrument is controlled by a multi-microcomputer system. One microcomputer controls the vidicon spectrophotometer. The complete automation of the stopped-flow mixing system is handled by a second microcomputer. Another microcomputer handles interactions with the operator, mass data storage and communications with a laboratory minicomputer. A fourth microcomputer coordinates the

activities of the other three. The use of a multi-microcomputer system simplifies the software requirements for the individual microcomputers, improves system modularity and allows a number of operations to proceed simultaneously.

A new computer language called CONVERS-M was developed to operate in the multi-microcomputer system. This language allows new commands, which are geared directly to the instrument, to be easily created and used. Commands have been created which dramatically simplify the coordinations of activities among the microcomputers. The use of CONVERS-M has greatly improved the versatility of the instrument by allowing new modes of operation to be rapidly developed and easily modified.

The instrument has been applied to an investigation of unusual reaction rates encountered in a kinetic method for determining the total protein content of human blood serum using the biuret reaction. The unusual behavior was found to be primarily due to a shift in the wavelength of maximum absorbance in the early part of the reaction. Further studies indicated that the cupric tartrate complex in the reagent was absorbing over the same wavelength region and causing the shift.

To my devoted wife Stephanie and Mom and Dad

ACKNOWLEDGMENTS

My sincere appreciation goes to Professor Chris Enke for the help and encouragement he has shown me throughout the course of my research at Michigan State University.

I want to especially thank Professor G. E. Leroi for serving as second reader for this thesis and Professor Stan Crouch for many valuable discussions.

I would like to thank Michigan State University and the Office of Naval Research for the financial support they provided.

I want to thank the members of Professor Enke's research group for their help and friendship, especially Spyros Hourdakis, Rick Yost and Jim Hornshuh.

My special appreciation goes to my parents for their encouragement and confidence which has helped so much.

To my wife, Stephanie, I offer my highest appreciation. Her love and good humor have been a constant aid. The help she provided in the preparation of this thesis, by typing the initial and final drafts, proofreading and drawing nearly all of the figures, was tremendous.

TABLE OF CONTENTS

	Page
LIST OF TABLES	vii
LIST OF FIGURES	viii
CHAPTER 1 - INTRODUCTION	1
CHAPTER 2 - ROUTINE OPERATION OF THE INSTRUMENT	4
A. Introduction	4
B. An Overview of the Instrument	4
C. Multi-Microcomputer System Software	6
D. Data Analysis Software	16
CHAPTER 3 - A GENERAL PURPOSE MULTI-MICROCOMPUTER SYSTEM FOR RESEARCH INSTRUMENTATION	21
A. Introduction	21
B. The Evolution of the Multi-Microcomputer System	22
C. The Structure of the Multi-Microcomputer System	30
D. The External Communications System	44
CHAPTER 4 - THE DESIGN OF A RAPID SCANNING SPECTROPHOTOMETER FOR STOPPED-FLOW KINETICS	47
A. Introduction	47
B. Rapid Scanning Techniques in Stopped-Flow Kinetics	48
1. Mechanical Scanning Techniques	48
2. Array Detectors	50
C. The Design of the Vidicon Rapid Scanning Spectrophotometer	51
CHAPTER 5 - THE DESIGN AND CONSTRUCTION OF AN AUTOMATED STOPPED-FLOW MIXING SYSTEM	63

Page

A. Introduction 63

B. The Stopped-Flow Mixing System 63

C. Stopping Syringe Velocity/Position Transducer. . . . 68

CHAPTER 6 - CHARACTERIZATION OF THE VIDICON, RAPID SCANNING, STOPPED-FLOW SPECTROPHOTOMETER. 73

A. Introduction 73

B. Characterization of the Vidicon Spectrophotometer. . 73

1. Introduction. 73

2. Vertical Scan Frequency 74

3. Resolution. 80

4. Electron Beam Current 87

5. Charge Integration. 87

6. Lag 91

C. Characteristics of the Stopped-Flow Mixing System. . 101

D. The Monitoring of a Relatively Rapid Reaction. . . . 108

CHAPTER 7 - AN INVESTIGATION OF THE BIURET REACTION 110

A. Background 110

B. Experimental 112

C. Analysis of the Biuret Reaction Spectra. 120

CHAPTER 8 - A USER-DEFINABLE LANGUAGE FOR COMPUTERIZED INSTRUMENTATION. 129

A. The Role of Software in Computerized Instrumentation. 129

B. Programming with CONVERS-M 136

C. The Structure of CONVERS-M 141

APPENDIX A - CONVERS-M USER'S GUIDE 147

A. Introduction 147

Page

B. Special Characters 147

C. Labels 148

D. Parameter Passing. 149

E. Mode Word. 152

F. Using the Floppy Disk Drive. 152

G. Assembly (Compilation of Stored Routines). 154

H. Communication with the PDP 11/40 155

I. Error Codes. 157

APPENDIX B - CONVERS-M BASIC COMMAND LIST 160

A. Coordinator's Basic Command List 160

B. Communicator's Basic Command List. 168

APPENDIX C - CONVERS-M FLOW CHART 171

APPENDIX D - PROGRAM LISTINGS 174

A. Microcomputer Programs 174

B. Minicomputer Programs. 206

REFERENCES. 214

LIST OF TABLES

Table		Page
2-1.	Routine Instrumental Commands	8
2-2.	Instrumental Parameter Codes	11
3-1.	Backplane System Bus Assignments	45
8-1.	Comparison of Common Programming Languages	135
A-1.	Monitor Error Codes	158
A-2.	'PIP' Error Codes	159

LIST OF FIGURES

Figure		Page
2-1.	Block Diagram of a Rapid Scanning Stopped-Flow Spectrophotometer	5
2-2.	Block Diagram of the Vidicon Rapid Scanning Stopped-Flow Spectrophotometer	7
2-3.	Absorption Spectra at Various Times During the Biuret Reaction with Human Serum	18
2-4.	Absorbance (575 nm) Versus Time for the Biuret Reaction with Human Serum.	19
2-5.	Absorbance (839 nm) Versus Time for the Biuret Reaction with Human Serum.	20
3-1.	Distributed Single Minicomputer System. LP = Line Printer; FL = Floppy Disk Drive; DSC = Disk Drive; TRM = Terminal	23
3-2.	Distributed Single Microcomputer System. LP = Line Printer; FL = Floppy Disk Drive; DSC = Disk Drive; TRM = Terminal; MC = Microcomputer .	24
3-3.	Dedicated Bus System	26
3-4.	Time Shared Bus System	26
3-5.	Cross-Bar Matrix Switch System	27
3-6.	Master-Slave System.	31
3-7.	Distributed Multi-Microcomputer System. LP = Line Printer; FL = Floppy Disk Drive; DSC = Disk Drive; TRM = Terminal; MC = Microcomputer; TTY = Teletype	32
3-8.	The Basic Bus Switch System.	34
3-9.	Memory Maps of Typical Slave and Coordinator Microcomputers	35
3-10.	Basic Slave Bus Switch	37

Figure		Page
3-11.	Addition of DMA Mass-Transfer	39
3-12.	Basic Coordinator Bus Switch	40
3-13.	Addition of the Register Bus	42
3-14.	Register Bus Switch	43
4-1.	The Silicon Vidicon Tube	53
4-2.	Raster Scan Pattern	53
4-3.	The Silicon Vidicon Target	56
4-4.	Equivalent Circuit for the Silicon Vidicon Target	57
4-5.	Detector Control System	58
4-6.	Data Aquisition System	61
5-1.	Block Diagram of the Stopped-Flow Mixing System	65
5-2.	Stopped-Flow System Mixer and Observation Cell	67
5-3.	Stopping Syringe Velocity/Position Transducer	69
5-4.	Velocity/Position Transducer Processing Circuitry	71
6-1.	Relative Intensity Versus Frequency	75
6-2.	Relative Intensity Corrected for Varying Exposure Time Versus Frequency	76
6-3.	Relative Vertical Drive Amplitude Versus Frequency	77
6-4.	The Effect of Instantaneous Changes in Vertical Scan Frequency	79
6-5.	Relative Intensity Versus Resolution	81
6-6.	Relative Intensity Versus Channel Separation	83
6-7.	Beam Width Versus Focus Coil Current	84
6-8.	The Effect of Instantaneous Change in Resolution	86
6-9.	Relative Intensity Versus Grid 1 Voltage at Various Frequencies	88

Figure		Page
6-10.	Effect of Rapid Changes in Charge Integration at Various Integration Levels	90
6-11.	Signal Decay After Removal of Illumination	93
6-12.	Relative Lag Versus Relative Intensity	95
6-13.	Relative Lag Versus Resolution (Frequency = 25 kHz).	96
6-14.	Relative Lag Versus Vertical Scan Frequency (Resolution = 255 Channels/Frame).	97
6-15.	Relative Lag Versus Relative Vertical Target Position	99
6-16.	Relative Lag Versus Grid 1 Voltage	100
6-17.	Flow Velocity Output for 5 Separate Runs	103
6-18.	A - Syringe Distance Traveled Versus Time B - Syringe Velocity Versus Time	104
7-1.	Absorption Spectra at Various Times During the Biuret Reaction with Human Serum (2.2 g/L)	115
7-2.	Absorbance Versus Time for the Biuret Reaction with Human Serum. (a) 575 nm; (b) 752 nm.	116
7-3.	Absorbance (839 nm) Versus Time for the Biuret Reaction with Human Serum.	117
7-4.	Absorption Spectra at Various Times During the Biuret Reaction with Bovine Serum Albumin (3.3 g/L).	118
7-5.	Absorbance Versus Time for the Biuret Reaction with Bovine Serum Albumin. (a) 586 nm; (b) 779 nm .	119
7-6.	Absorption Spectra at Various Times During the Biuret Reaction with Bovine Serum Globulin (3.3 g/L)	120
7-7.	Absorbance Versus Time for the Biuret Reaction with Bovine Serum Globulin. (a) 578 nm; (b) 775 nm.	122
7-8.	Absorption Spectrum of the Copper-Protein Complex from the Biuret Reaction with Human Serum After About 25 Minutes	123
7-9.	Absorption Spectrum of the Biuret Reagent.	124

Figure		Page
7-10.	Absorption Spectrum of the Copper-Protein Complex from the Biuret Reaction with Bovine Serum Albumin After About 25 Minutes	126
8-1.	Programming Levels of Various Computer Languages . .	134
8-2.	The Structure of a Command Definition.	142
8-3.	CONVERS-M Library Searching Procedure.	143
8-4.	CONVERS-M Memory Allocation.	145
C-1.	Flow Chart of the CONVERS-M Command Decoding Software.	171
C-2.	Flow Chart of the CONVERS-M Assembler.	172
C-3.	Flow Chart of the Label Decoding Software.	173

CHAPTER 1

INTRODUCTION

Most traditional spectrophotometric detection systems used for stopped-flow kinetics are capable of monitoring reactions at only a single wavelength at a time. This is a severe limitation for many applications. For example, shifts in the wavelength of maximum absorbance due to interfering species and the presence of intermediates in reactions are very difficult to detect with this type of system. Recent developments in techniques for rapid wavelength scanning have helped to overcome this limitation. Rapid scanning systems allow spectra to be acquired rapidly enough to allow reactions to be monitored over an entire spectral region.

This dissertation describes the design and characterization of a multi-microprocessor controlled vidicon rapid scanning stopped-flow spectrophotometer and its application to stopped-flow kinetics. Each chapter was written to be as independent of the other chapters as possible. This allows the reader to study only those chapters in which he has special interest. A better understanding of the total instrument, however, can be obtained by first reading Chapter 2. Chapter 2 contains an overview of the instrument and a description of its normal operation.

A powerful new multi-microcomputer system for instrumental control is described in Chapter 3. The first part of the chapter reviews traditional multi-microcomputer architecture. The rest of the chapter contains a discussion of the design details of the new

multi-microcomputer system. This system simplifies hardware and software design and makes possible the acquisition of multiple types of data, coupled with simultaneous data analysis and instrument control. An understanding of this chapter is not necessary before reading any of the other chapters; however it does help to show how the various parts of the instrument are coupled together.

A new user-definable language which operates in the multi-microcomputer system is described in Chapter 8. This language allows the creation of commands which interact directly with the instrument. Because of this, instrumental control routines can be rapidly created and modified making the total instrument very versatile.

Chapters 4 and 5 provide construction details of the vidicon spectrophotometer and the stopped-flow mixing system respectively. The first part of Chapter 4 also includes a review of all of the rapid scanning techniques which have been applied to stopped-flow kinetics.

Before any instrument can be operated with confidence, its operating characteristics must be well understood. Chapter 6 describes a variety of experiments which were designed to help characterize both the vidicon rapid scanning spectrophotometer and the stopped-flow mixing system. The vidicon detector's slow response time is its most serious limitation to its use in stopped-flow kinetics. Chapter 6 contains a detailed study of the conditions under which the vidicon's response time can be optimized. A final test of the system was its application to the determination of the rate constant for the well known iron/thiocyanate reaction.

The biuret reaction is the basis for the standard clinical chemistry technique for determining total serum protein. The traditional procedure involves letting the biuret reaction go to completion, which takes at least 20 to 30 minutes. Recently, kinetic methods utilizing the biuret reaction have decreased the analysis time to only a few seconds. Although these techniques appear promising, a number of problems still exist. Chapter 7 describes the application of the rapid scanning stopped-flow system to an investigation of these problems.

CHAPTER 2

ROUTINE OPERATION OF THE INSTRUMENT

A. Introduction

The purpose of this chapter is to introduce the reader to the routine operation of the instrument. This is done by first presenting an overview of the basic parts of the instrument and describing the function of each part. The next section describes the instrument's basic operating software. The data evaluation routines for use on our minicomputer are presented in the last section.

B. An Overview of the Instrument

In traditional stopped-flow spectrophotometers the reaction under investigation is monitored by recording the change in absorbance with time of either a reactant, product or intermediate in the reaction. Since only a single wavelength can be monitored at a time, this process must be repeated for each wavelength at which the reaction is to be studied. The use of a rapid scanning spectrophotometer adds another dimension to the spectrophotometric study of stopped-flow reactions by enabling the reaction to be monitored over an entire spectral region. A block diagram of a typical rapid scanning stopped-flow instrument is shown in Figure 2-1. The upper half of the figure shows the stopped-flow mixing system. This is simply a solution handling system for rapidly mixing two solutions (here labeled sample and reagent) and delivering them to an observation cell. To monitor any subsequent reaction, light is passed

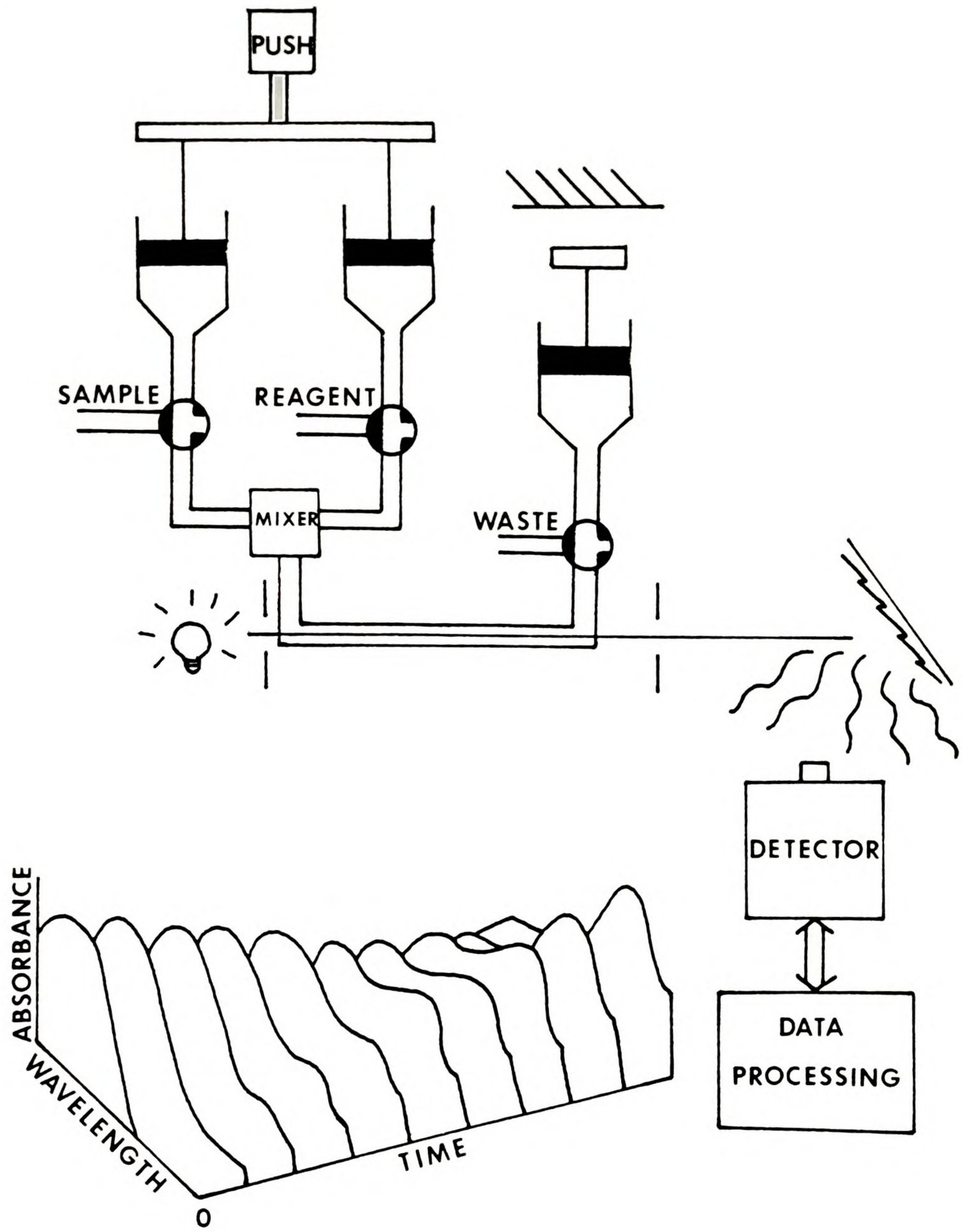


Figure 2-1. Block Diagram of a Rapid Scanning Stopped-Flow Spectrophotometer.

through the observation cell, dispersed by a monochromator and focused onto the rapid scanning detection system. The result is a three-dimensional portrait of the reaction, with time, absorbance and wavelength forming the three dimensions.

The instrument described in this dissertation is controlled by four microcomputers. A block diagram of this system is shown in Figure 2-2. Three of the microcomputers are used for controlling the three major sections of the instrument, while the fourth microcomputer coordinates the activities of the other three. Microcomputer number 1 of Figure 2-2 controls the instrument's communication with the outside world (see Chapter 3). This includes interaction with the operator and a floppy disk mass storage device, and communication with our PDP 11/40 minicomputer. Microcomputer number 2 controls the vidicon rapid scanning spectrophotometer (see Chapter 4). The vidicon spectrophotometer is a very sophisticated television camera which allows spectra to be acquired very rapidly. The operation of the stopped-flow mixing system is controlled by microcomputer number 3 (see Chapter 5). This microcomputer not only handles the entire mixing sequence, it also determines when the solutions have completed mixing. This is very important, for it is at that time that the rapid scanning system must begin acquiring data.

C. Multi-Microcomputer System Software

A listing of all of the routines mentioned in this chapter can be found in Appendix D. The most important instrumental routines are summarized in Table 2-1. This table lists the routine's name,

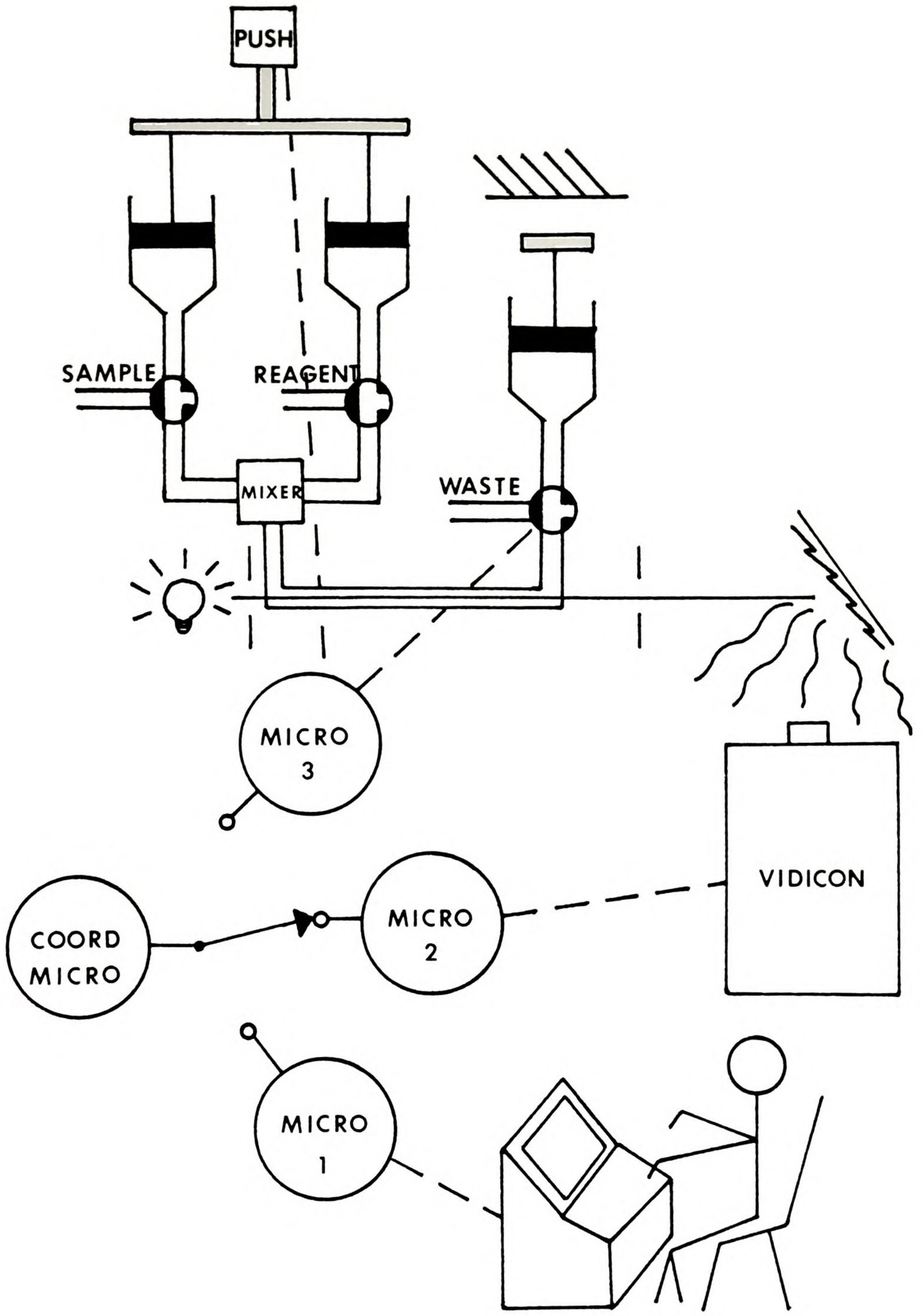


Figure 2-2. Block Diagram of the Vidicon Rapid Scanning Stopped-Flow Spectrophotometer.

Table 2-1. Routine Instrumental Commands.

Command	Parameters	Description
BACK	DS1,DS2	Subtracts dataset DS2 from DS1 and stores result in DS1.
CLEAR	DS	Clears dataset DS.
CLR8K	-	Clears vidicon data memory.
CUT	CS,N	Takes the data at channel N of each vidicon dataset and stores in cut-set CS.
DMEM	N,O	Creates the maximum possible number of datasets of length N in the vidicon data memory.
FLUSH	N	Flushes the stopped-flow system N times.
LOC	X	Locates the position of X on the X,Y plot by drawing a line at the top of the graph.
MAX	DS	Finds the maximum and minimum of dataset DS.
MCUT	N	Creates N new cut-sets.
MINIT	-	Removes all datasets.
MPLOT	DS	Plots dataset DS without erasing previous plots.
OPEN	-	Opens the shutter.
PLOT	DS	Plots dataset DS after erasing previous plots.

Table 2-1. (cont'd.).

POUT	-	Outputs the current values of the instrumental parameters.
SCAN	-	Acquires scans from the vidicon as specified by the scan parameters.
SET	-	Allows instrumental parameters to be altered.
SFSCAN	N,TP,SK	Flushes the mixing system N times; begins data acquisition after trigger point TP; acquires 1 spectrum after skipping SK spectra.
SHUT	-	Closes the shutter.
STORE	DS,FN	Stores dataset DS on floppy disk and names the file FN.
TOUT	DS	Outputs dataset DS to the terminal in decimal form.
TSTORE	FN	Stores the entire array of spectra on floppy disk and names the file FN.
VINIT	-	Initializes the vidicon scanning circuitry.
VEL	CS	Stores a syringe velocity profile in cut-set CS.

any input parameters and the function of each routine. All of this software is written in the new computer language CONVERS-M, which is described in Chapter 8. The normal operation of the instrumental software is presented in this section by describing the procedure involved in a typical stopped-flow experiment.

Before the instrument can be used for an experiment, the desired scan parameters for the vidicon spectrophotometer must be established. The input and output of these scan parameters, as well as a number of other instrumental parameters, are controlled by a set of parameter handling routines. Each parameter is identified by a 2 to 4 letter code. The important instrumental parameters, their identification codes, and their default values are listed in Table 2-2. When the operator wishes to examine the current value of the parameters, he simple types POUT (Parameter OUTput) at the terminal. This causes the terminal display to be cleared and the identification code for each parameter along with its current value to be displayed. The operator can alter any of the parameters by using the SET command. Upon typing this command, the routine outputs an asterisk (*) to the terminal. The operator then types the identification code for the parameter he wishes to alter along with its new value. After this parameter has been entered the routine outputs another asterisk indicating it is ready to input another parameter. Parameter input is completed by typing END. For example, if the operator wished to change the number of scans which were averaged together (AVGS) to 4 and the number of integration frames per readout frame (INT) to 2, he would type:

Table 2-2. Instrumental Parameter Codes.

Code	Default Value	Description
AVGA	1	Number of arrays of spectra to average together.
AVGS	1	Number of spectra to average together.
BEAM	100(-40V)	Grid #1 voltage (beam current).
CPTS	0	Number of words in each cut-set.
DELY	113	Electronic delay and integration mask.
DPTS	0	Number of words in each vidicon dataset.
FREQ	57(25kHz)	Vertical scan frequency mask.
GPTS	0	Number of graph points.
INT	0	Number of scans to integrate.
MAX	0	Maximum value of a dataset or maximum of the plot.
MIN	0	Minimum value of a dataset or minimum of the plot.
PTS	40(102 channels)	Horizontal scan increment. (Determines the number of channel per scan).
SCNS	1	Number of spectra to acquire.
#C	0	Number of cut-sets.
#D	0	Number of datasets.

```
SET
* AVGS     4
* INT      2
* END
```

Many of the data handling routines of the instrument operate on an entire set of data at a time. The size and number of these sets of data are variable and must be initialized by the operator. The software recognizes two distinct types of data. The most common type of data is the data generated by the vidicon spectrophotometer. Sets of this type of data are simply referred to as "datasets". Sets of the other type of data are referred to as "cut-sets". This name comes from the type of data which is normally stored in these sets, which involves "cuts" of intensity with time through the stopped-flow spectra at a single wavelength. All vidicon datasets must reside within the 8 k-byte vidicon data memory area. The memory allocation for the datasets is established by the DMEM command. If the following command sequence is typed at the terminal:

```
DMEM  102  0
```

the DMEM routine will divide the data memory into datasets, each of which contains 102 words of storage (2 bytes/word). The largest possible number of datasets will be created (40 in this case) and each will be initialized to contain zeros. The creation of cut-sets is controlled by the command MCUT. The creation of two cut-sets would be accomplished by typing:

MCUT 2

The length of each cut-set in words would be equal to the current number of datasets. When a reference to a specific set of data is required by a data handling routine, the reference must consist of two parts. The first part of the reference is a single letter which specifies the type of data involved (C for cut-sets and D for vidicon datasets). The second part of the reference specifies the decimal number of the set of data (from 1 to a number equal to the total number of sets). For example, if the operator wished to plot the data contained in vidicon dataset number 23, he would type:

PLOT D 23

Normally it is desired that the reaction data from the instrument be in the form of the absorbance versus time over a specified spectral region. Since the only output from the vidicon spectrophotometer is in the form of the signal intensity versus wavelength, it is necessary to calculate the absorbance values. This is done by a data evaluation routine on our laboratory minicomputer. The absorbance (A) at wavelength I is derived from the signal intensity (S) of a reaction spectrum by the equation:

$$A_I = - \log \left(\frac{S_I - D_I}{R_I - D_I} \right),$$

where R_I is the signal intensity with a reference solution and D_I is the signal obtained when no light is falling on the detector (dark

current signal). This requires that three different types of data be acquired. First, the observation cell must be filled with the selected reference (100% transmittance) solution. In order to assure that old solutions have been removed completely from the observation cell and drive syringes of the stopped-flow mixing system, the system must be flushed a number of times with the reference solution. This is accomplished by the FLUSH command. This command acquires an octal number from the operator which designates the number of times the stopped-flow mixing system is to be flushed. After the flushing is completed, the operator acquires and stores the reference spectrum by typing the following set of commands:

```
OPEN
CLR8K
SCAN
STORE  D  1
* FILE1
```

The OPEN command causes the shutter to the light source to open. The next instruction (CLR8K) clears the dataset memory. The spectrum is obtained by the SCAN command. The spectrum, which is placed in dataset D 1 by the SCAN command, is then stored on a floppy disk mass storage device by the command STORE. This command sends an asterisk to the terminal and the operator then specifies the name for the file which is to be created. The dark spectrum is obtained in a similar manner except that the command SHUT (which closes the shutter) is used in place of the OPEN command.

The next step in the procedure is to fill the syringes with the two reactant solutions. This is accomplished by again flushing the system by using the FLUSH command. The following command sequence is then used to acquire the intensity versus time profile for the reaction:

```
OPEN
CLR8K
SFSCAN    1,47,5
```

The first two commands open the shutter and clear the dataset memory. The next command (SFSCAN) requests three parameters from the operator. The first parameter indicates the number of times the mixing system is to be flushed before the actual data run is taken. The next parameter is the trigger point. The trigger point is used to specify the time at which the reactant solutions are mixed and the spectra can begin to be taken. Under normal operation the trigger point is always set to 47. (For more information on the trigger point and its selection, see Chapters 5 and 6.) The final parameter specifies the number of spectra which are skipped between the acquisition and storage of succeeding spectra. The array of spectra which is acquired by the SFSCAN command can be stored on floppy disk by using the command TSTORE. This command is similar to the STORE command described earlier, except that TSTORE outputs the entire array of datasets instead of just a single dataset. TSTORE also sends an asterisk to the terminal, indicating that the operator should enter the name of the file that is to be created.

The reference and dark spectra should be taken immediately before the reaction spectra. This helps to eliminate any errors caused by slow drifts in the light source, dark current, etc. in the vidicon spectrophotometer.

The more powerful data analysis routines are all run on our PDP 11/40 minicomputer. The process by which the data files are transferred from the instrument to the minicomputer is described in Section H of Appendix A. The operation of some of the data evaluation routines is the subject of the next section.

D. Data Analysis Software

A great many routines were created on our PDP 11/40 minicomputer for the analysis of various types of data from the vidicon stopped-flow system. All but two of these were special purpose routines which were used for only a short time (primarily during the characterization of the instrument). The two routines which are described in this section are general purpose routines which were used for most of the standard data analysis. All operator interactions are handled through the minicomputer's terminal.

When the data files are first transferred from the instrument to the minicomputer, they are stored in binary format to conserve space. Since files in binary format are more difficult to work with, almost all of the data evaluation and plotting routines were designed to work with standard ASCII file formats. The program CONV is used to convert the binary files into ASCII files. The program first requests the file names of the input and output files. The program assumes

that the body of the data in the input file consists of only a single variable (the 'Y' value). The operator then supplies an initial value and an increment for a second variable (the 'X' value). Upon completion of the conversion the output file contains the entire set of data in the form of X and Y data pairs.

The evaluation of the stopped-flow reaction data is performed by the SF program. This program starts by requesting three binary input files. The first 2 files must contain the reference and dark spectra. The third file contains the array of stopped-flow reaction spectra. At this point, the data can be thought of as a three-dimensional array with absorbance, wavelength and time forming the three dimensions. The SF program allows two-dimensional slices to be made through this array. These slices can be in the form of either the absorbance versus time at a specified wavelength of the absorbance versus wavelength at a specified time.

Figure 2-3 shows typical data from the vidicon stopped-flow spectrophotometer. This figure shows the absorbance versus wavelength at various times during the biuret reaction with human serum protein (see Chapter 7). Figures 2-4 and 2-5 show the absorbance versus time for this reaction at wavelengths of 575 nm and 839 nm respectively. The sensitivity of the vidicon system for monitoring reactions is quite apparent in Figure 2-5, where the total absorbance change is less than 0.01.

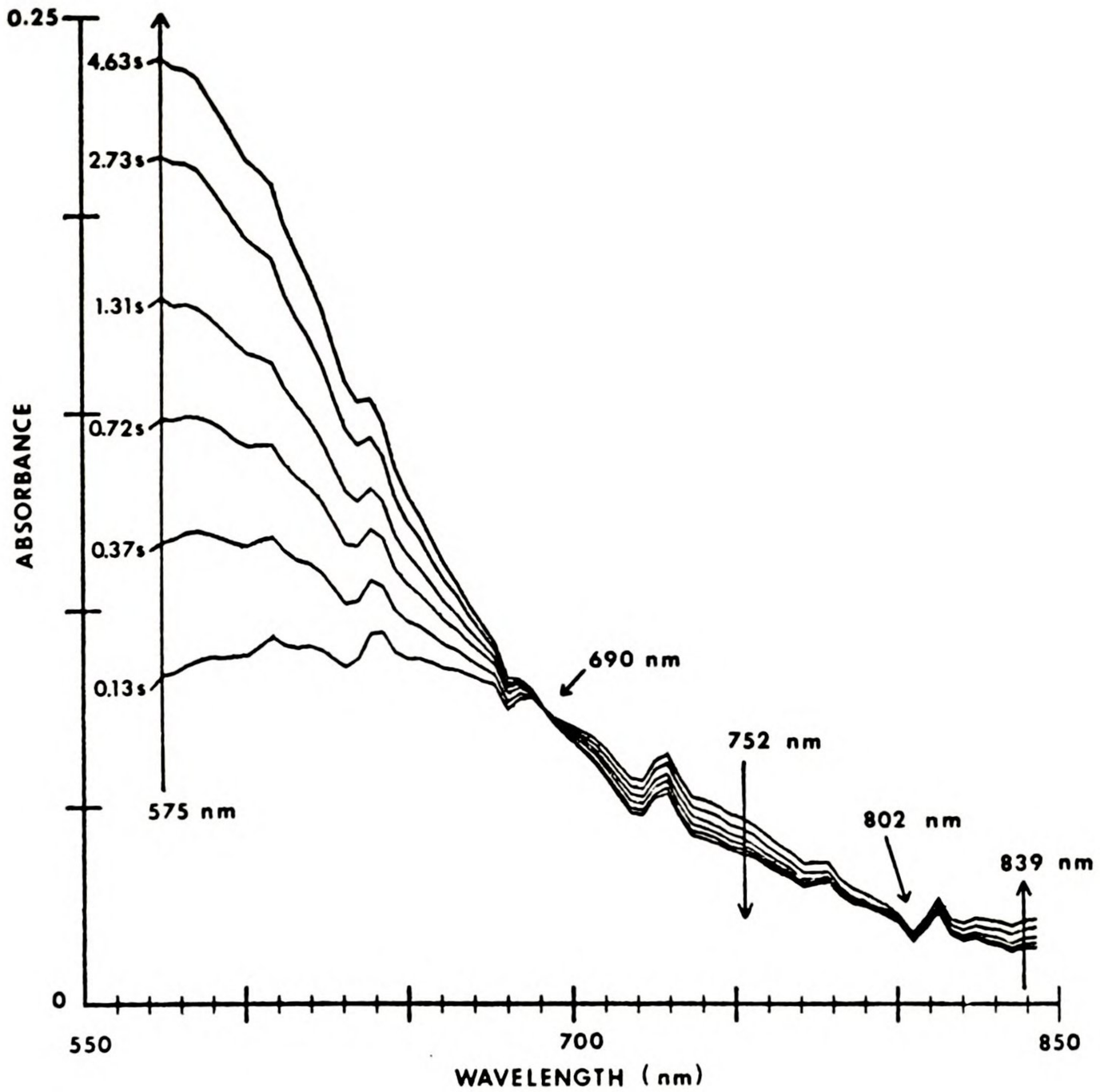


Figure 2-3. Absorption Spectra at Various Times During the Biuret Reaction with Human Serum.

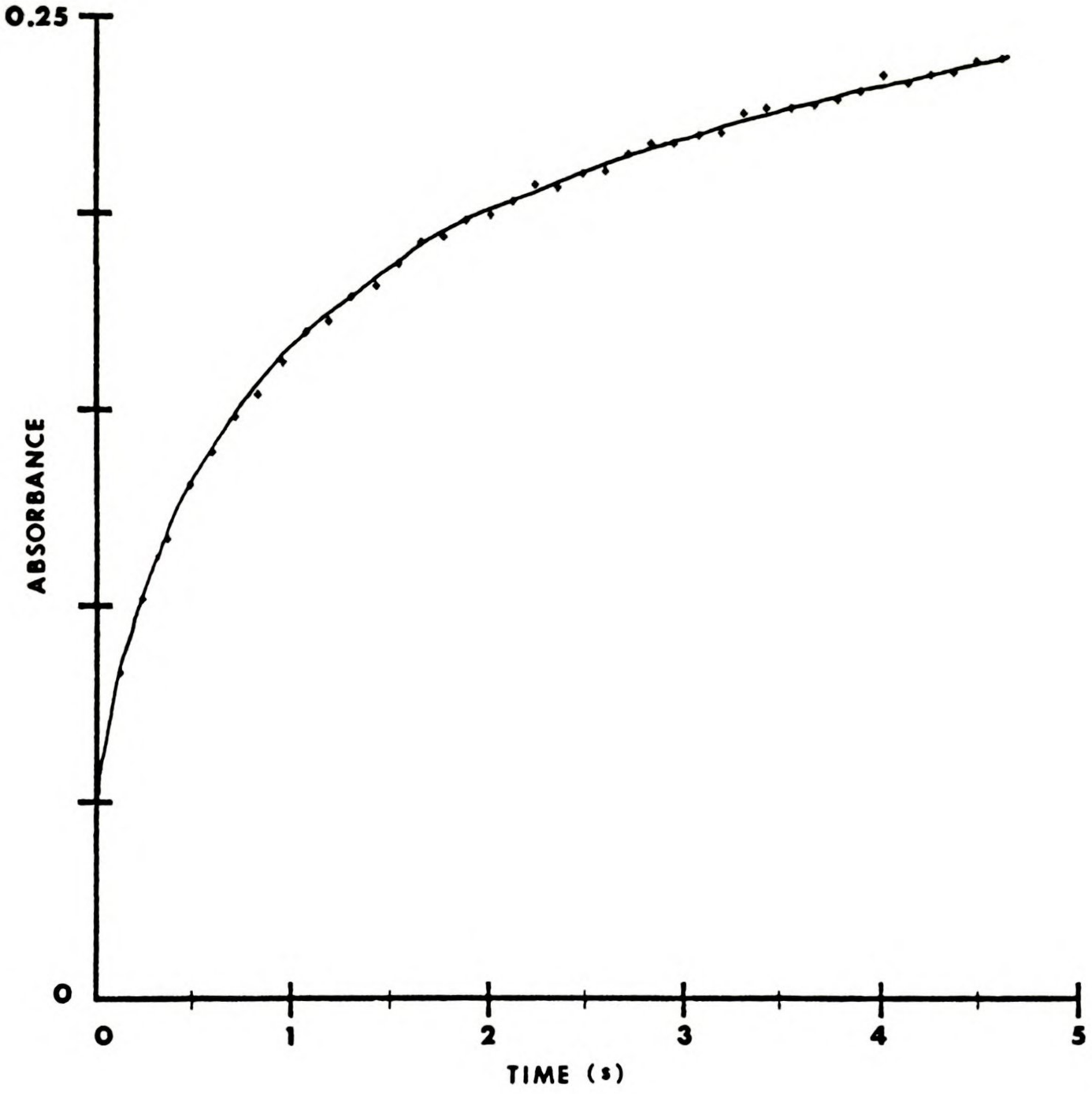


Figure 2-4. Absorbance (575 nm) Versus Time for the Biuret Reaction with Human Serum.

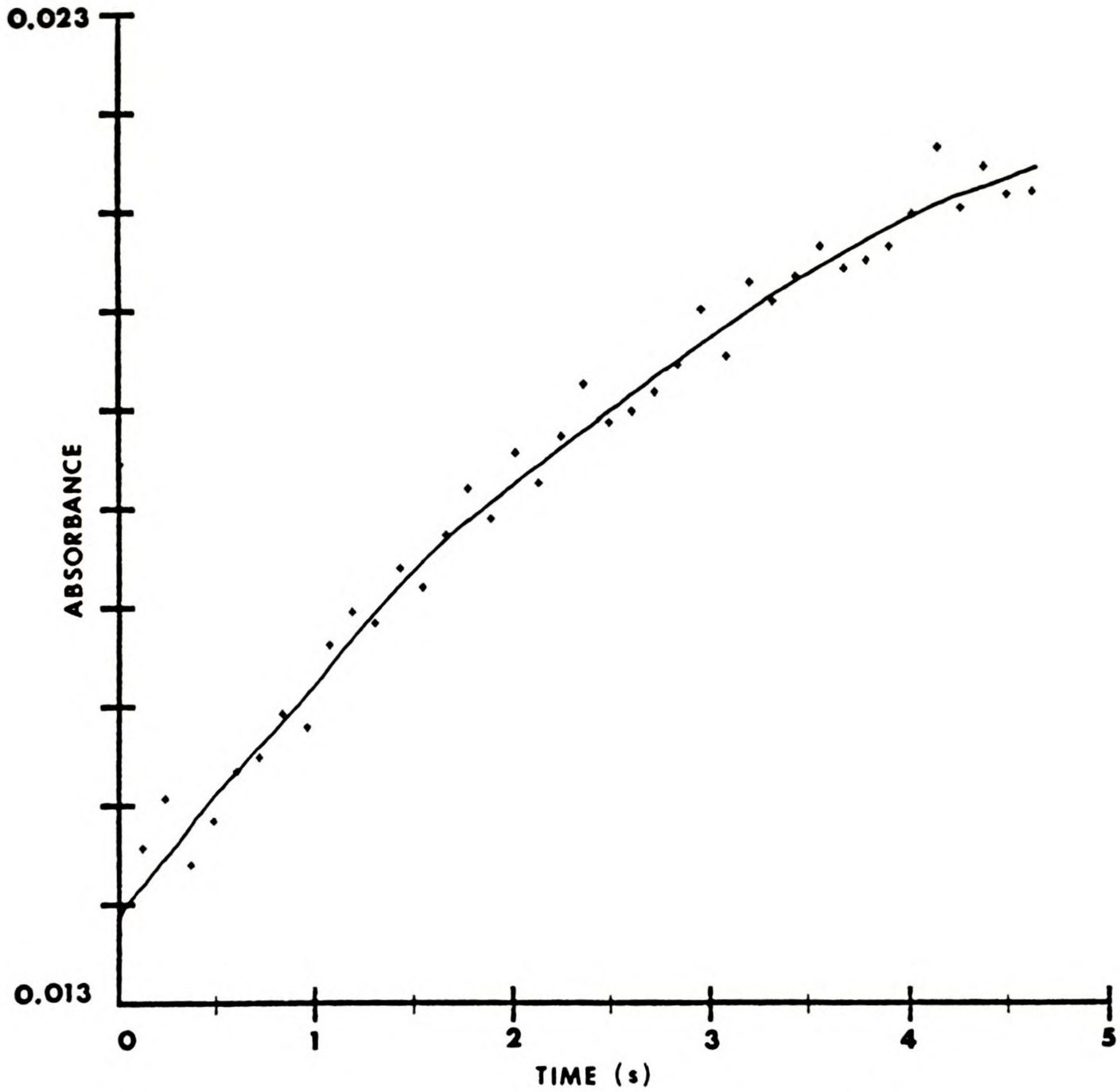


Figure 2-5. Absorbance (839 nm) Versus Time for the Biuret Reaction with Human Serum.

CHAPTER 3

A GENERAL PURPOSE MULTI-MICROCOMPUTER SYSTEM FOR RESEARCH INSTRUMENTATION

A. Introduction

For many years the automation of laboratory instrumentation was accomplished almost exclusively through the use of minicomputers. Although minicomputers were far less expensive than large computer systems, they were still expensive enough that most research laboratories could not afford to use a separate minicomputer for each automated instrument. Normally, a single minicomputer was used to control a number of instruments. This reduced the total cost of the computer system; however, the finite speed of the minicomputer meant that only a small number of these instruments could be operated simultaneously. More recently, the falling costs of minicomputer systems and particularly the introduction of small, inexpensive microprocessors has made it economically feasible to use one or even more computers to control each instrument. This has made it possible for laboratory computer systems to be implemented in a variety of configurations (1-6). The optimum configuration of a laboratory computer system depends on the environment in which it is found. In our research group we have developed a powerful multi-microcomputer system for controlling sophisticated instrumentation in research laboratories. The evolution of our laboratory computer system from a single minicomputer is traced, and details of its current hierarchical multi-microcomputer configuration is presented in the following sections.

B. The Evolution of the Multi-Microcomputer System

Originally, all the computerized instrumentation developed in our research laboratory was controlled by a single minicomputer. In its most advanced form, this single minicomputer system contained a Digital Equipment Corporation PDP 11/40 minicomputer and various peripheral devices. This system is shown in Figure 3-1. This type of laboratory computer system had a number of disadvantages, the major one being that almost all of the instruments interfaced to the system required the full-time attention of the minicomputer. This meant only a single instrument could be operated at any one time. Also, relatively complex interfaces were required to connect the instruments to the minicomputer. In order to overcome these difficulties, we began to search for ways to improve our laboratory computer system.

The first proposed improvement to the laboratory computer system was to place a microcomputer on each instrument. The proposed system is shown in Figure 3-2. The microcomputers would handle the high-speed data acquisition and instrumental control for each instrument. Any high-level data evaluations which were required would still be handled by the minicomputer. With this system, routine operation of the instrument could be performed without requiring intervention from the minicomputer, thereby allowing any number of instruments to be operated simultaneously. Also, due to the microcomputer's simpler bus structure, interfacing to the microcomputer was much simpler than interfacing directly to the minicomputer.

The use of a single microcomputer to control an entire instrument would be perfectly adequate for a large variety of applications.

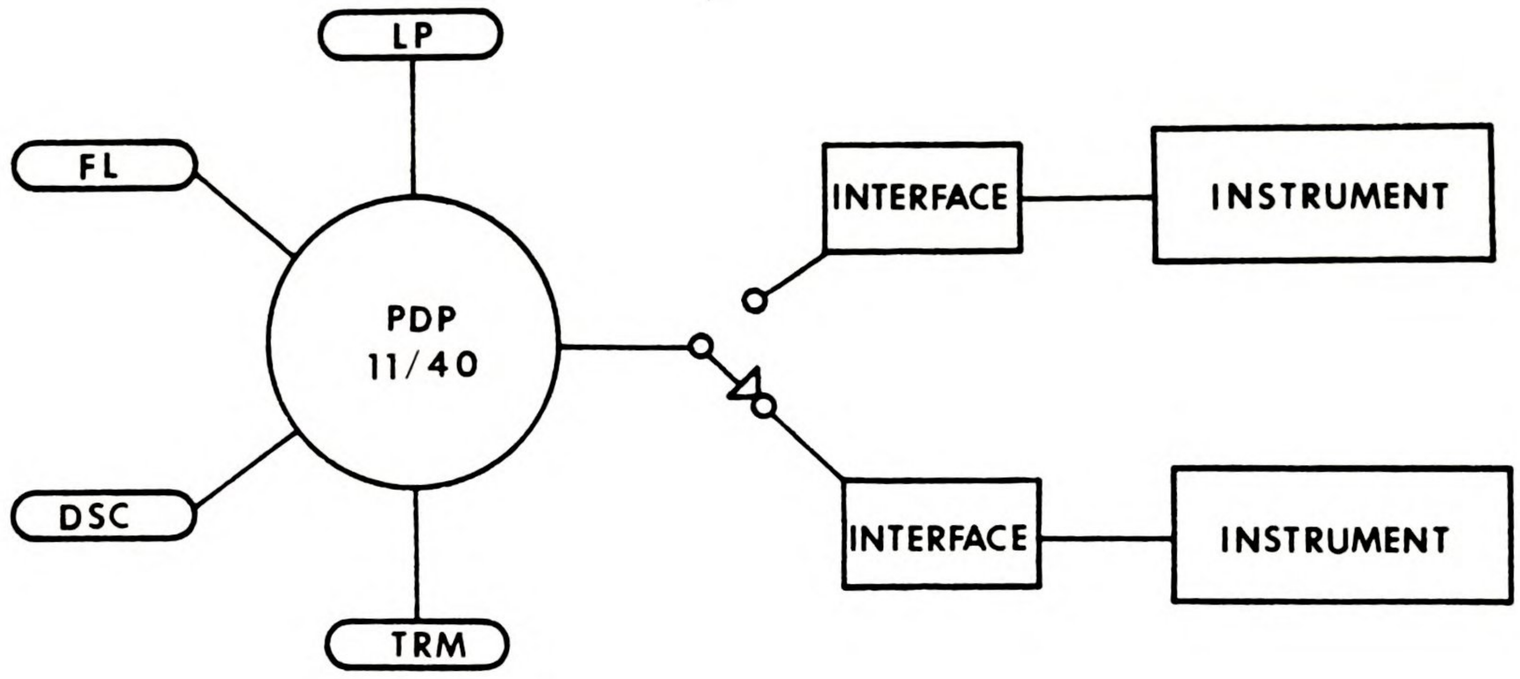


Figure 3-1. Distributed Single Minicomputer System. LP = Line Printer; FL = Floppy Disk Drive; DSC = Disk Drive; TRM = Terminal.

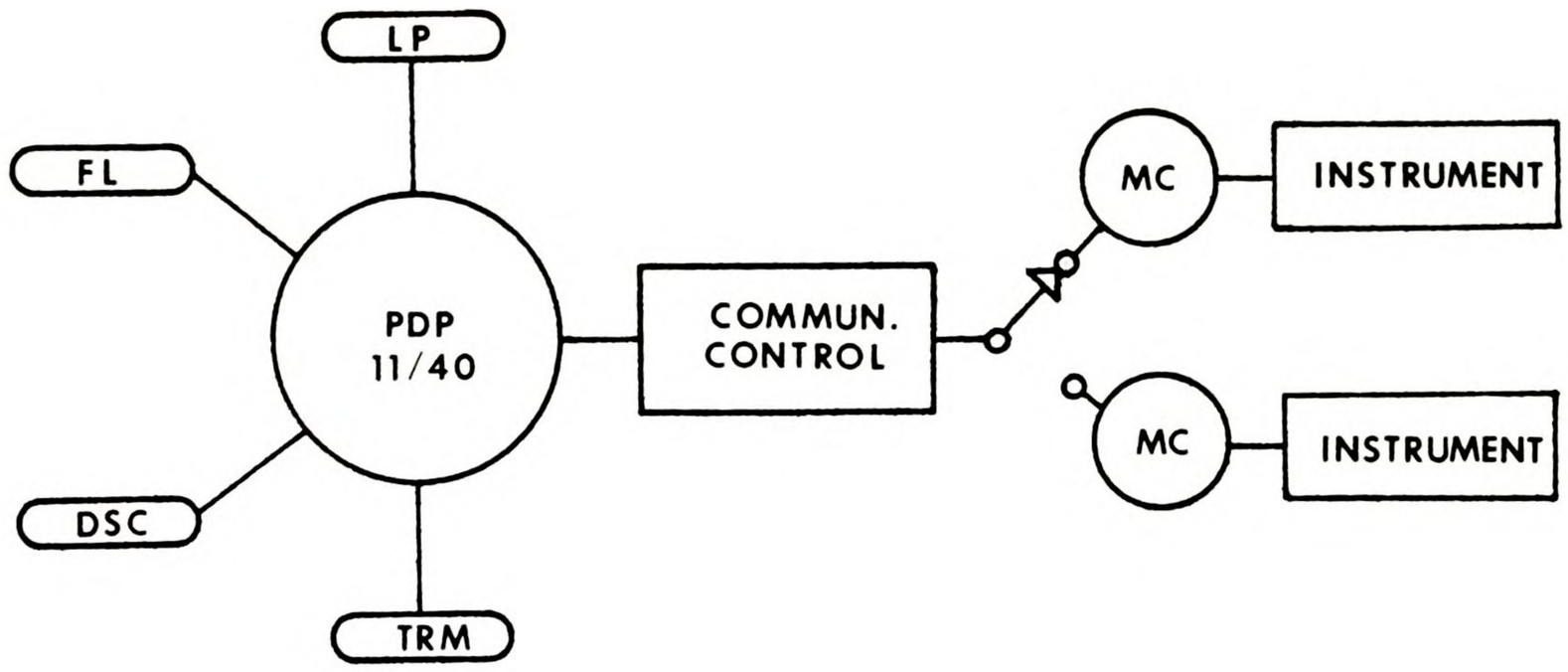


Figure 3-2. Distributed Single Microcomputer System. LP = Line Printer; FL = Floppy Disk Drive; DSC = Disk Drive; TRM = Terminal; MC = Microcomputer.

However, many of the instruments we planned on implementing required the asynchronous acquisition of multiple types of data at extremely high data rates, coupled with simultaneous data evaluation and instrumental control. These instruments were complex enough that it would have been extremely difficult to control them even with a dedicated minicomputer and obviously impossible with a slower microcomputer. The solution to this problem was to use a number of microcomputers. By careful planning in the initial design of an instrument, tasks which must be performed simultaneously can be assigned to different microcomputers. In this way complex problems of simultaneous data acquisition and/or instrumental control can be reduced to a few virtually independent, synchronous tasks. This makes the required hardware and software for any single microcomputer in the system far easier to create and modify.

The overall effectiveness of any multiple microcomputer system is determined to a large extent by the structure of the communication system within and between the various microcomputers in the system. A great many different configurations of multi-microcomputer systems have appeared in the literature (7-16). Almost all of these multi-microcomputer systems can be described in terms of three basic configurations: the dedicated bus system (Figure 3-3), and the time-shared bus system (Figure 3-4), and the cross-bar matrix switch system (Figure 3-5). More complex configurations consist of either expansions of these basic systems (e.g. multiple time-shared buses) or combinations of them.

The dedicated bus configuration (Figure 3-3) is the simplest form

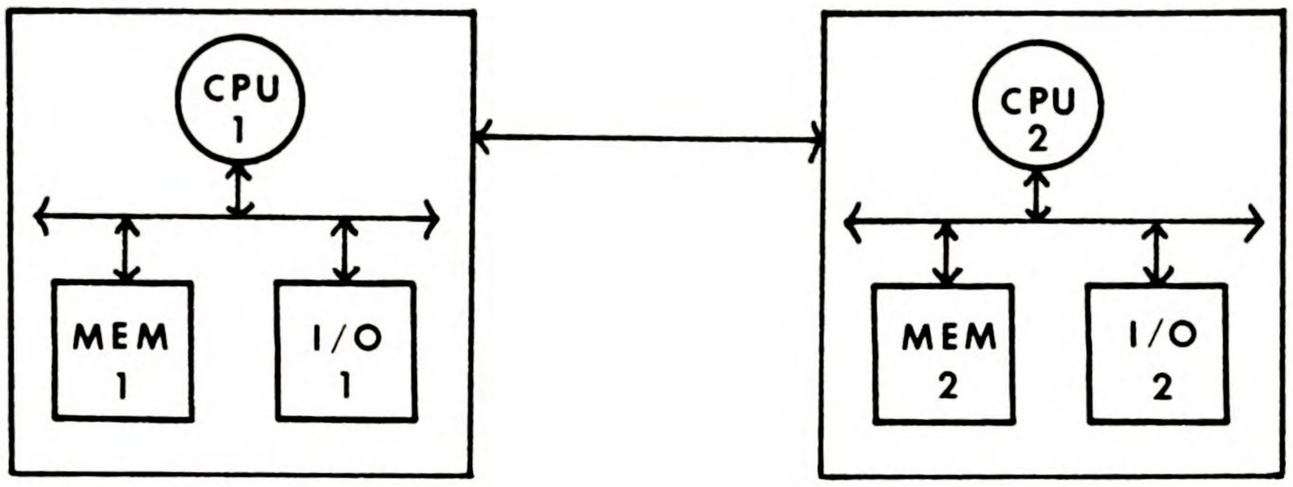


Figure 3-3. Dedicated Bus System.

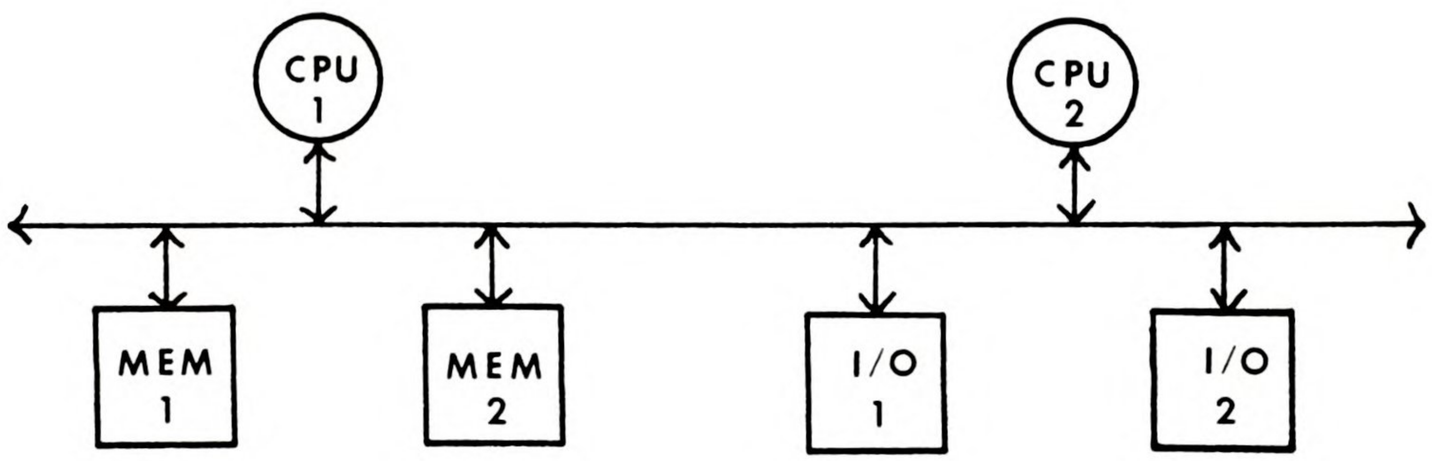


Figure 3-4. Time Shared Bus System.

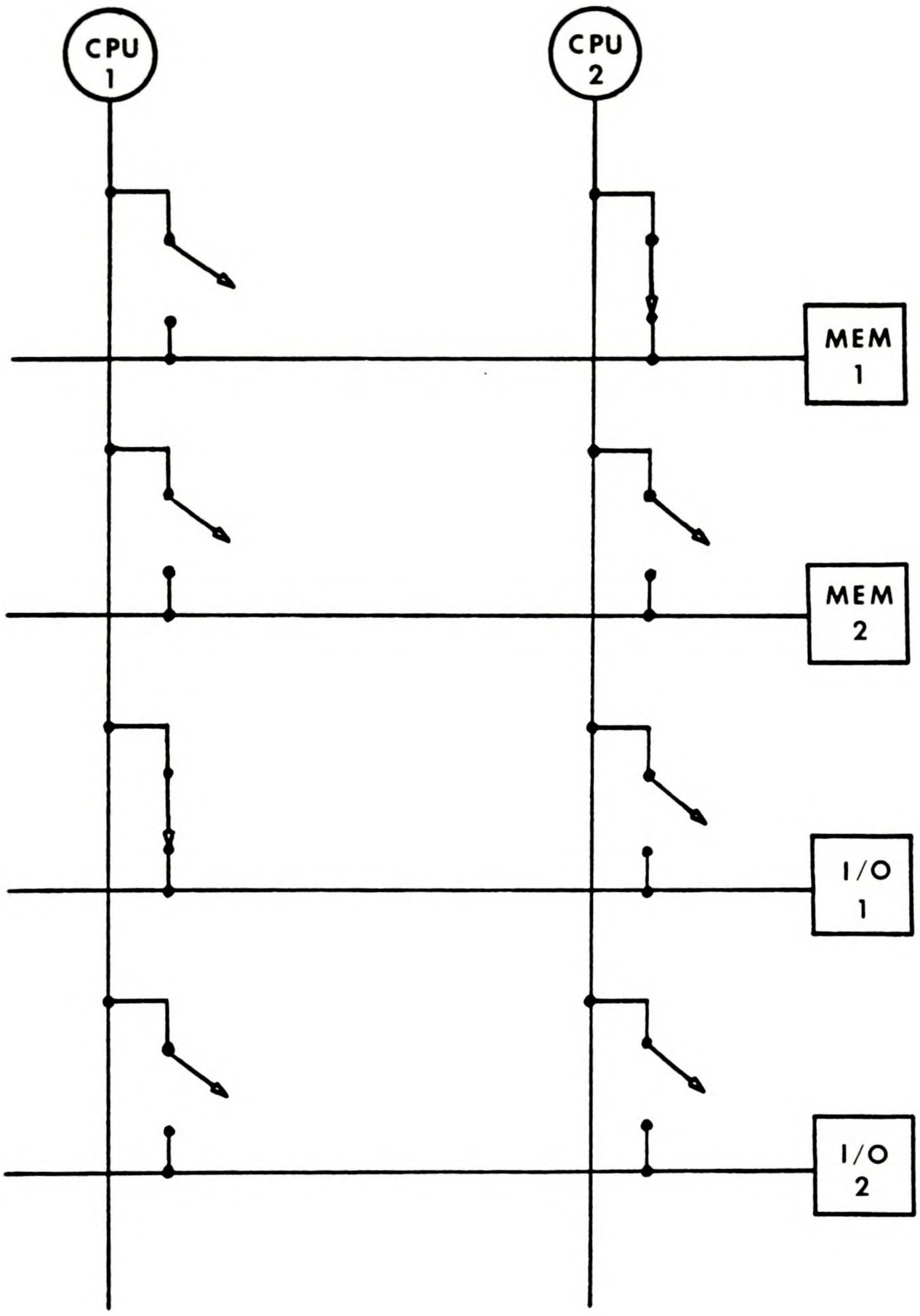


Figure 3-5. Cross-Bar Matrix Switch System.

of multi-microcomputer system. It consists of a number of virtually independent microcomputers interconnected by means of a communications bus. The communications bus can have many different forms. The bus can connect I/O ports on the microcomputers or even be a direct CPU to CPU link. Both serial and parallel data transfer formats can be implemented, depending on the speed and complexity requirements of the system. Since the individual microcomputers are virtually independent, timing conflicts between them are essentially eliminated. This results in overall high system throughput with relatively simple hardware and software required to maintain the communications link. The major disadvantages of this type of system are difficulty in expansion and the inability of the system to reassign tasks dynamically in case of a single microcomputer failure.

In its simplest form, the time-shared bus system, (Figure 3-4) consists of a single bus shared by all the CPU's, memory and I/O ports in the system. A bus arbitrator is required to resolve timing conflicts on the bus. The major disadvantage of this type of system is the reduction in system throughput caused by bus conflicts between CPUs. This situation can be improved (but never eliminated) by removing the program memory and some of the I/O ports from the time-shared bus and dedicating them to individual CPUs. In this way bus conflicts can occur only when the microcomputers are accessing common memory and I/O ports found on the time-shared bus. Communication between microcomputers in this system is normally accomplished by storing status words in "mailbox addresses" located in the common memory. The major advantages of this type of configuration are relatively low interconnection costs, modularity and the ability to reassign tasks dynamically

on the time-shared bus.

The most powerful of the three basic systems is the cross-bar matrix switch configuration (Figure 3-5). With this configuration, any CPU can access any memory or I/O port in the system. Conflicts only occur when one device is accessed simultaneously by more than one CPU. This configuration has very high system throughput, while retaining the ability for dynamic task reassignment. Unfortunately, the extreme complexity and cost of the matrix switch severely limits the size and expandability of this configuration.

In any multi-microcomputer configuration the individual microcomputers can either all be of the same "importance" or one or more can be selected to be more "important" than the others. In a system utilizing dynamic task assignments, this can take the form of a microcomputer which determines the tasks that will be assigned to the rest of the microcomputers. Its most extreme form is a hierarchical master-slave system in which the slave microcomputers are essentially intelligent peripherals of the master microcomputer.

The multi-microcomputer configuration which we finally adopted can best be described as a hierarchical master-slave system with a hybrid dedicated-bus/time-shared bus configuration. The master-slave configuration was selected to simplify the coordination of the various tasks in the system. The high throughput required in the complex instruments we planned to computerize was best accomplished through the use of an essentially dedicated bus configuration. Some characteristics of a shared memory system were also utilized for a simple yet powerful interprocessor communications system. An approximate block diagram of the

system is shown in Figure 3-6. A detailed description of this system is presented in the next section.

The resulting laboratory computer system is shown in Figure 3-7. Communications between the minicomputer and the multi-microcomputer systems is accomplished through the use of a communications controller developed in our laboratory (17). Simple instruments which require only a single microcomputer could also be developed and could communicate with the minicomputer through this communications controller.

C. The Structure of the Multi-Microcomputer System

The multi-microcomputer system which we developed is based on Intel's 8080A microprocessor chip. This microprocessor was selected because of its high speed, extensive and relatively powerful instruction set and the availability of a large variety of standard support chips. Although the system hardware would allow the use of virtually any microprocessor chip, this has never been attempted due to the complexity of the software which would be required in a system having a variety of instruction sets. Each individual microcomputer in the system has a standard 8080A structure with memory-mapped I/O used exclusively. A basic familiarity of this structure and the standard 8080A bus signals is necessary to fully understand the detailed structure of the multi-microcomputer system. This information is readily available in the literature (18).

During the actual execution of an assigned task, each microcomputer is completely independent of the others. The sequencing and coordination of the individual tasks is accomplished by a single

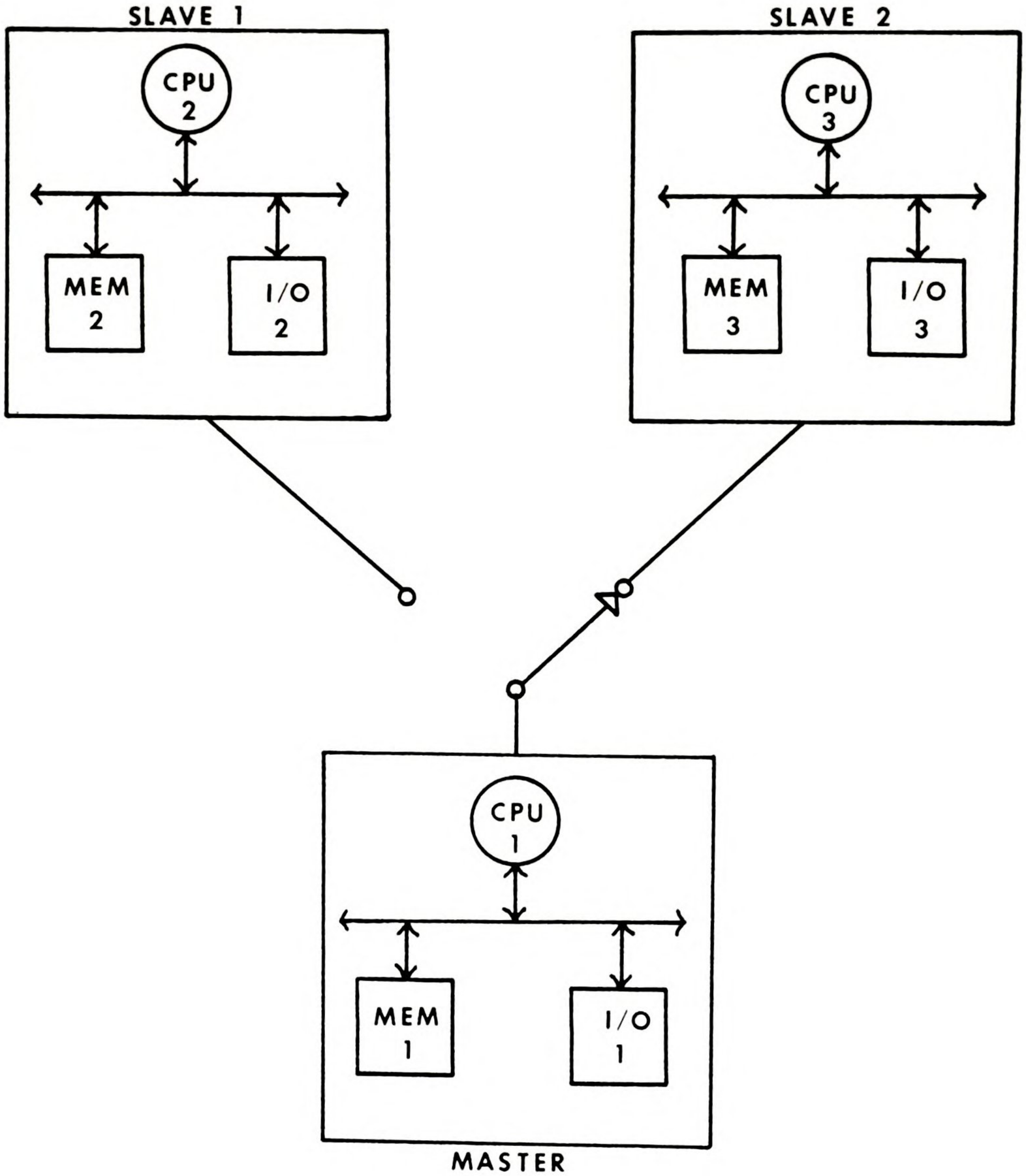


Figure 3-6. Master-Slave System.

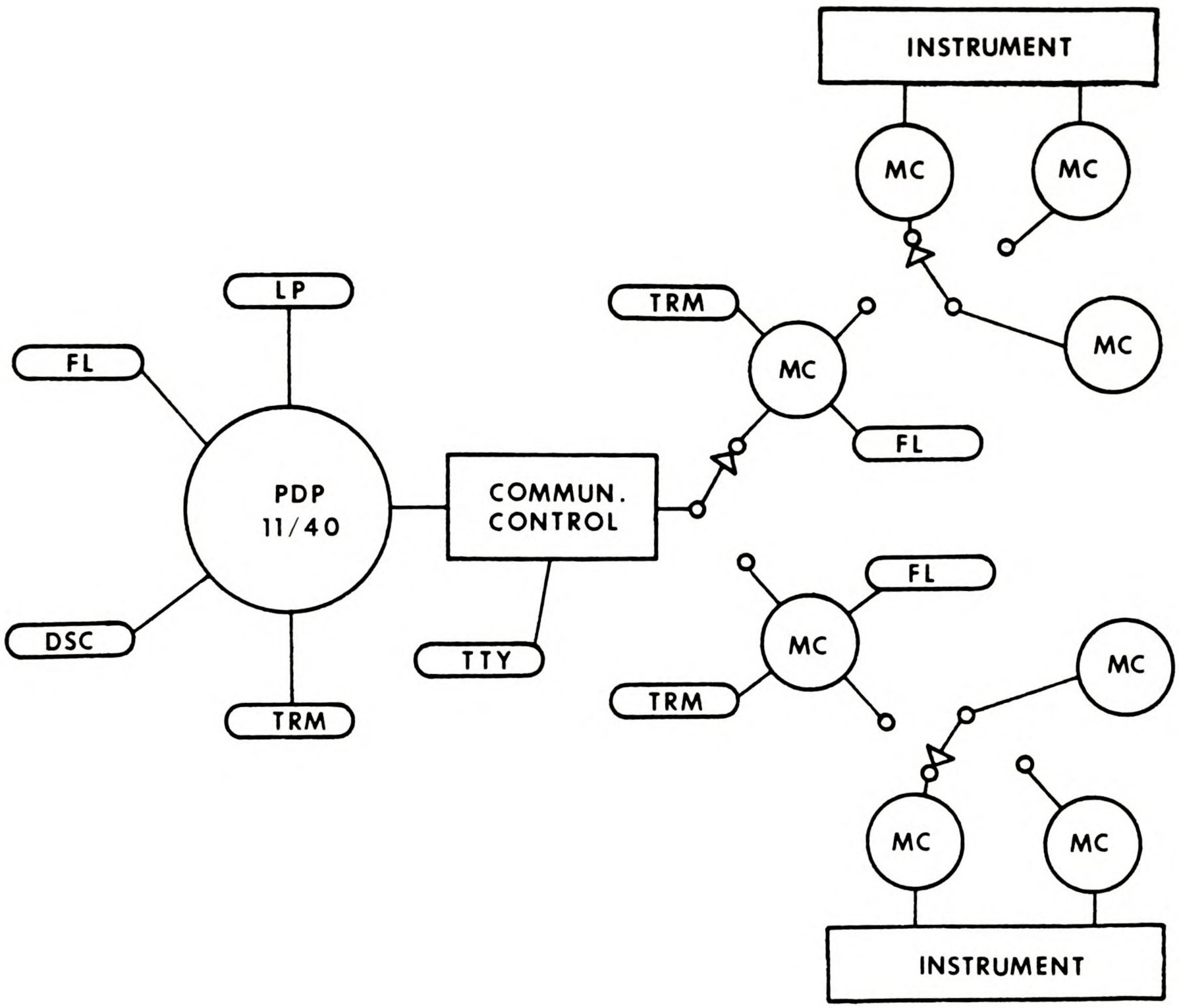


Figure 3-7. Distributed Multi-Microcomputer System. LP = Line Printer; FL = Floppy Disk Drive; DSC = Disk Drive; TRM = Terminal; MC = Microcomputer; TTY = Teletype.

microcomputer which we have appropriately labelled the coordinator microcomputer. The coordinator microcomputer communicates with the slave microcomputers in the system through the use of a dedicated communications bus. When in use, this communications bus is a simple extension of the coordinator's local bus. The communications bus consists of 16 address lines, 8 data lines and 5 control lines ($\overline{\text{MEMW}}$, $\overline{\text{MEMR}}$, DBIN, READY and $\overline{\text{MASTER RESET}}$). A bus switching system is utilized to insure that only one slave is accessed at a time. A basic block diagram of this system is shown in Figure 3-8. The bus switch is controlled by the upper address bits. We have found that in most cases a total of three address bits (allowing a total of 7 slave microcomputers) is optimum. The memory maps of typical slave and coordinator microcomputers are shown in Figure 3-9. With this configuration, an 8K block of each slave is accessible by the coordinator. This address block is always arranged to coincide with the start of the address space in the slave. Normally this block contains the slave's random access memory (RAM) and any I/O ports which might be useful to have accessible to the coordinator. The remaining 56K of the slave's memory contains the rest of the I/O ports and the slave's read only memory (ROM).

The basic communications system also includes a set of three lines connecting the coordinator with the slaves. These lines consist of a slave hold line, task completed flag and reset line for each slave in the system. The hold line is used to allow the coordinator to control when each slave in the system is running. The task completed flag indicates when a slave has completed an assigned task. The coordinator

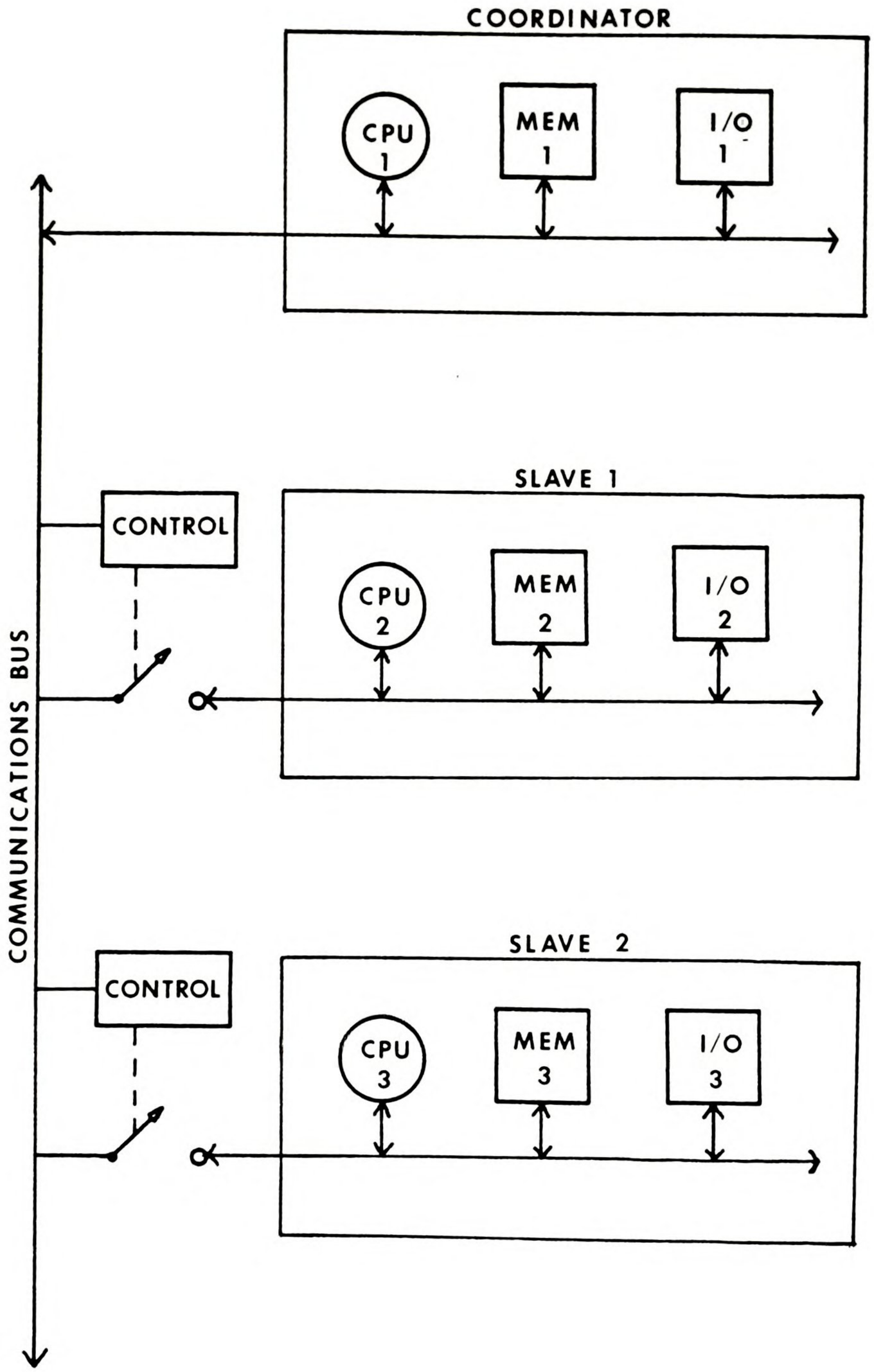


Figure 3-8. Basic Bus Switch System.

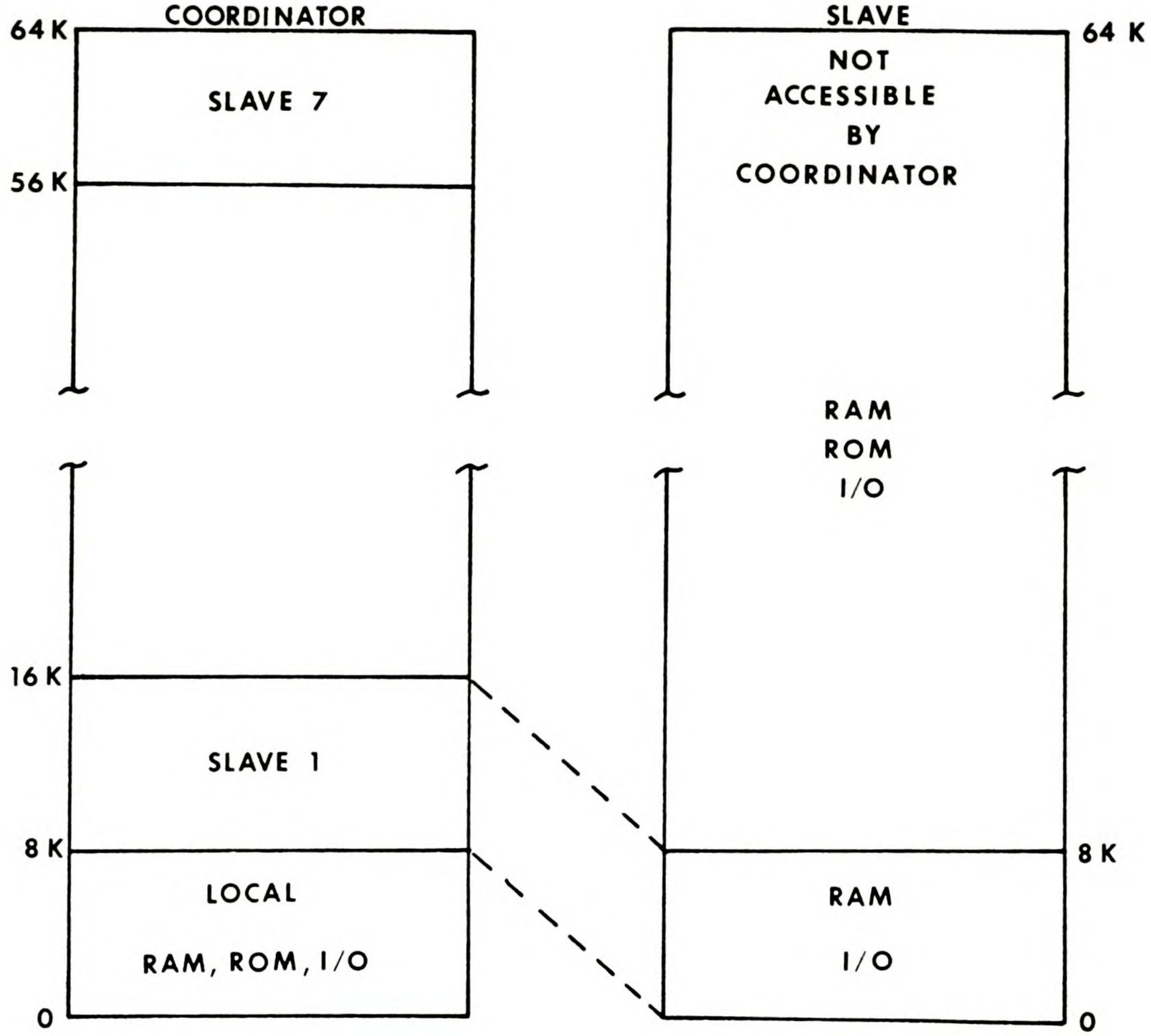


Figure 3-9. Memory Maps of Typical Slave and Coordinator Microcomputers.

can force any slave to start program execution at address zero by using that slave's reset line.

The circuit diagram of a standard slave bus switch is shown in Figure 3-10. When the coordinator attempts to access an address on the slave, the address decoder causes a hold to be sent to the slave CPU. The coordinator's READY line is also pulled down causing the coordinator to go into a wait state. When the slave completes its current instruction cycle the HLDA (HoLD Acknowledge) line goes high, and the slave's system bus is tri-stated. This causes the bus switch to be enabled and the coordinator to be released from its wait state to complete the transfer. The DBIN signal from the coordinator's control bus determines the direction of the data transfer through the bus switch.

The bus switch and the three external control lines enable the coordinator to exercise a high degree of control over the execution of tasks by the slaves. The lowest address block in each slave contains RAM program memory. The first few addresses always contain a subroutine call instruction. When the coordinator requires a task to be performed by a slave, the coordinator uses the bus switch to place the address of the subroutine which accomplishes that task into this subroutine call instruction. The coordinator then resets the slave and releases its hold line. The slave will execute the subroutine call and perform the desired task. On returning from the task subroutine the slave sends the task completed flag to the coordinator. For simplicity, the task completed flag is usually generated by the INTE (INTerrupt Enable) signal in the slave's CPU. This allows a

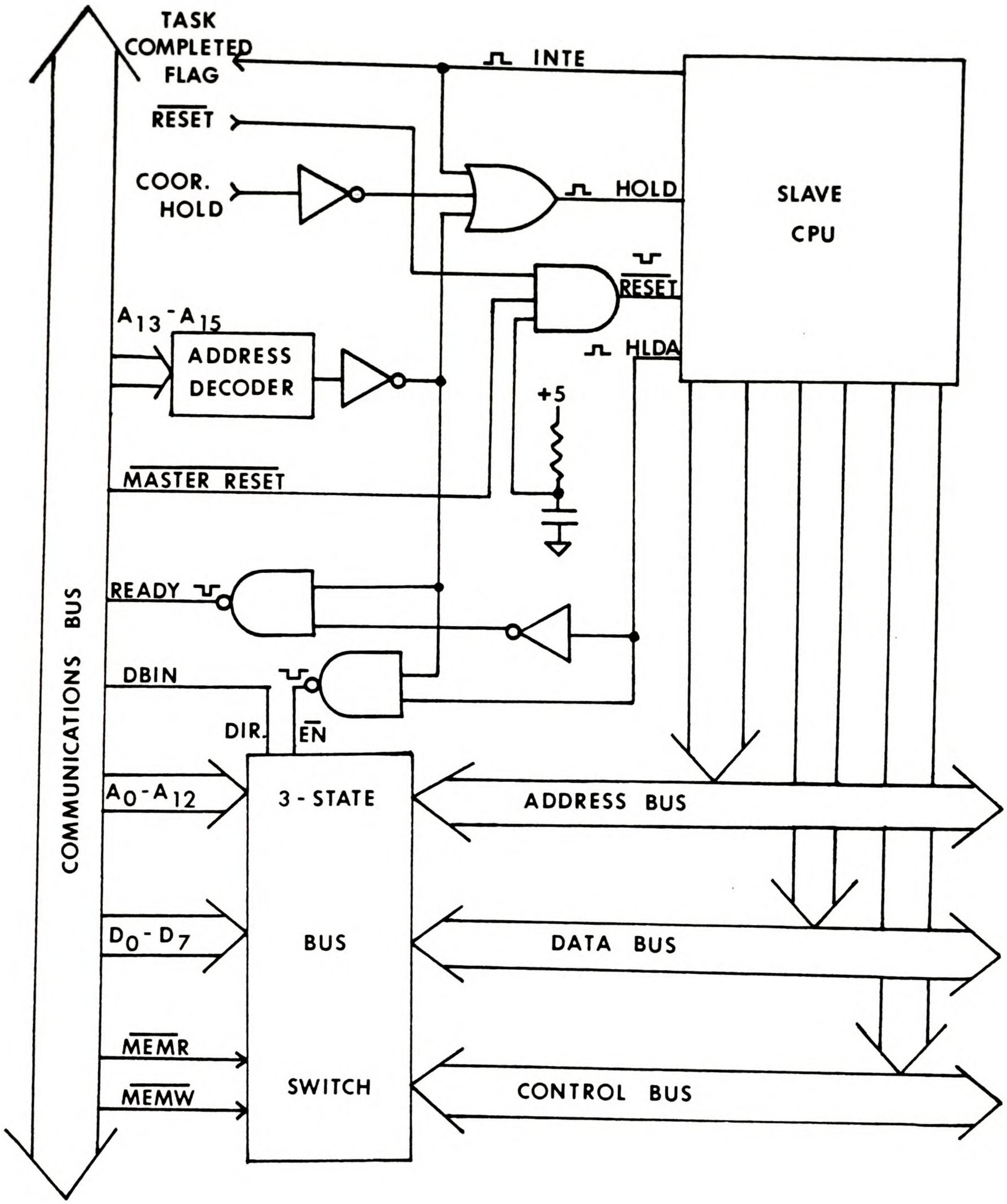


Figure 3-10. Basic Slave Bus Switch.

single, rapid one byte instruction to generate the flag. General purpose routines have been created which greatly simplify the programming required to enable the coordinator to force a slave to execute a specific subroutine. Further details about these routines are found in Chapter 8.

In most applications, the bus switch and external control lines are adequate for all forms of communication between the various microcomputers. Certain applications however, require more advanced forms of communication. One such situation is the need to transfer large blocks of data at very high data rates. The coordinator can be used to transfer blocks of data, but its data transfer rate is limited to a maximum of 50 kHz. By using direct memory access (DMA) circuits the data transfer rate can be raised to at least 1 MHz. If these DMA circuits were located on the coordinator's bus the coordinator's throughput would be seriously reduced. To alleviate this problem, the basic system can be modified as shown in Figure 3-11. By placing a bus switch between the coordinator's system bus and the communications bus, the coordinator can be isolated from the communications bus when it is being used for DMA data transfers between slaves. DMA transfers are set up by the coordinator. Any conflicts on the communications bus between the coordinator and the DMA circuitry are resolved by the switch control. The coordinator is always given the highest priority for communications bus access. Figure 3-12 shows the standard bus switch for the coordinator. When the coordinator attempts to access an address requiring the use of the communications bus, the communication bus request line (CBR) goes high. Whether or not the coordinator

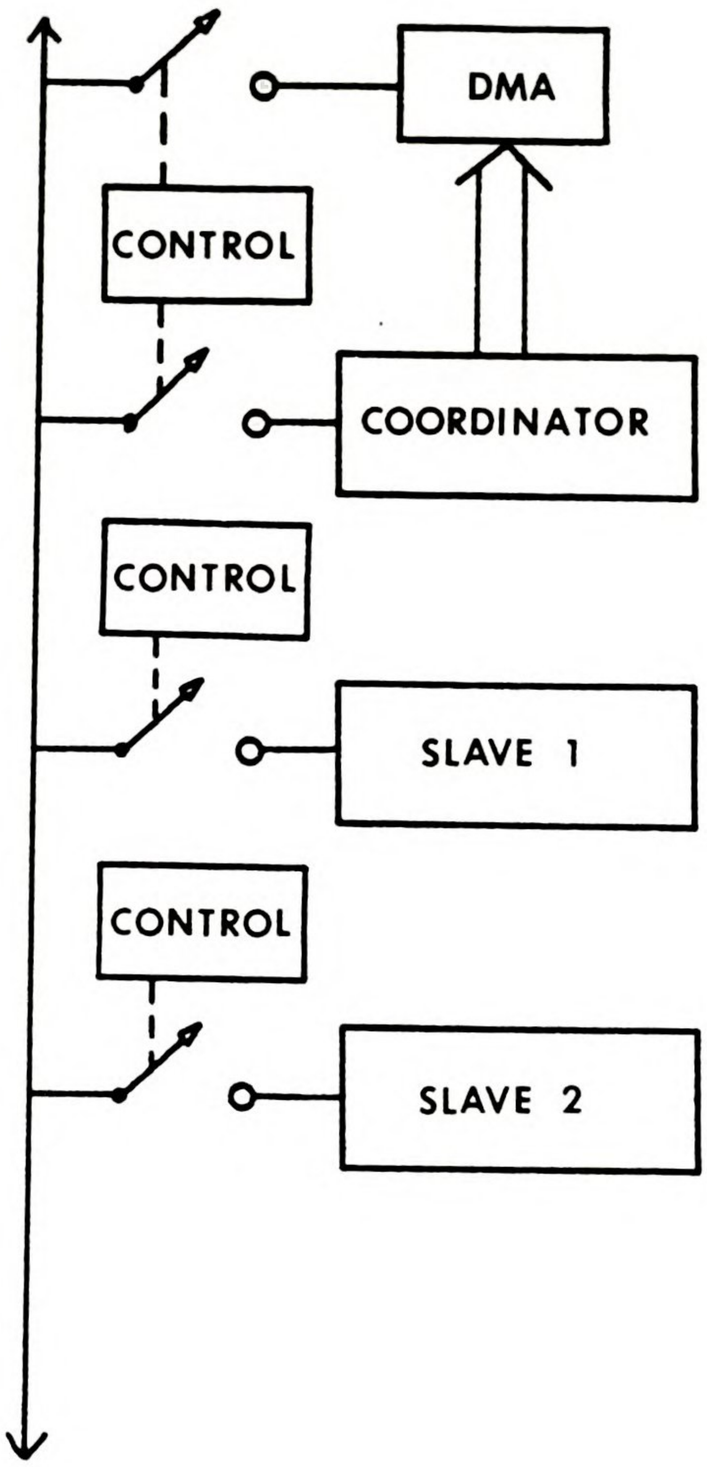


Figure 3-11. Addition of DMA Mass-Transfer.

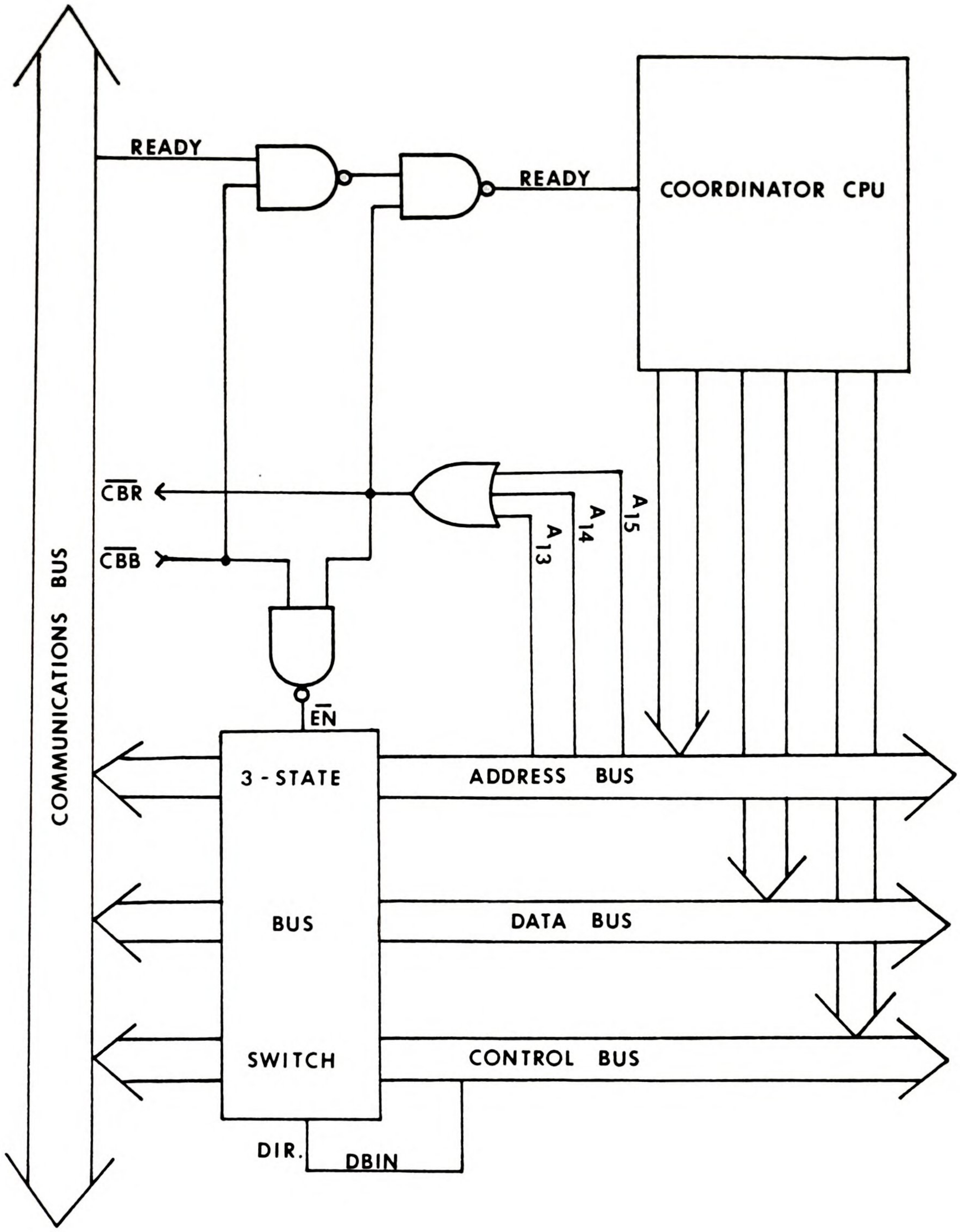


Figure 3-12. Basic Coordinator Bus Switch.

is granted immediate access to the bus is determined by the state of the communication bus busy line ($\overline{CBB}$). When this line is high, the coordinator is granted immediate access. If the bus is currently busy, this line will be low, and the coordinator will enter a wait state until the current DMA transfer cycle is completed.

Another form of communication which is sometimes required is the transfer of status words or a few bytes of data between slaves. This can be accomplished by the coordinator, but the resulting transfer is extremely slow and degrades the throughput of the coordinator and both slaves. To overcome this limitation, an additional dedicated communications bus was designed. This new bus, named the register bus, is shown in Figure 3-13. The register bus consists of 8 address lines, 8 data lines and 3 control lines ($\overline{MEMR}$, $\overline{MEMW}$ and DBIN). All microcomputers in the system have equal priority for access to the register bus. By means of the register bus, it is possible to access special registers which are located on the individual microcomputers. In this way, slave 1 could write a status word into a special register located on its bus and that status word could be read by any other microcomputer in the system by using the register bus. Bus conflicts on the register bus are resolved by the register bus switch circuit (Figure 3-14) and a register bus controller circuit. Each microcomputer requiring access to the register bus must contain the bus switch circuit. When a microcomputer requires the use of the register bus, it generates a chip select ($\overline{CS}$) signal that causes a register bus request ($\overline{RBR}$) to be sent to the controller. The microcomputer will be in a wait state until the register bus busy line ($\overline{RBB}$) goes high and

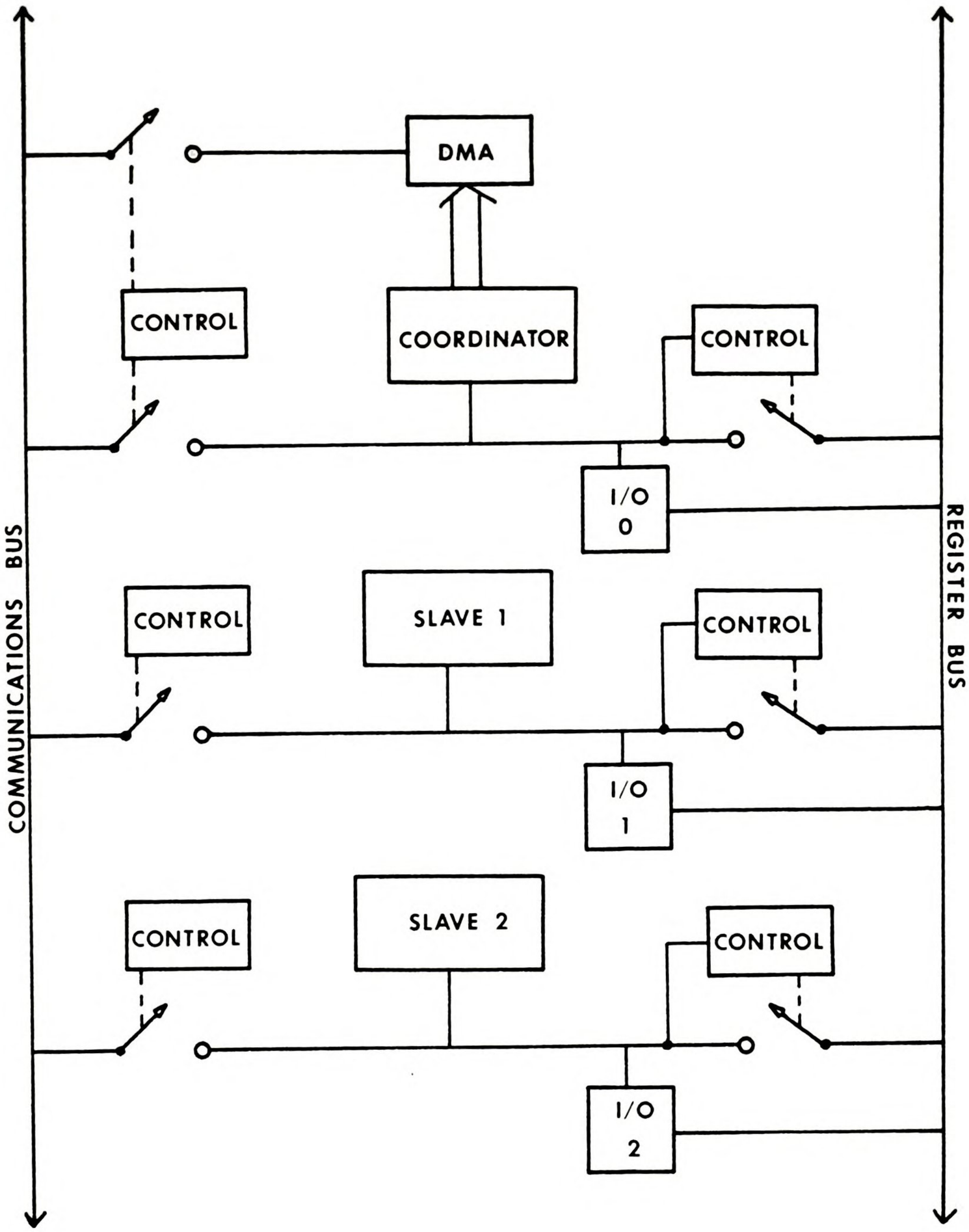


Figure 3-13. Addition of the Register Bus.

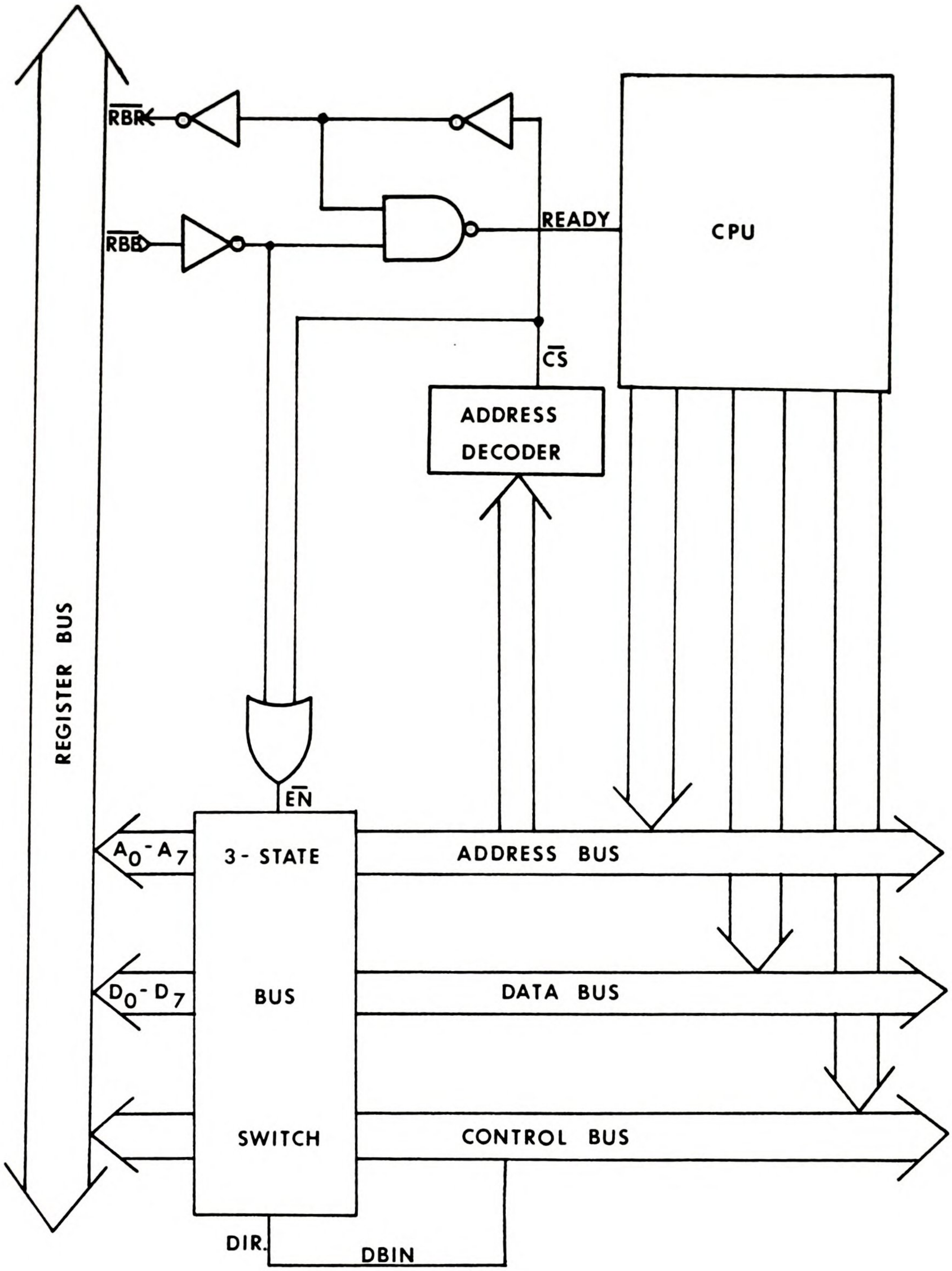


Figure 3-14. Register Bus System.

it is granted access to the bus.

Even when the register bus and DMA transfer circuits are included, the total circuitry required to implement the inter-processor communications system is surprisingly small. The standard use of wire wrapping for all digital circuit connections allows the microcomputer and interface chips to be packed tightly together. Often the entire slave microcomputer system including the CPU, memory, task interface and all the bus interfacing circuits can be easily contained on a single 6 x 8 inch circuit card. An entire multi-microcomputer system is normally contained in a standard BUD card rack. This card rack contains twelve, 58 pair printed circuit board edge connectors. The lower 39 pairs of the edge connectors are bused together. The pin assignments of this backplane system bus are given in Table 3-1. The upper 19 pair of connectors are available for individual connections between boards and to external circuits.

D. The External Communications System

One of the primary design goals of this laboratory computer system was to make each instrument virtually independent of the minicomputer during standard operation. In order to accomplish this goal adequately, the operator must be able to run a variety of experiments with the instrument, evaluate the results and store selected results without any intervention from the minicomputer. The ability to design and execute totally new experiments without using the minicomputer would also greatly improve overall laboratory throughput.

The realization of these goals requires rather complex

Table 3-1. Backplane System Bus Assignments.

Pin	Component Side	Wire Wrap Side	Comments
20	Data 6	Data 7	Register Bus
21	" 4	" 5	"
22	" 2	" 3	"
23	" 0	" 1	"
24	Address 6	Address 7	"
25	" 4	" 5	"
26	" 2	" 3	"
27	" 0	" 1	"
28	DBIN	$\overline{\text{MEMR}}$	"
29	-	$\overline{\text{MEMW}}$	"
30	-	-	"
31	-	READY	Communications Bus
32	$\overline{\text{MASTER RESET}}$	DBIN	"
33	-	-	"
34	-	-	"
35	-	-	"
36	-	-	"
37	Data 6	Data 7	"
38	" 4	" 5	"
39	" 2	" 3	"
40	" 0	" 1	"
41	Address 14	Address 15	"
42	" 12	" 13	"
43	" 10	" 11	"
44	" 8	" 9	"
45	Address 6	Address 7	"
46	" 4	" 5	"
47	" 2	" 3	"
48	" 0	" 1	"
49	+ 12V	$\overline{\text{MEMW}}$	"
50	- 5V	$\overline{\text{MEMR}}$	"
51	GND	GND	Power
52	+ 24V	+ 24V	"
53	- 24V	- 24V	"
54	- 15V	- 15V	"
55	+ 15V	+ 15V	"
56	GND	GND	"
57	+ 5V	+ 5V	"
58	GND	GND	"

communications external to the multi-microcomputer system. The system currently provides a variety of modes of external communication. Typically, communication between the operator and the system is accomplished through a video terminal. This terminal consists of a Ball 9 inch CRT monitor, a Matrox MTX-1632SL 16 x 32 character alpha-numeric display circuit, a Matrox MTX-256^2-1 graphics display circuit and an Electronics Warehouse EW-100 ASCII keyboard. The alphanumeric capabilities of the terminal allow it to be used not only for operating the instrument, but also for program development. The terminal's graphics system allows preliminary data from the instrument to be easily displayed. A Persci Model 270 Dual Diskette Drive with a Model 1070 Intelligent Diskette Controller provides local mass storage for programs and data. Parallel, high speed data transfers between the minicomputer and the multi-microcomputer systems are performed through the use of a communications controller developed in our laboratory (17). This controller also contains a link to a teletype for generating printed copies of programs and data. All of these communications tasks are performed by a single slave micro, the communicator slave micro. Standard routines have been developed for all of these task; details of these routines can be found in Chapter 8.

CHAPTER 4

THE DESIGN OF A RAPID SCANNING SPECTROPHOTOMETER FOR STOPPED-FLOW KINETICS

A. Introduction

Most spectrophotometric detectors used for stopped-flow kinetics monitor the course of a reaction at only one or two wavelengths. The ability to monitor the course of a reaction simultaneously over a range of wavelengths offers tremedous advantages in certain applications. One of the most important applications of rapid wavelength scanning techniques in stopped-flow kinetics is in the determination of the number and types of species present, particularly the detection of reaction intermediates and the subsequent investigation of the kinetics of the formation and decay of these intermediates (19, 20). Rapid scanning techniques also allow two or more reactions to be monitored at different wavelengths simultaneously (21, 22). Also, many standard kinetics investigations could benefit by the initial use of a rapid scanning system for establishing the optimum analysis condition for later study with a more sensitive single wavelength system (23).

The first part of this chapter contains a review of the types of rapid wavelength scanning techniques which have been applied to stopped-flow kinetics. The last section describes the design of the silicon vidicon rapid scanning spectrophotometer which was used throughout this research.

B. Rapid Scanning Techniques in Stopped-Flow Kinetics

A great many techniques have been developed which allow a range of wavelengths to be rapidly scanned, and a number of excellent reviews can be found in the literature (24-27). However, only a few rapid scanning techniques have been applied to stopped-flow kinetics. These techniques can be divided into two general categories according to the method by which the wavelength is scanned. Mechanical scanning techniques involve the mechanical movement of a mirror, grating, or other element in the optical system, which causes the wavelength region to be swept past a stationary detector. Array detectors comprise the other general category. In this method, the dispersed spectrum falls directly on the array detector, which performs the wavelength sorting electronically.

1. Mechanical Scanning Techniques

The first application of rapid scanning techniques to stopped-flow kinetics was reported by Dye and Feldman (28). The system utilized a Perkin-Elmer Model 108 rapid scanning monochromator (no longer commercially available), and wavelength scanning was accomplished by a rapidly rotating mirror. The resulting spectra were displayed on an oscilloscope. This system has since been modified and interfaced to a minicomputer (29, 30). The scan rate can be varied between 3 and 150 spectra per second. With a fixed sampling rate of 20.4 kHz, the resolution at the above scan rates varies from 136 to 6800 sampled points per spectrum. The width of the wavelength region which is scanned can be varied by the tilt of the rotating

mirror. Noise levels as low as 0.002 A.U. (absorbance units) have been reported (31).

A mechanical scanning technique is also used in the system developed by Wightman, et al. (32). This system is based on the Harrick Scientific Corporation rapid scanning spectrometer originally developed by Kuwana (33). Wavelength scanning is accomplished by the mechanical movement of a small mirror mounted on a low inertia galvanometer armature. The movement of the mirror is produced by the application of a sawtooth or triangular waveform to the galvanometer. The width of the spectral region scanned, the number of sampled points per scan, the scan rate and the repetition rate can all be varied under computer control. At maximum, a 50-point 250 nm spectrum can be scanned in 1.2 msec with a repetition rate of 500 Hz. However, wavelength nonlinearities were observed at high scan rates. Single scan noise levels as low as 0.01 A.U. were reported (32).

Hollaway and White have developed a rapid scanning stopped-flow spectrophotometer for applications in biochemical studies (34). This system utilizes a Norcon Model 501 scanning monochromator. Wavelength scanning is accomplished by the rotation of a wheel containing 24 mirror pairs. Variations in the mounting of the 24 mirror pairs causes a different baseline for each spectrum. This problem has been overcome by a circuit which switches in a correcting voltage for each spectrum. The detection system is linear over a range of 1.5 absorbance units. The single spectrum noise was found to be normally less than 0.02 A.U. A single spectrum of about 200 nm is taken in 1 msec with a repetition rate of 800 spectra/sec. A total

of 32 spectra, each consisting of 256 8-bit words, can be stored with the current system.

Another rapid scanning stopped-flow system for biochemical applications has been reported by Faini, et al. (35). Mechanical wavelength scanning is accomplished by the movement of a small grating attached to a galvanometer. A computer-generated triangular waveform is used to move the galvanometer. A 128 point 400 nm spectrum can be acquired in 6 msec. However, the repetition rate is only 50 spectra/sec.

2. Array Detectors

The first application of an array detector to stopped-flow kinetics was demonstrated by Milano, et al. (36). This system utilized a silicon vidicon TV camera tube (RCA No. 4532A) as a detector. The original system could operate over the wavelength region from 400 to 800 nm. Later, the use of a silicon vidicon tube with a fused silica faceplate (RCA No. C23246) extended the useful wavelength region down to about 200 nm (19). The wavelength region scanned in single spectrum could be varied from 100 to 400 nm by changing the grating. Scanning rates could be varied from less than 0.1 Hz up to 250 Hz for the standard 200 point spectrum. The detector response was found to be linear with intensity over a range of four orders of magnitude. The single scan noise of this system was 0.002 A.U.

Nieman and Enke also developed a silicon vidicon rapid scanning spectrophotometer which has been applied to stopped-flow kinetics (23,

37, 38). This system was shown to be linear over at least 3 1/2 orders of magnitude with a single scan signal-to-noise ratio of 220. The number of electronic channels in the 230 nm wide wavelength window could be varied from 32 to 4096 under computer control. Scan repetition rates as high as 500 Hz could be achieved.

Dessy has reported the design of a rapid scanning stopped-flow spectrophotometer which utilizes a self-scanning linear photodiode array as a detector (39). However, no data have ever appeared on the characterization or application of this system. The lack of application of linear photodiode arrays as stopped-flow detectors is probably due to the low sensitivity and S/N ratio which have been reported for these devices. Yates and Kuwana evaluated the use of a linear photodiode array as a rapid scanning detector and found that it was linear from 0 to 1.5 A.U. and had a maximum signal-to-noise ratio of only 100 (40). They believed that much of their noise was due to their signal processing electronics. Other sources have reported somewhat better results (41, 42).

C. The Design of the Vidicon Rapid Scanning Spectrophotometer

This section contains an overview of the design of the vidicon spectrophotometer which was used throughout this research. The actual construction of the scanning and detection circuitry was done by Jim Hornshuh, and construction details can be found in his thesis (43).

The silicon vidicon tube was selected as the detector in the stopped-flow system for a number of reasons. Previous experience in

our group with the vidicon detector (38) had convinced us of the power of this detector and suggested methods by which the overall performance of the system could be improved. In general, vidicon detectors tend to show lower noise levels than mechanically scanned detector systems. This is due primarily to a combination of two factors. The main source of noise in mechanical systems often results from mechanical vibrations in the scanning element (32), which are eliminated in the vidicon detector. Further noise improvements are a result of the integrating character of the vidicon detector. Finally, the vidicon detector allows sophisticated yet flexible computer control of the electronic scanning circuitry.

The silicon vidicon tube which was selected in this work was the RCA C23246. This tube has a fused silica faceplate to allow operation in the UV region. Figure 4-1 shows the typical arrangement of a silicon vidicon tube and associated magnetic deflection coils. When the dispersed radiation is focused on the vidicon target an equivalent charge pattern is generated. This charge pattern is read out by scanning an electron beam across the target. Normally, scanning is performed by a raster scan technique as shown in Figure 4-2. With this type of pattern, the horizontal or wavelength axis is generated by a staircase deflection of the electron beam by the horizontal deflection coil. Each step in the staircase represents a separate wavelength. Vertical scanning along the slit height axis is accomplished by applying a sawtooth waveform to the vertical deflection coil. The dashed lines in Figure 4-2 represent the vertical and horizontal retrace paths followed by the electron beam.

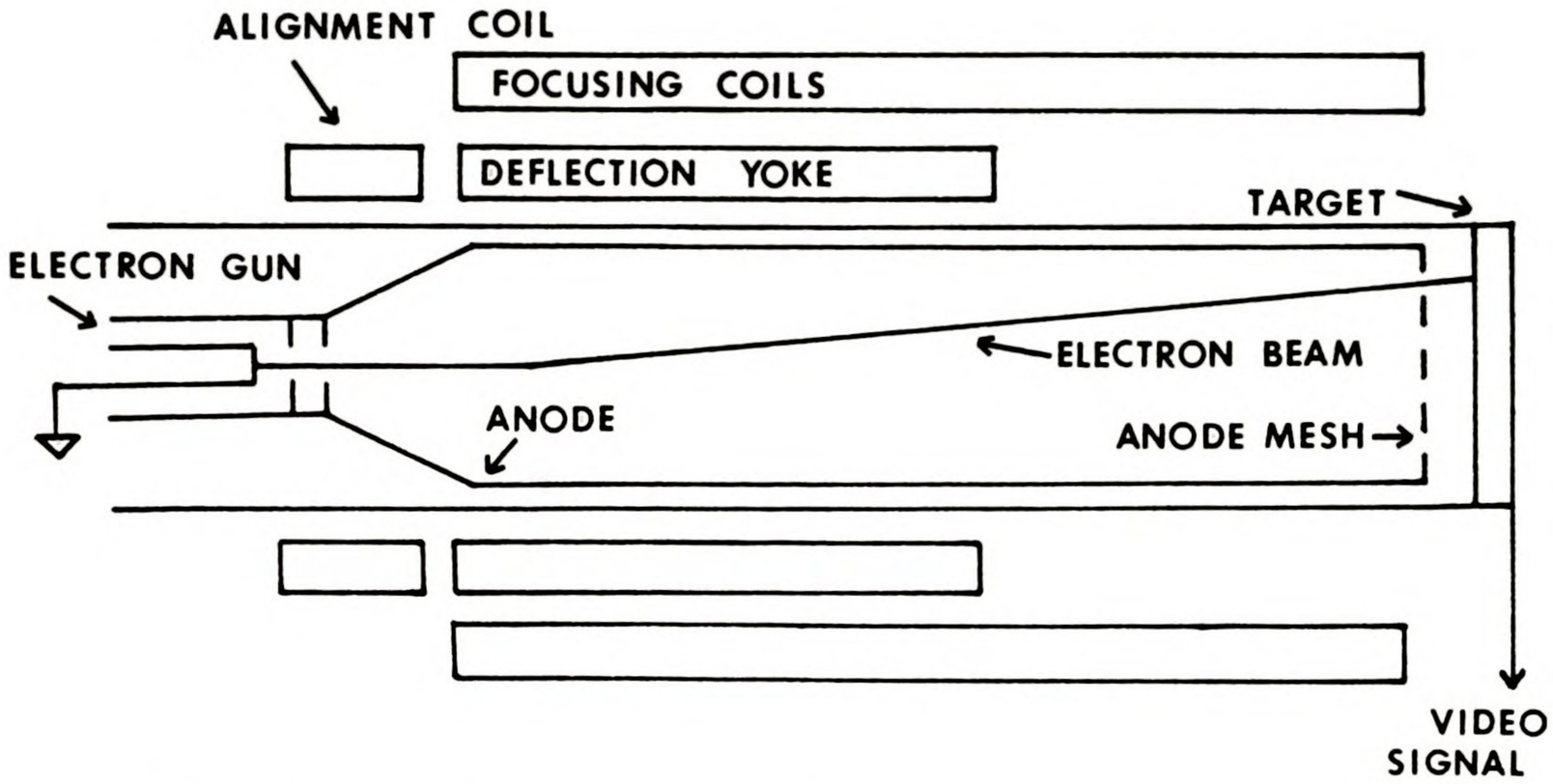


Figure 4-1. The Silicon Vidicon Tube.

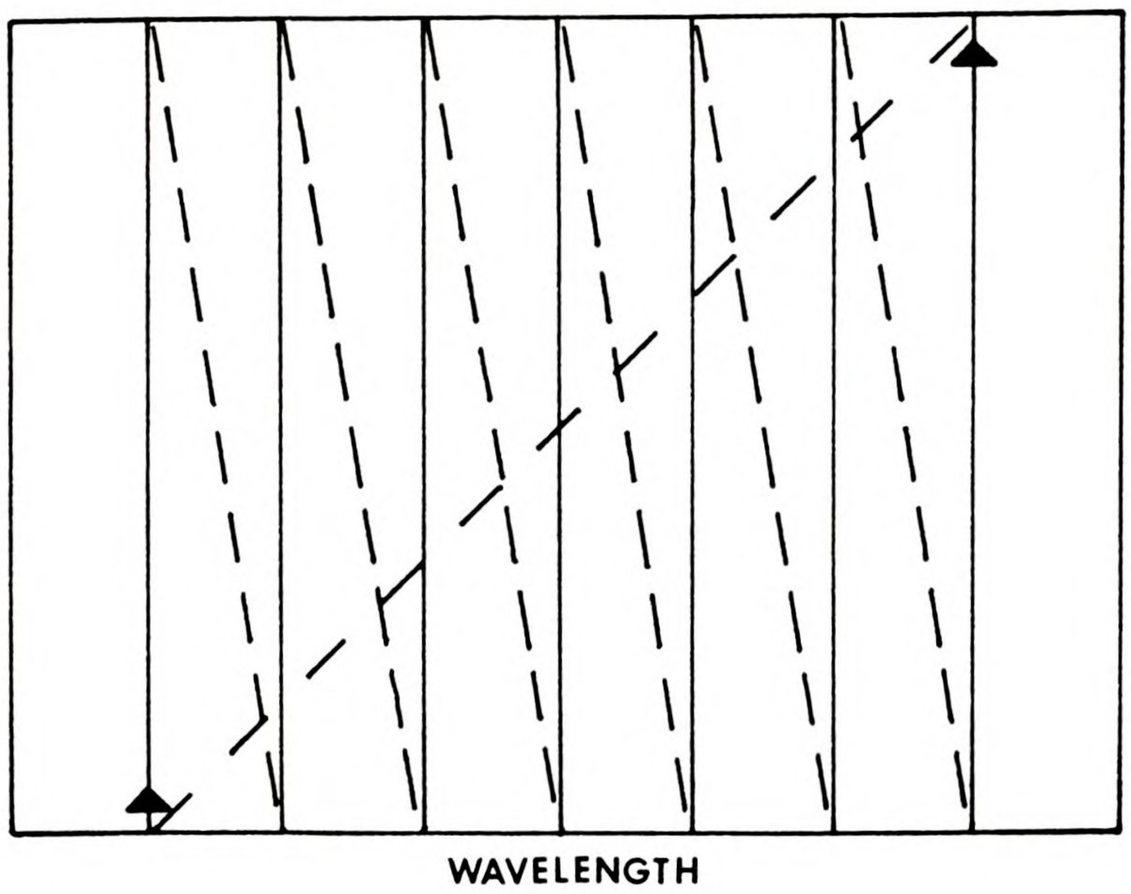


Figure 4-2. Raster Scan Pattern.

Any undesired read out of the target during retrace can be eliminated by blanking (turning off) the electron beam. For a silicon vidicon, the amount of current which exists during the scanning of the photo-induced charge pattern, can be related to the incident photon flux by the equation (44):

$$I = QeN \frac{AT}{T_a}$$

where: I = Video current, amps

Q = Quantum efficiency of the photosurface

e = Electron charge, c

N = Incident photon Density, quanta $cm^{-2}s^{-1}$

A = Target area scanned per channel, cm^2

T = Detector exposure time, s

T_a = Time to scan area A, s

Typically, the video output current varies from a few nanoamperes to a maximum of about 700 - 800 nA. The integrating character of the silicon vidicon detector is indicated by the fact that the video current is directly proportional to the detector exposure time.

To understand fully the way in which the vidicon detector integrates the incident photon flux, it is necessary to look at the vidicon target in more detail. The target consists of an n-type silicon wafer which is approximately 7/8 inch in diameter (45). The edges of this wafer are made relatively thick for improved strength while the center is extremely thin, normally about 5 - 50 μm thick. The central section of the wafer contains a large array of silicon

photodiodes, which are simply islands of p-type silicon (see Figure 4-3). The packing density of the photodiode array is typically 1800 diodes per linear inch. The active video area of the wafer is typically 5/8 inch in diameter. The p-n junctions form charge depletion regions which essentially isolate the p-type islands from the substrate and allow them to function as charge storage areas. The p-type islands can be thought of as the combination of a capacitor (charge storage region) and a photodiode. Figure 4-4 shows an equivalent circuit for the vidicon target based on this concept. The electron beam is normally kept at a voltage of -8V with respect to the substrate. As the electron beam moves across the target the p-type islands are charged to -8V. When photons fall on the opposite side of the wafer, they generate electron-hole pairs. The voltage across the wafer tends to draw the holes into the p-type islands, which thereby slightly discharge the stored charge. When the electron beam rescans these diodes the current necessary to recharge the diodes to -8V is equal to the amount of charge discharged by the photon-generated electron-hole pairs. When no light is incident on the target, charge depletion still occurs due to thermally generated electron-hole pairs. This gives rise to a video output in the absence of illumination known as the dark current.

The silicon vidicon spectrophotometer control functions, which are of primary importance to its use in rapid scanning applications, are shown in Figure 4-5. All of these functions are completely under computer control. The vertical scan frequency can be varied from 3.91 kHz to 1 MHz, but the data acquisition system limits the

photodiodes, which are simply islands of p-type silicon (see Figure 4-3). The packing density of the photodiode array is typically 1800 diodes per linear inch. The active video area of the wafer is typically 5/8 inch in diameter. The p-n junctions form charge depletion regions which essentially isolate the p-type islands from the substrate and allow them to function as charge storage areas. The p-type islands can be thought of as the combination of a capacitor (charge storage region) and a photodiode. Figure 4-4 shows an equivalent circuit for the vidicon target based on this concept. The electron beam is normally kept at a voltage of -8V with respect to the substrate. As the electron beam moves across the target the p-type islands are charged to -8V. When photons fall on the opposite side of the wafer, they generate electron-hole pairs. The voltage across the wafer tends to draw the holes into the p-type islands, which thereby slightly discharge the stored charge. When the electron beam rescans these diodes the current necessary to recharge the diodes to -8V is equal to the amount of charge discharged by the photon-generated electron-hole pairs. When no light is incident on the target, charge depletion still occurs due to thermally generated electron-hole pairs. This gives rise to a video output in the absence of illumination known as the dark current.

The silicon vidicon spectrophotometer control functions, which are of primary importance to its use in rapid scanning applications, are shown in Figure 4-5. All of these functions are completely under computer control. The vertical scan frequency can be varied from 3.91 kHz to 1 MHz, but the data acquisition system limits the

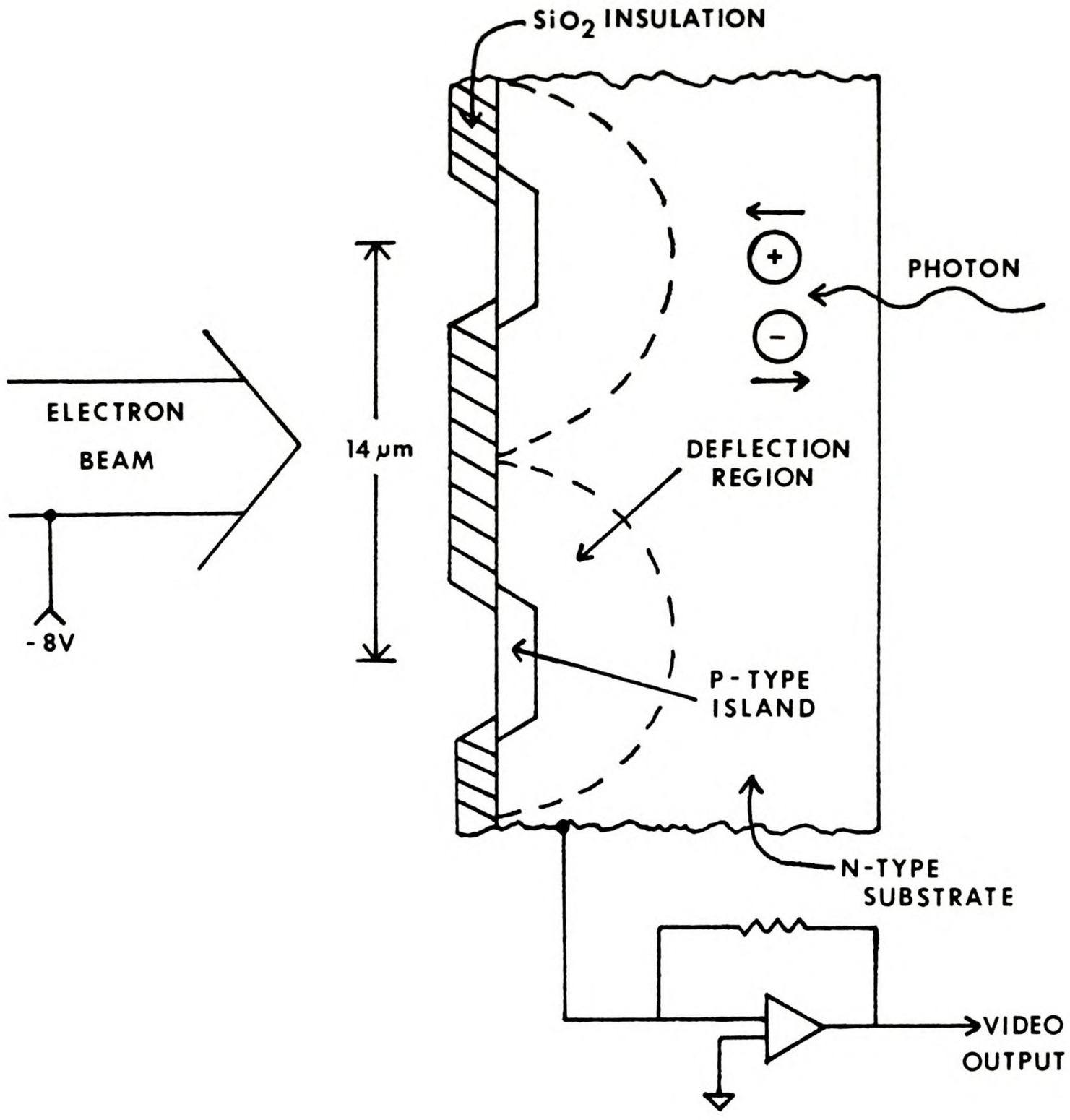


Figure 4-3. The Silicon Vidicon Target.

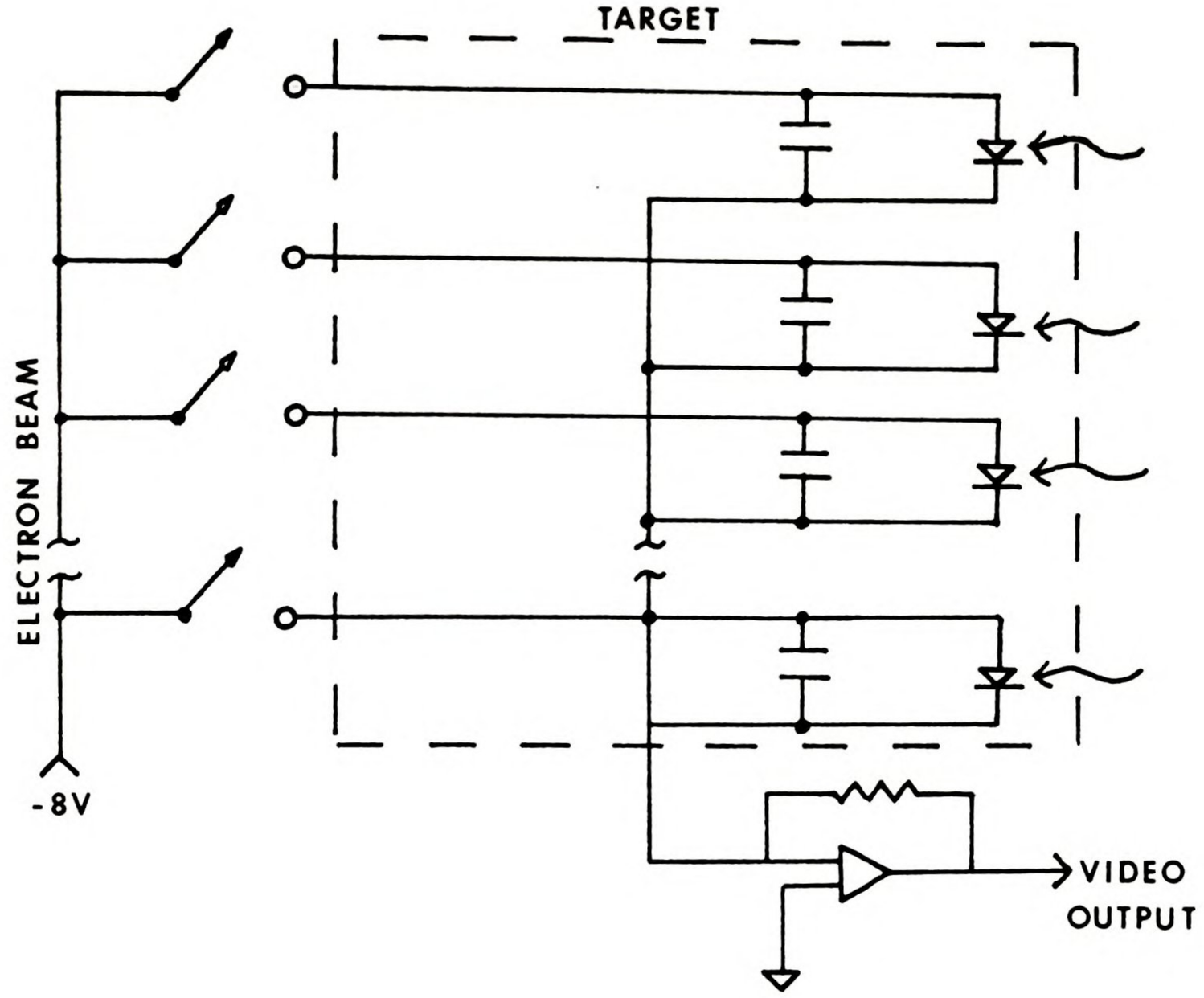


Figure 4-4. Equivalent Circuit for the Silicon Vidicon Target.

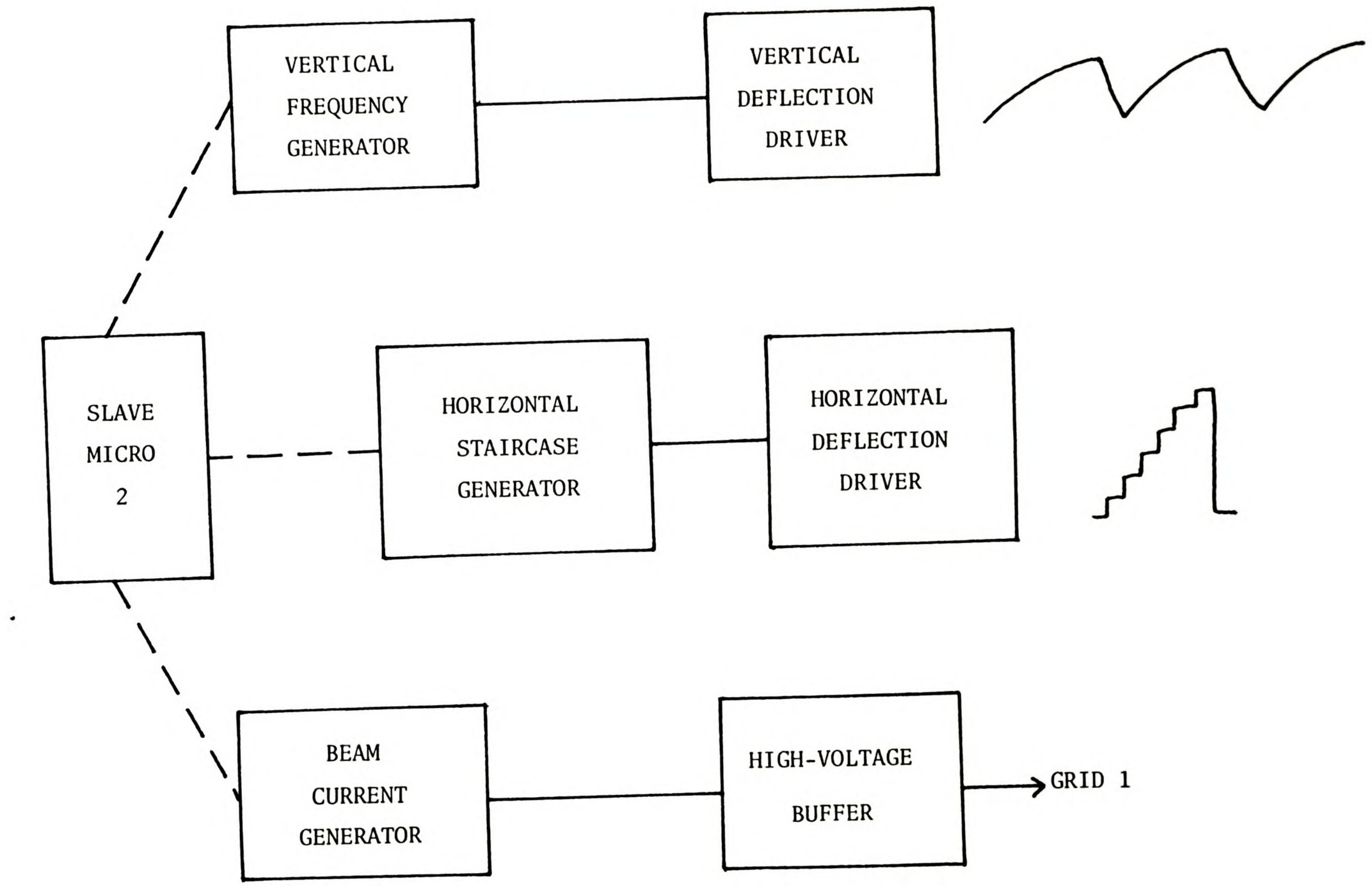


Figure 4-5. Detector Control System.

maximum practical vertical scan frequency to 100 MHz. The vertical scan frequency is produced by taking the output of a 10 MHz crystal oscillator through an 8-bit programmable divider circuit. The vertical scan frequency is then produced by dividing the divider circuit frequency by 10. The square wave created by the vertical frequency generator is then sent to the vertical deflection driver, which creates an approximately 200 mA peak-to-peak sawtooth waveform in the vertical deflection coil. The final division by 10 in the frequency generation system is done to create a control signal which is always exactly 10 times higher in frequency than the vertical scan frequency. This control signal is used in the video signal processing system discussed later.

The horizontal deflection circuitry creates the programmable staircase waveform used for electron beam deflections on the wavelength axis. The staircase waveform is produced by the continuous addition, into an accumulator, of a programmable 8-bit step width value. The 12-bit accumulator is reset to zero on overflow. The number of steps per staircase can be found by dividing 4096 by the step width value. This technique allows the number of channels per spectrum (steps per staircase) to be varied from 16 to 4096. The addition is controlled by the vertical scan. The accumulator output goes to a D/A converter (DAC) and finally to the horizontal deflection driver, which drives the 20 mA pp staircase through the horizontal deflection coil.

The length of time required to read out a single spectrum depends on both the vertical scan frequency and the number of channels

per spectrum. The single spectrum scan time in msec can be determined by dividing the number of channels per spectrum by the vertical scan frequency in kHz. For example, a vertical scan rate of 50 kHz with 100 channels per spectrum would result in a spectral scan time of 2 msec.

The final control system of major importance in rapid scanning regulates the electron beam current. The beam current is controlled by the negative voltage applied to Grid 1 in the vidicon tube; the more negative the voltage the smaller the beam current. This voltage is controlled by an 8-bit DAC coupled with a high voltage buffer amplifier. The voltage on Grid 1 can be varied from about zero V to over -100 V.

Figure 4-6 shows the data acquisition system. Amplifiers A1 - A3 constitute the preamplification system for the vidicon detector. They are mounted within a few inches of the vidicon tube. The video output current from the vidicon target is converted to a voltage signal by amplifier A1. This voltage level is then amplified and driven back to the multi-microcomputer control system by A2 and A3. Additional voltage gain is provided by A4, and the resulting signal is electronically integrated by A5. The conversion of the integrator output to a digital form is accomplished by a sample-and-hold and a 12-bit A/D converter. A programmable, crystal-controlled sequencer is used to control the data acquisition system. The control of solid state switches S1 and S2 assures that A5 integrates the signal only from a single vertical scan. These switches can also be programmed to enable a limited amount of vertical scan

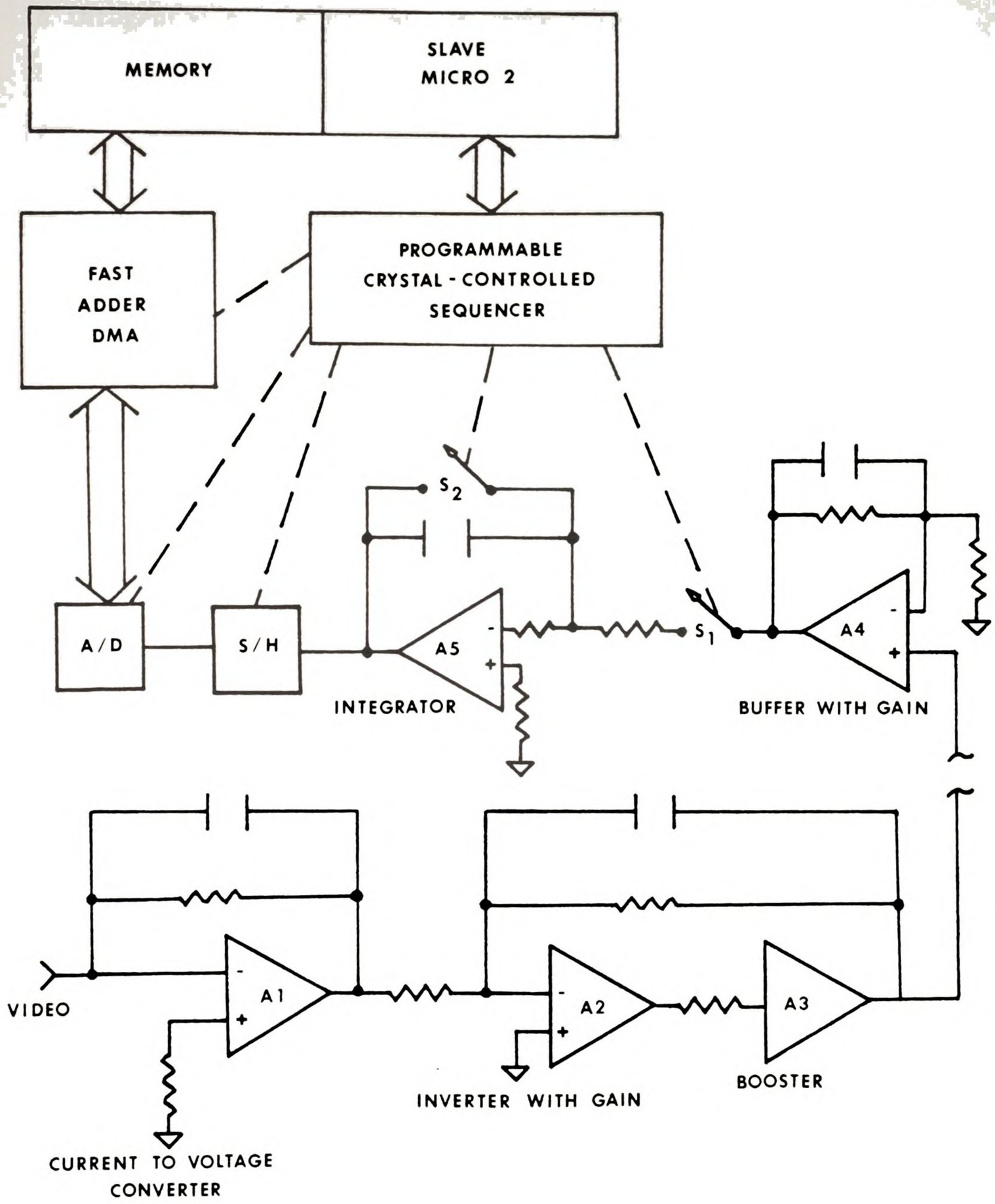


Figure 4-6. Data Acquisition System.

resolution to be obtained. The section of the vertical scan which is integrated can be varied, under computer control, by varying the timing of the closure of S1. In this way, up to 8 or 9 bands (depending on the vertical scan frequency) can be resolved on the vertical axis.

The data rates of up to 100 kHz of which the system is capable are much too fast for the microcomputer to acquire through software. To overcome this limitation, a specialized direct memory access (DMA) circuit was designed. This DMA circuit can be programmed to store spectra at any location within an 8K-byte data memory block. The DMA circuit operates by adding together the contents of the data memory and the incoming spectral data. This allows simple averaging of multiple spectra. The number of bytes of precision used can be varied under software control.

CHAPTER 5

THE DESIGN AND CONSTRUCTION OF AN AUTOMATED STOPPED-FLOW MIXING SYSTEM

A. Introduction

The stopped-flow mixing system described in this chapter is simply a sample handling system which allows two solutions to be rapidly mixed (in less than 0.01 s) and delivered to an observation cell. Any subsequent reaction between the two solutions can then be monitored by the vidicon spectrophotometer (see Chapter 4). The first stopped-flow mixing system was constructed by Chance in 1940 (46-48). Since that time a large number of modifications and applications of stopped-flow mixing systems have appeared in the literature. Recently an excellent review of the stopped-flow technique and existing stopped-flow instrumentation was published (49). The level of automation in stopped-flow systems varies widely. In the most highly automated systems (50-53), the instrument is nearly completely automated from the initial sample preparation through data acquisition and analysis. In the stopped-flow system described in this chapter, a slightly more conservative approach to the automation of the instrument was taken. In this system the sample preparation has been left to the operator.

B. The Stopped-flow Mixing System

The stopped-flow mixing system will be described in this section by first describing the overall operation of the system. The function

and construction of each of the major components of the system will then be presented in more detail.

A block diagram of the stopped-flow mixing system is shown in Figure 5-1. In a normal stopped-flow run (or push), the first operation is the filling of the drive syringes with the sample solutions. This is accomplished by the pneumatic drive piston shown at the top of Figure 5-1. The drive piston pulls the syringe plungers back, while the check valves insure that the syringes are filled from the sample solutions only. Next, the direction of the air pressure in the drive piston is switched, energizing the drive syringes. No solution flows at this time since the check valves keep the solutions from returning to the sample containers and the 3-way slider valve shown at the right of Figure 5-1 is in the closed position. The flow begins when the position of the pneumatically controlled slider valve is switched, connecting the observation cell to the stopping syringe. The flow continues to force the plunger of the stopping syringe to move back until the end of the plunger strikes the stopping block. At this time the solution flow stops rapidly and the reaction occurring in the observation cell can then be monitored. After the reaction is completed, the slider valve changes position, which connects the stopping syringe to the waste container. Springs mounted on the stopping syringe push the plunger into the syringe, and force the solution out into the waste container.

The various parts of the system are connected together by 1/8 inch Teflon tubing (1.5 mm I.D.) with Altex chemically inert tubing end fittings (Altex Scientific Inc., part no. 200-15). The pneumatic

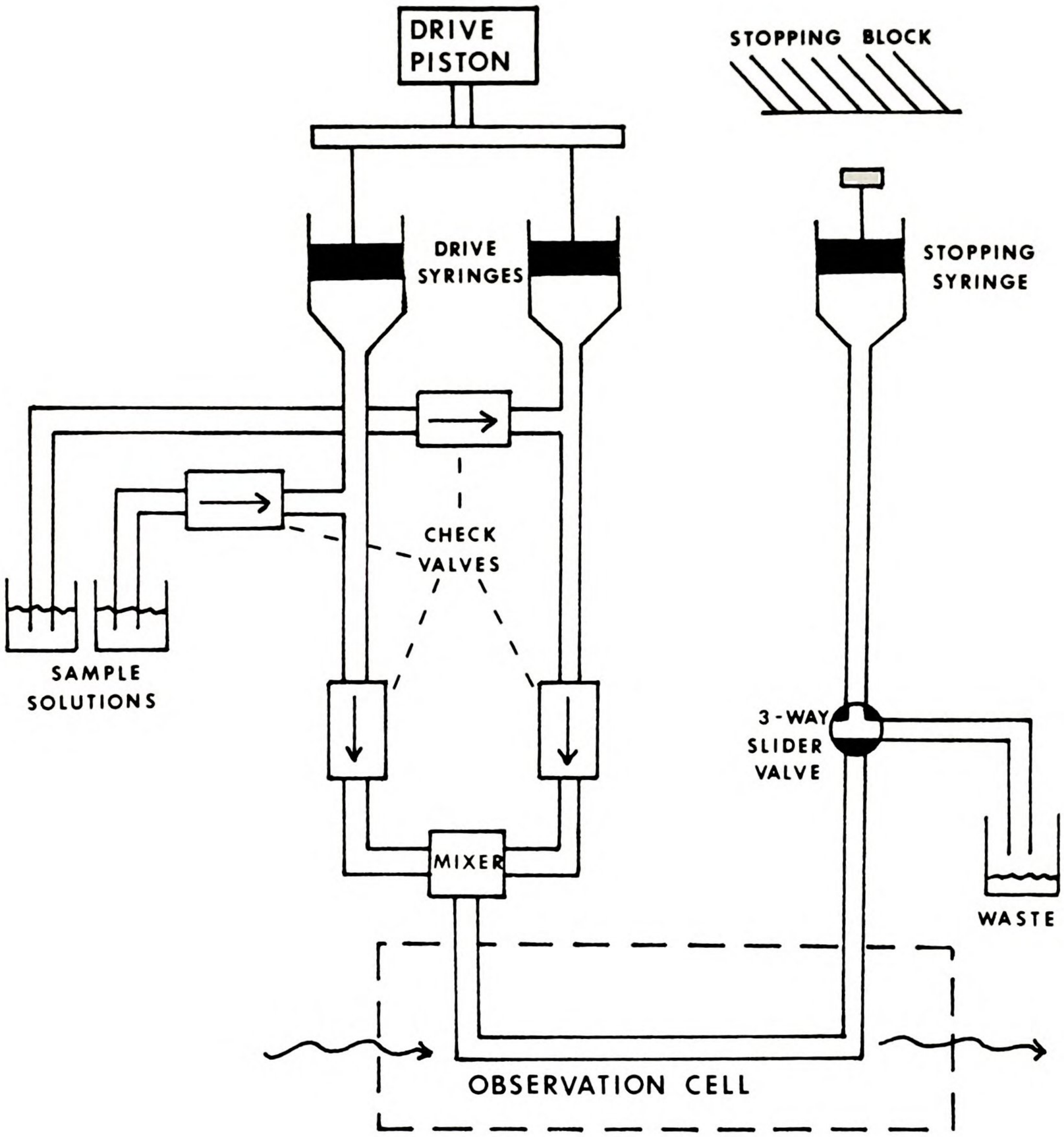


Figure 5-1. Block Diagram of the Stopped-Flow Mixing System.

drive piston (Schrader Gold Medallion miniature pneumatic cylinder; 1 1/8 inch bore, 1 inch stroke) is controlled by a Scoville pneumatic solenoid air valve (part no. 41001-1115). The drive and stopping syringes were made by the Michigan State University Glass Shop by sealing together precision bore pyrex tubing (Wilmad Glass Co. Inc., with 0.1736 and 0.1751 inch I.D.) and Altex glass connectors (Altex Scientific Inc., part no. 200-28). The syringe plungers are 1 ml gas-tight syringe plungers (Glenco Scientific Inc., part no. 19925-P-1). The check valves (The Lee Co., part no. TCKA6201050A) were modified to fit the Altex connector system by drilling out the ends of the check valves. The 3-way slider valve (Altex Scientific Inc., part no. 201-04, 201-12, 201-13) is controlled by a solenoid air valve (Altex Scientific Inc., part no. 903-00). Both this solenoid air valve and the one controlling the drive piston can be opened and closed under computer control through the use of solid state relays (Teledyne Relays, part no. 611-3).

A cut-away view of the stopped-flow mixer and observation cell is shown in Figure 5-2. All of the components, except for the tubing end fittings and the quartz rods, were constructed by the Chemistry Department Machine Shop. The light path is connected to the stainless steel observation cell by 2 mm quartz rods with ground and polished ends (Wilmad Glass Co., Inc.). The quartz rods are sealed to the ends of the observation cell by the compression of teflon washers by aluminum compression fittings. The Notz-type, Kel F mixer (54) is also sealed to the observation cell by compression from the Teflon mixer adapter.

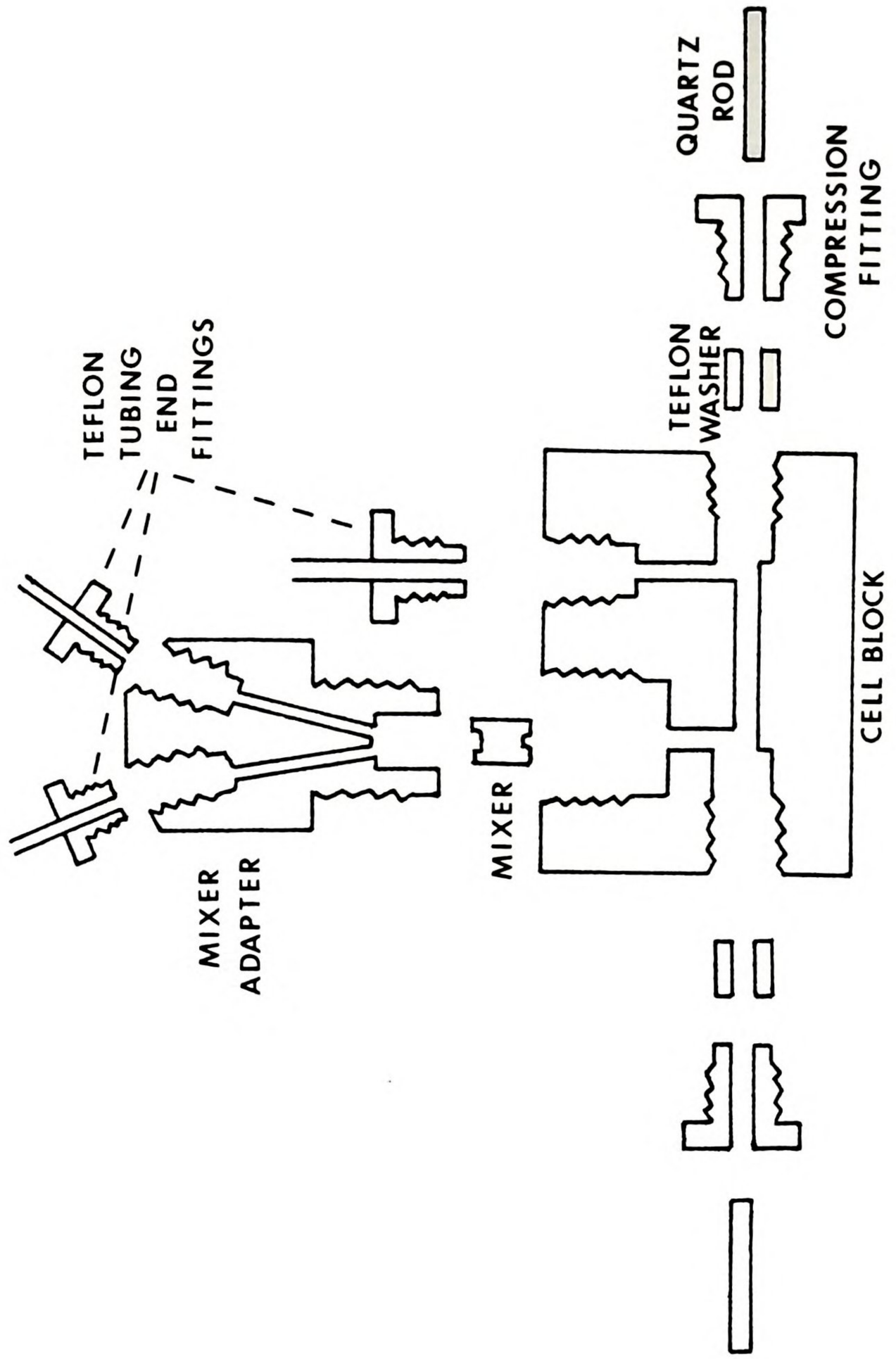
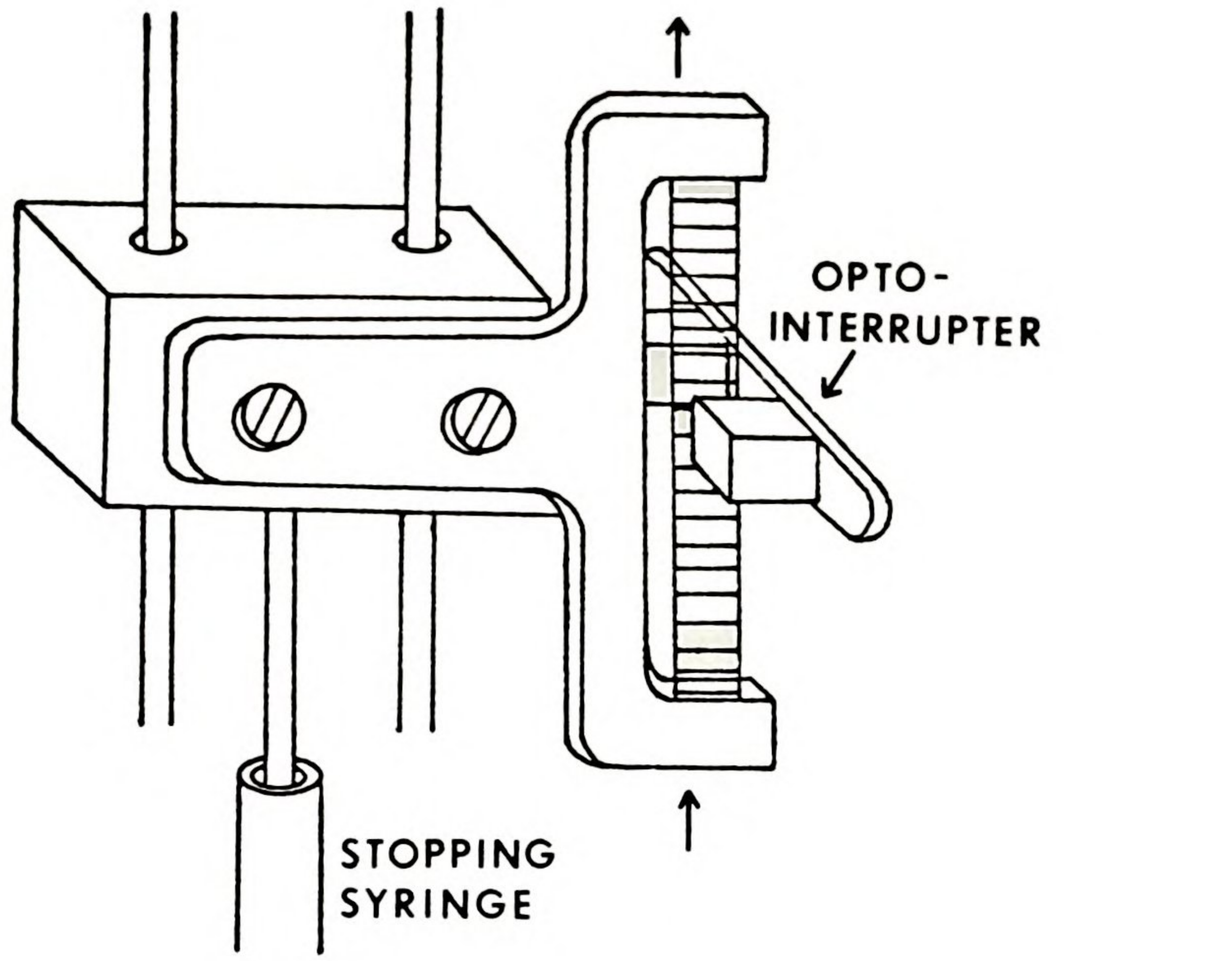


Figure 5-2. Stopped-Flow System Mixer and Observation Cell.

C. Stopping Syringe Velocity/Position Transducer

The velocity/position transducer developed in this research is based upon a design originally presented by Holler (55). The transducer enables the position of the stopping syringe plunger to be determined at any time during a stopped-flow run. In routine operation, the transducer allows the computer system to determine the time at which the solution flow stops (the trigger point). This is accomplished by determining when the stopping syringe plunger reaches the position at which it strikes the stopping block. The determination of the trigger point is very important, since it is at this time that the vidicon spectrophotometer must begin data acquisition. The transducer can also be used to monitor the stopping syringe velocity during a stopped-flow run. This is useful in diagnosing any irregularities in the solution flow and also in determining the dead time of the mixing system (see Chapter 6).

The transducer is shown in Figure 5-3, and consists of two parts. A precision ruling with 50 lines per inch is mounted on the stopping syringe plunger. The ruling consists of a photographic contact negative made from a ronchi ruling (Edmund Scientific Co., part no. 30 511). The second part of the transducer is an opto-interrupter module (General Electric H13A1). As shown at the bottom of Figure 5-3, the opto-interrupter consists of a light emitting diode and a phototransistor separated by a narrow slot. The opto-interrupter is mounted so that the ruling passes through the slot. As the dark lines of the ruling pass through the slot, they block part of the light falling on the phototransistor. This causes the resistance of the phototransistor



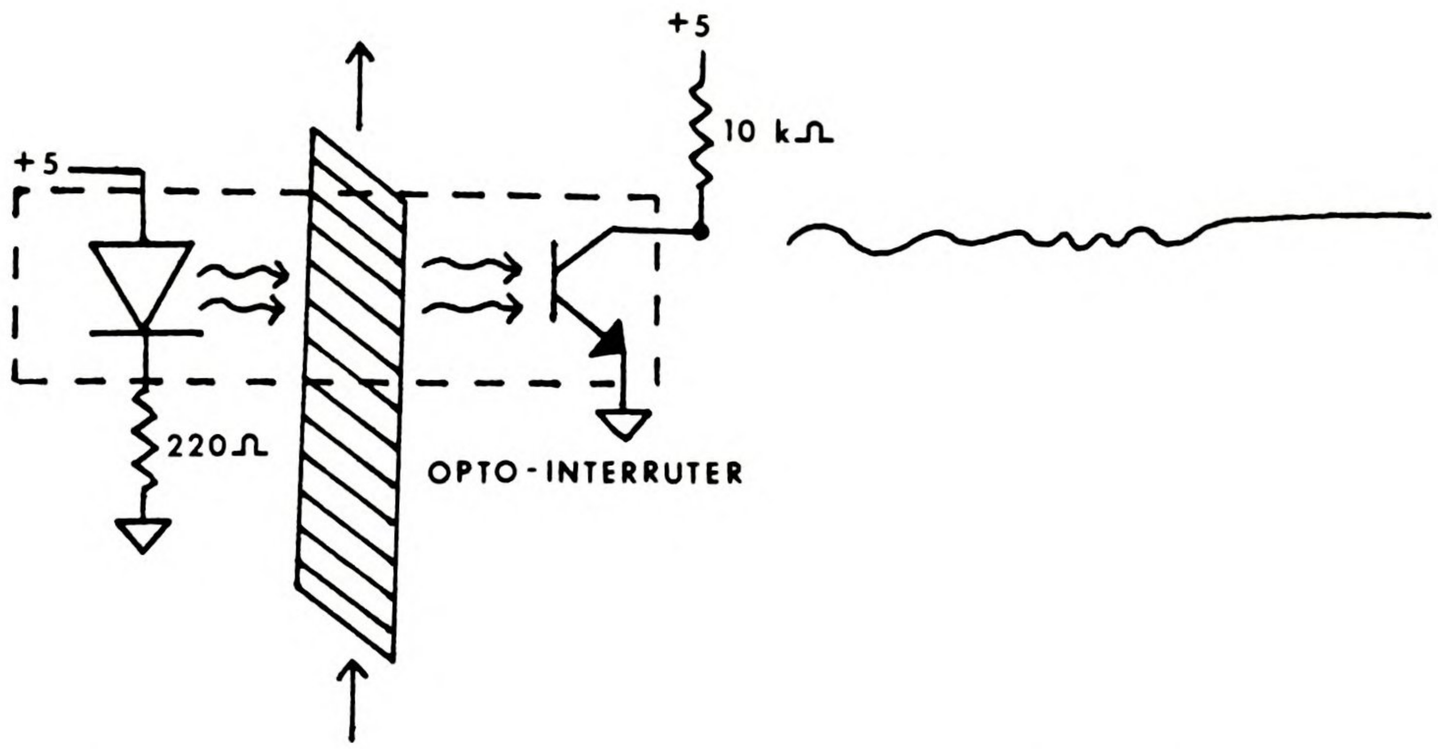


Figure 5-3. Stopping Syringe Velocity/Position Transducer.

to vary slightly, which produces a voltage waveform at the photo-transistor's collector. This waveform resembles a sine wave of varying frequency with an amplitude of a few tenths of a volt. The period of the sine wave is equal to the length of time required for the stopping syringe to move from one line to the next (0.0508 cm with the 50 lines/inch ruling).

Figure 5-4 shows the processing circuitry which measures the period of the transducer output. The output of the phototransistor (T1) has a 1 to 2 V d.c. offset which is removed by the high-pass filter formed by C1, R2 and R3. R2 and R3 are used to set the d.c. offset of approximately 0.1 V. Amplifier A1 (RCA CA3160) amplifies this signal and produces an output waveform of about 3 Vpp with a d.c. offset of 1.5 V. A square wave with indentical frequency is created by passing the signal through comparator A2 (National Semi-conductor LM311). Resistor R7 prevents false triggering of the comparator by providing about 0.5 V of hysteresis. The square wave is then sent to the microcomputer rack. The period of the square wave is measured by two 16 bit counters (Intel C8253) with a 1 MHz time base. The square wave causes the output of the flip-flop (c_1) to change state after each period. Since the opposite outputs of the flip-flop are connected to the enables of the two counters, each counter is on for exactly one period of the square wave. The micro-computer acquires the period measurement by watching the Q output of the flip-flop. When the output changes state the microcomputer reads the count acquired by the counter which was enabled during the previous period of the square wave. The microcomputer then clears

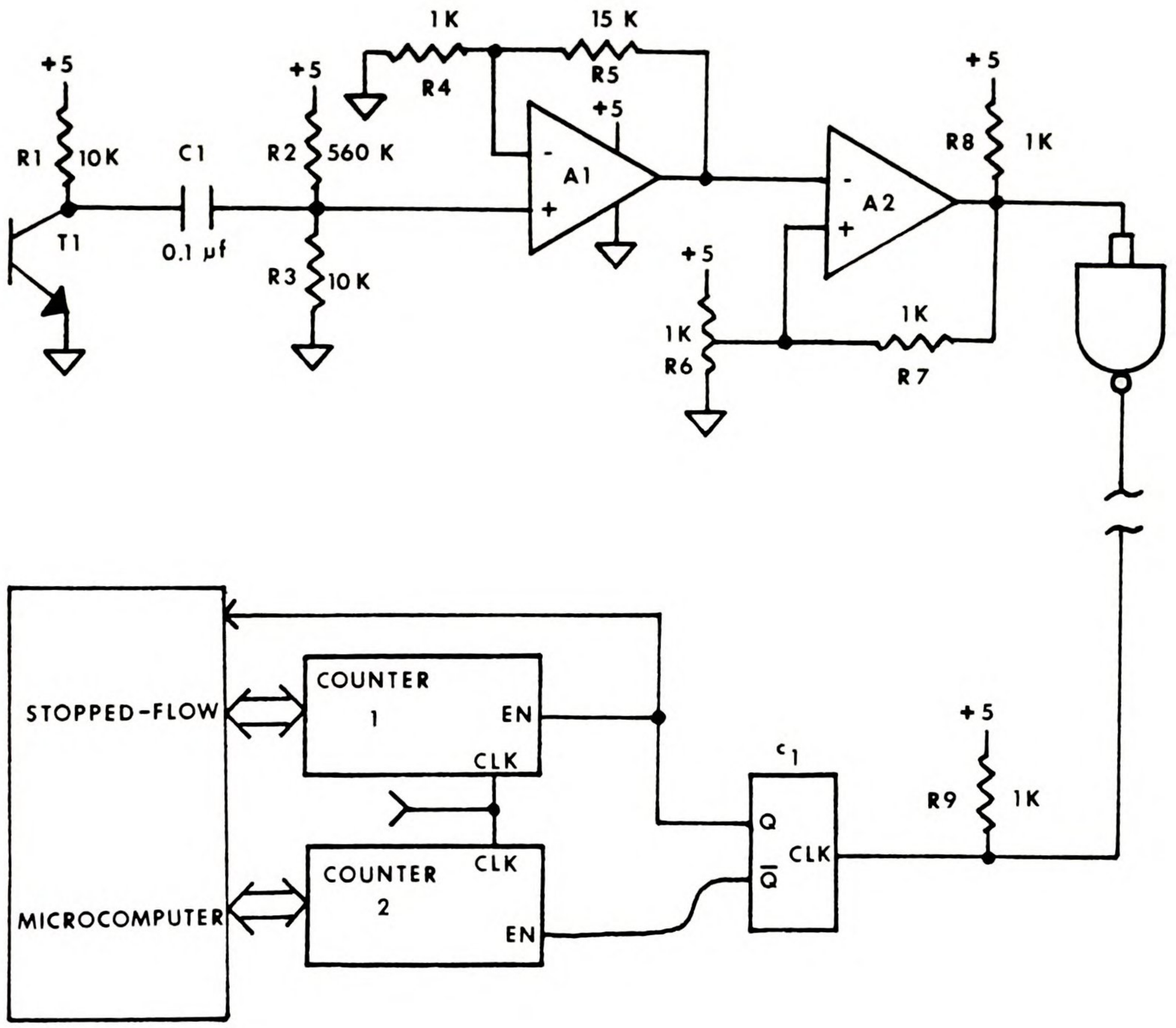


Figure 5-4. Velocity/Position Transducer Processing Circuitry.

the counter and begins watching for the next transition of the flip-flop output. The result is an array of data which represents the length of time (in microseconds) required for each pair of lines on the ruling to pass the opto-interrupter. The position of the syringe at any time can be independently determined by counting the number of transitions of the flip-flop output.

CHAPTER 6

CHARACTERIZATION OF THE VIDICON, RAPID SCANNING, STOPPED-FLOW SPECTROPHOTOMETER

A. Introduction

In order to realize the full potential of any instrument, its operating characteristics must be well understood. Of primary interest is the determination of the instrument's optimum operating conditions and the identification of any features which limit its effectiveness. This chapter examines some of the operating characteristics of the vidicon, stopped-flow system which are important to its application in stopped-flow kinetics. The first section presents the characteristics which are specific to the vidicon spectrophotometer, while the last section concentrates on the characteristics of the stopped-flow solution handling system.

B. Characterization of the Vidicon Spectrophotometer

1. Introduction

The operating characteristics of the vidicon spectrophotometer can be classified as either static or dynamic characteristics. Static characteristics are observed when the vidicon detector is at steady state (i.e. constant illumination and scan parameters). When the system is perturbed from steady state by changing the illumination or scan parameters, dynamic characteristics are observed. Dynamic characteristics are often expressed as the time or number of scans

required to establish a new steady state condition.

Many of the basic static characteristics of the vidicon spectrophotometer developed in our laboratory and of other similar spectrophotometers have already been studied (36, 38, 43). The characterization studies presented in this section include new studies on both static and dynamic characteristics of the vidicon spectrophotometer.

2. Vertical Scan Frequency

The vertical scan frequency of the vidicon spectrophotometer, which can be varied from 3.91 kHz to 100 kHz (see Chapter 4), has a dramatic effect upon the intensities observed from the vidicon detector. To study this effect, the vidicon target was illuminated with a tungsten continuum and read out with 203 channels per frame at varying frequencies. Figure 6-1 shows the relationship between the relative intensity observed at the center of the target and the vertical scan frequency. The rapid decrease in observed intensity at higher frequencies is due primarily to two factors. Much of this decrease is due to the decrease in exposure time at higher frequencies. When the observed intensities are normalized to a constant exposure time, a maximum response is observed at about 33 kHz as seen in Figure 6-2. The decrease in intensity at frequencies above and below 33 kHz is mainly due to incomplete scanning of the vidicon target. Figure 6-3 shows the relative vertical drive amplitude at various frequencies. The decrease in amplitude at low and high frequencies results in incomplete scanning of the upper and lower edges of the

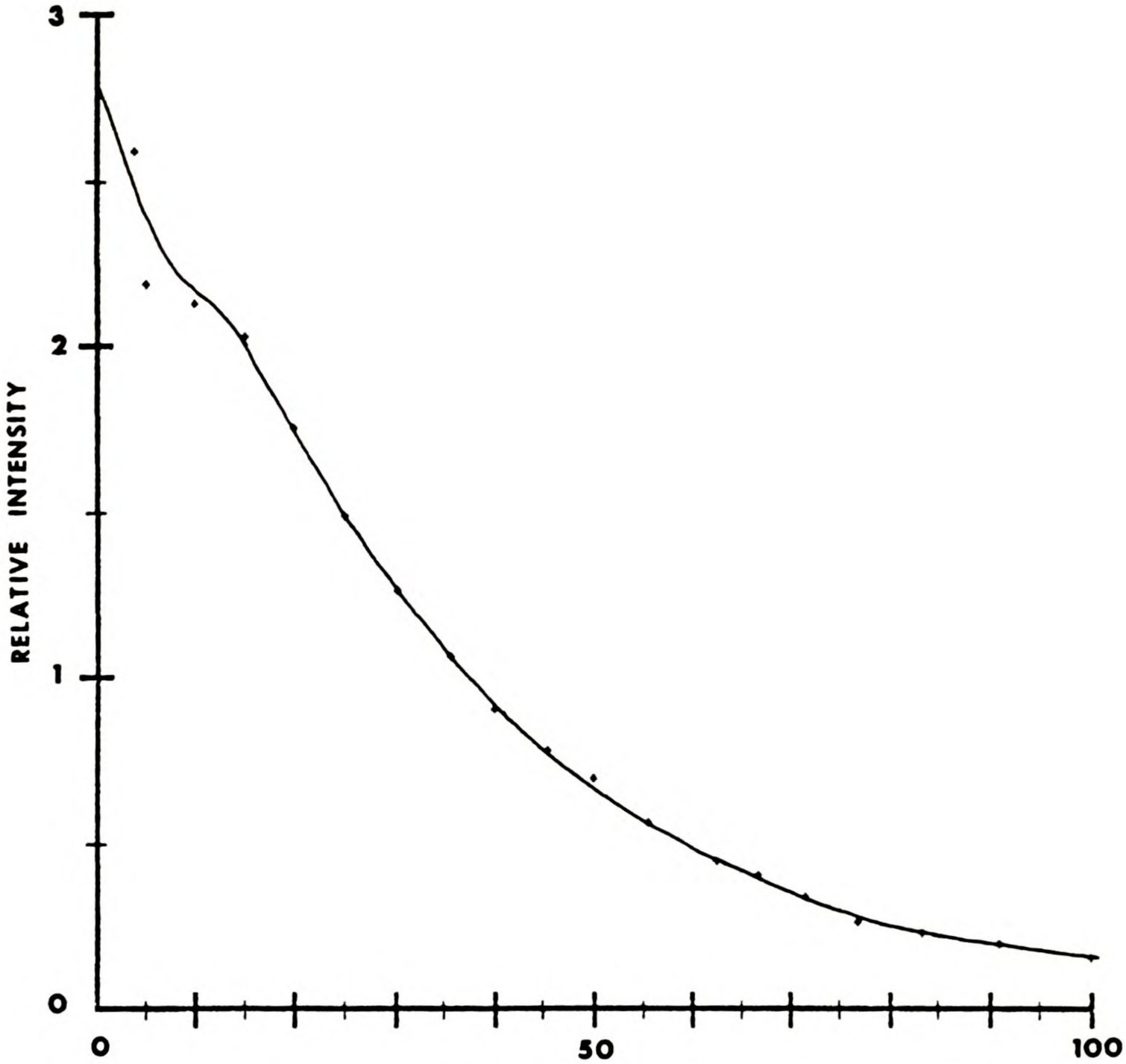


Figure 6-1. Relative Intensity Versus Frequency.

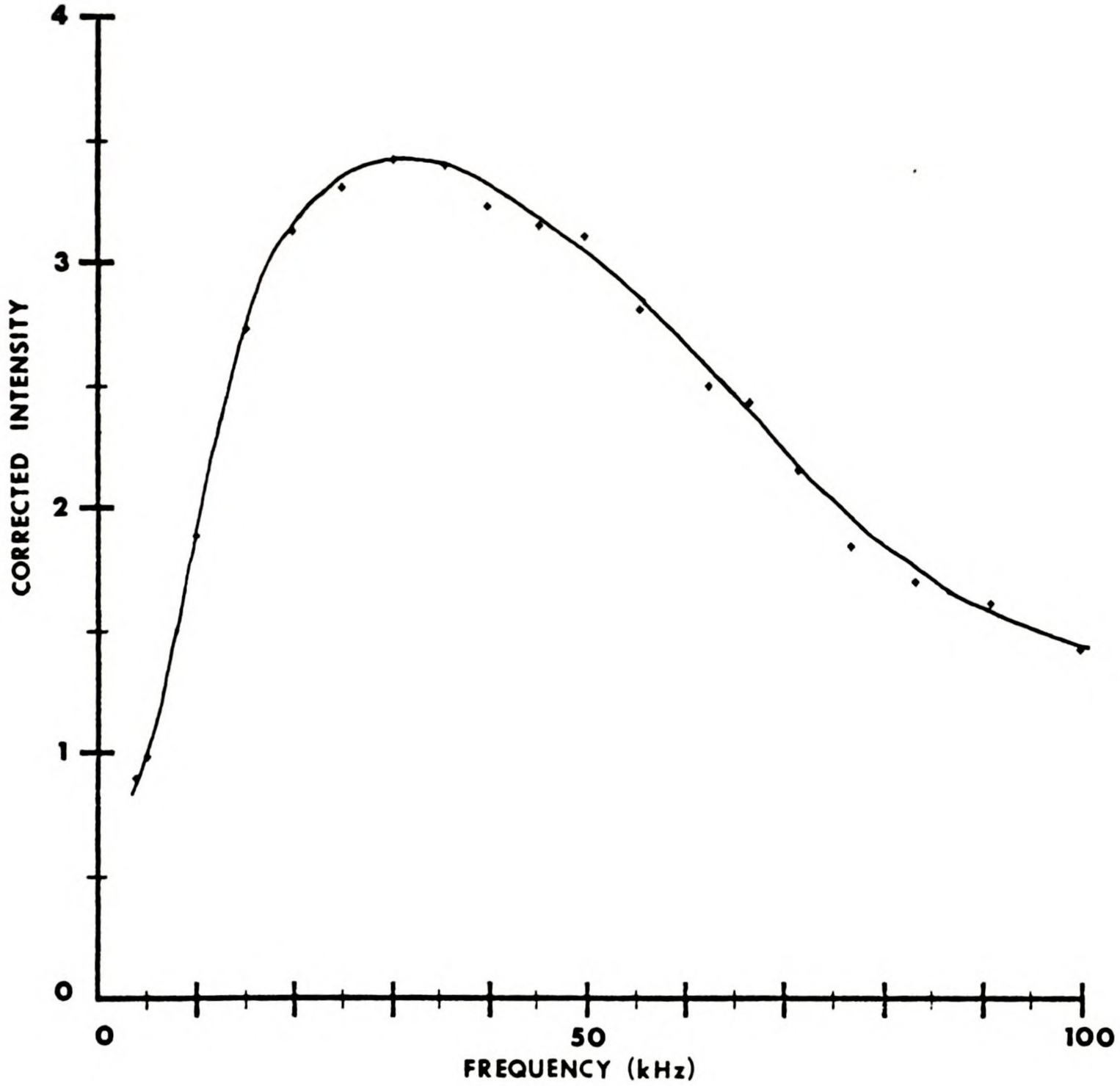


Figure 6-2. Relative Intensity Corrected for Varying Exposure Time Versus Frequency.

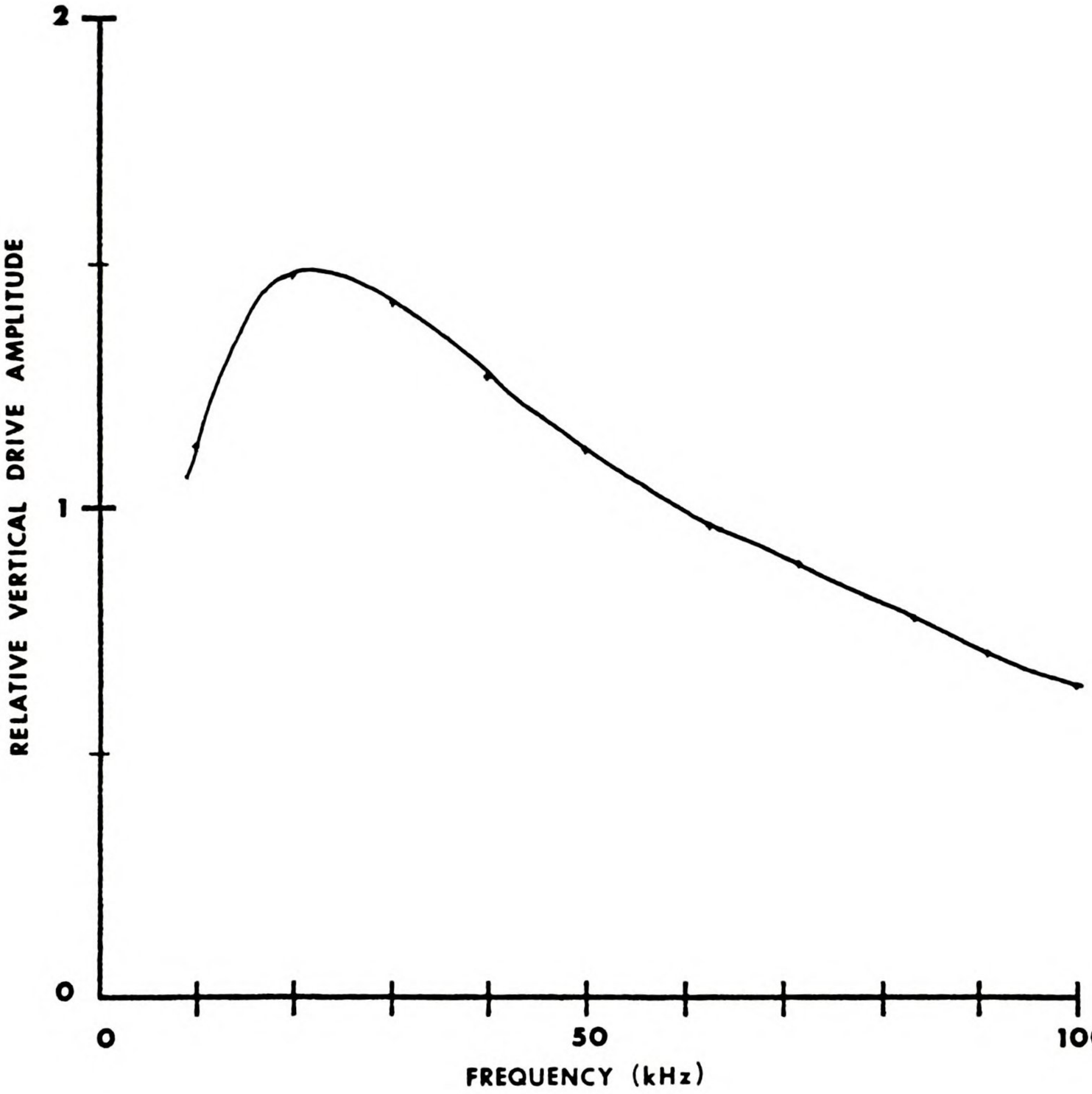


Figure 6-3. Relative Vertical Drive Amplitude Versus Frequency.

target and, therefore, a decrease in intensity.

In kinetics experiments it would often be very useful to be able to change the vertical scan frequency during the course of a reaction. For example, the early, rapid portion of a reaction could be observed at a high scan frequency for good time resolution and then, as the reaction slows down, the scan frequency could be decreased to maximize the observed intensity. To study the effect of changes in frequency, the vidicon target was exposed to constant illumination by placing a red LED a few inches from the target. The horizontal resolution was set at 255 channels per frame. The vidicon detector response was observed at the center of the target for 10 frames. The frequency was then changed and the response was observed for an additional 40 frames. Figure 6-4 shows the results that were obtained. When the frequency was increased from 25 kHz to 50 kHz the observed intensity decreased as expected. Approximately 11 scans were required to bring the response within 10% of the final 50 kHz value. When the frequency was decreased from 50 kHz to 25 kHz, a tremendous spike was observed in the vidicon response. This can be explained by remembering that at 50 kHz a significantly smaller portion of the target is being scanned. The portions of the target which are not being scanned are in saturation. When the frequency is reduced to 25 kHz, the portions which were in saturation are again being scanned. This results in a sharp increase in observed intensity which requires at least 30 scans to bring it within 10% of the final 25 kHz value. The large number of scans required to re-establish steady state severely limits the usefulness of frequency changes during the course of a reaction.

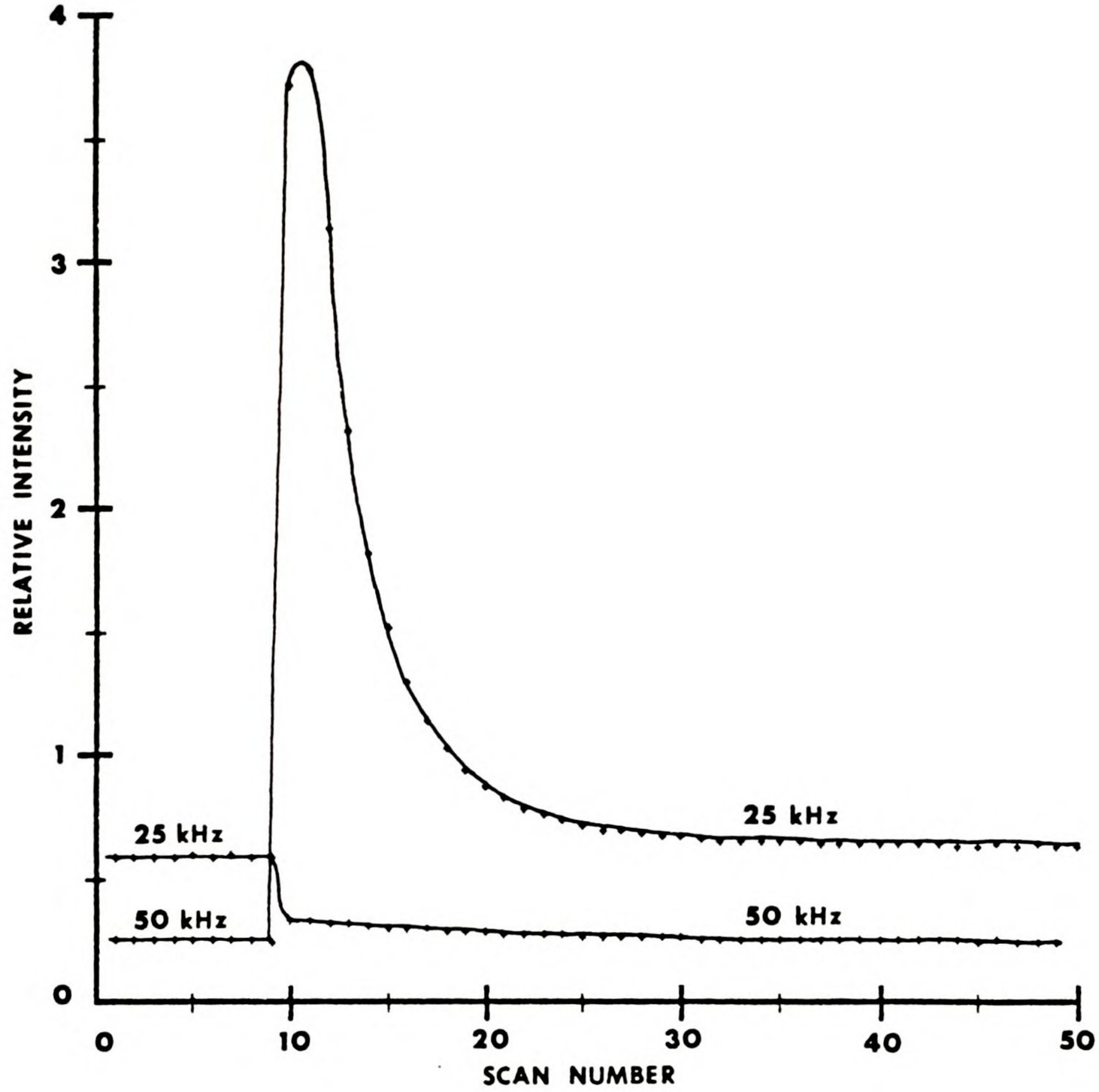


Figure 6-4. The Effect of Instantaneous Changes in Vertical Scan Frequency.

3. Resolution

It has already been established that the resolution of the vidicon spectrophotometer is limited by the vidicon detector (43), and it is determined by the number of channels per frame which are used. The maximum resolution which can be obtained is limited by the electron beam width of the vidicon detector. To study the resolution of the vidicon detector, the target was illuminated with a tungsten continuum. Figure 6-5 shows the relationship between the observed intensity at a vertical scan frequency of 25 kHz and the number of channels per frame. As the number of channels per frame is increased the exposure time is also increased. Normally an increase in exposure time results in an increase in observed intensity. However, the observed intensity is relatively constant from about 65 channels/frame to over 400 channels/frame. Previous studies (37, 38) have interpreted this plateau region as an area in which the increase in exposure time is balanced by the reduction in signal caused by the partial overlap of adjacent channels. The extreme width of the plateau region makes the "balance" theory rather suspect.

A more reasonable explanation of the plateau region can be developed as follows. Since the exposure time is proportional to the number of channels, the intensity is also proportional to the number of channels. The number of channels per frame is also inversely proportional to the effective target area scanned. Therefore, the signal intensity increase due to the increased exposure time is exactly canceled by the decrease in the effective target area per channel. In order for this explanation to be valid, the charge

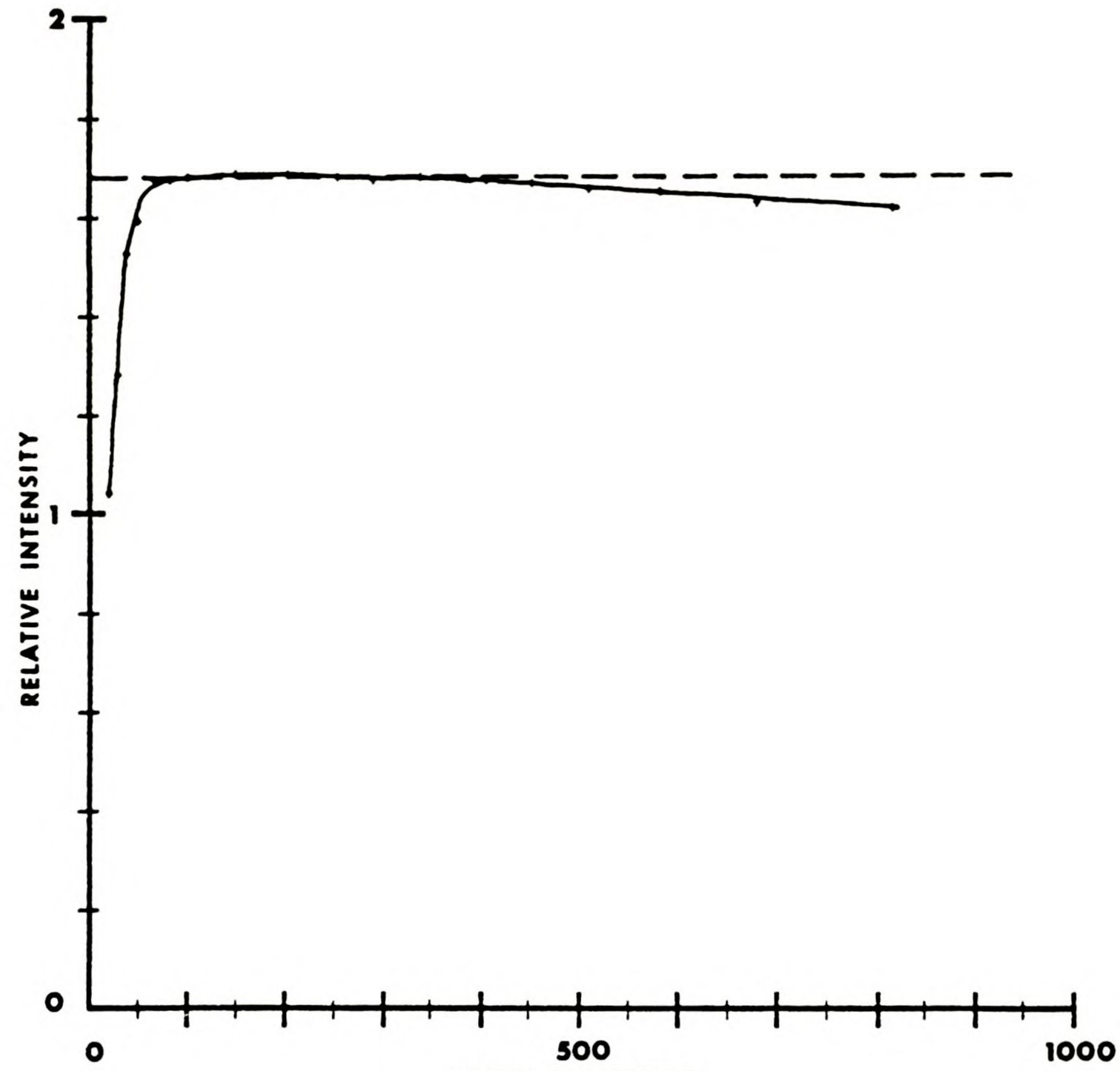


Figure 6-5. Relative Intensity Versus Resolution.

generated by photons falling upon the areas between the channels must also be read out by the electron beam. This requires the migration of charge from the saturated regions between the channels into the channel areas themselves. Charge migration in regions of saturation is a well known phenomemon which is normally referred to as "blooming" or "bleeding" (56). The implications can be more easily understood if the data of Figure 6-5 are expressed in terms of the observed intensity as a function of the channel separation. The channel separation is determined by dividing the width of the scanned portion of the vidicon target (12.8 mm wide) by the number of channels. This relationship is shown in Figure 6-6. The rapid decrease in intensity at small channel separations is due to the increasing overlap of adjacent channels. A much less dramatic decrease in intensity is also observed when the channel separation becomes relatively large. This is due to the increasing probability of electron-hole recombination when relatively extensive charge migration is required. The intersection of the region of adjacent channel overlap and the plateau region should give a good indication of the electron beam width. This technique was used in an experiment to determine the minimum electron beam width which can be obtained. The results are shown in Figure 6-7. The optimum current through the focus coil was determined to be approximately 39.2 mA. This is very close to the recommended current of approximately 40 mA (57). The minimum beam width, as determined by this technique, is approximately 30 μm, which is significantly narrower than the 54 μm beam width previously reported for this instrument (43). It also differs

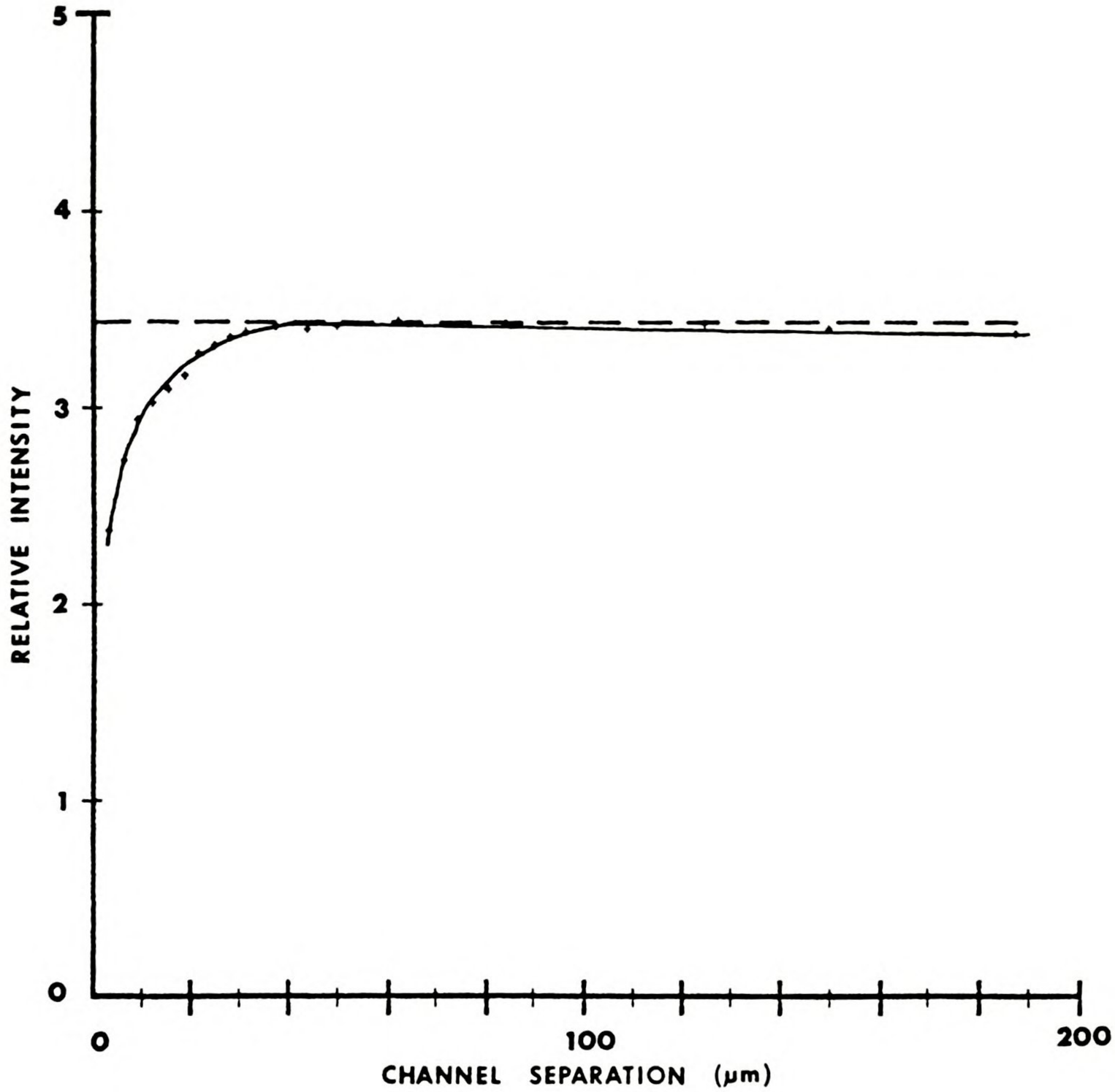


Figure 6-6. Relative Intensity Versus Channel Separation.

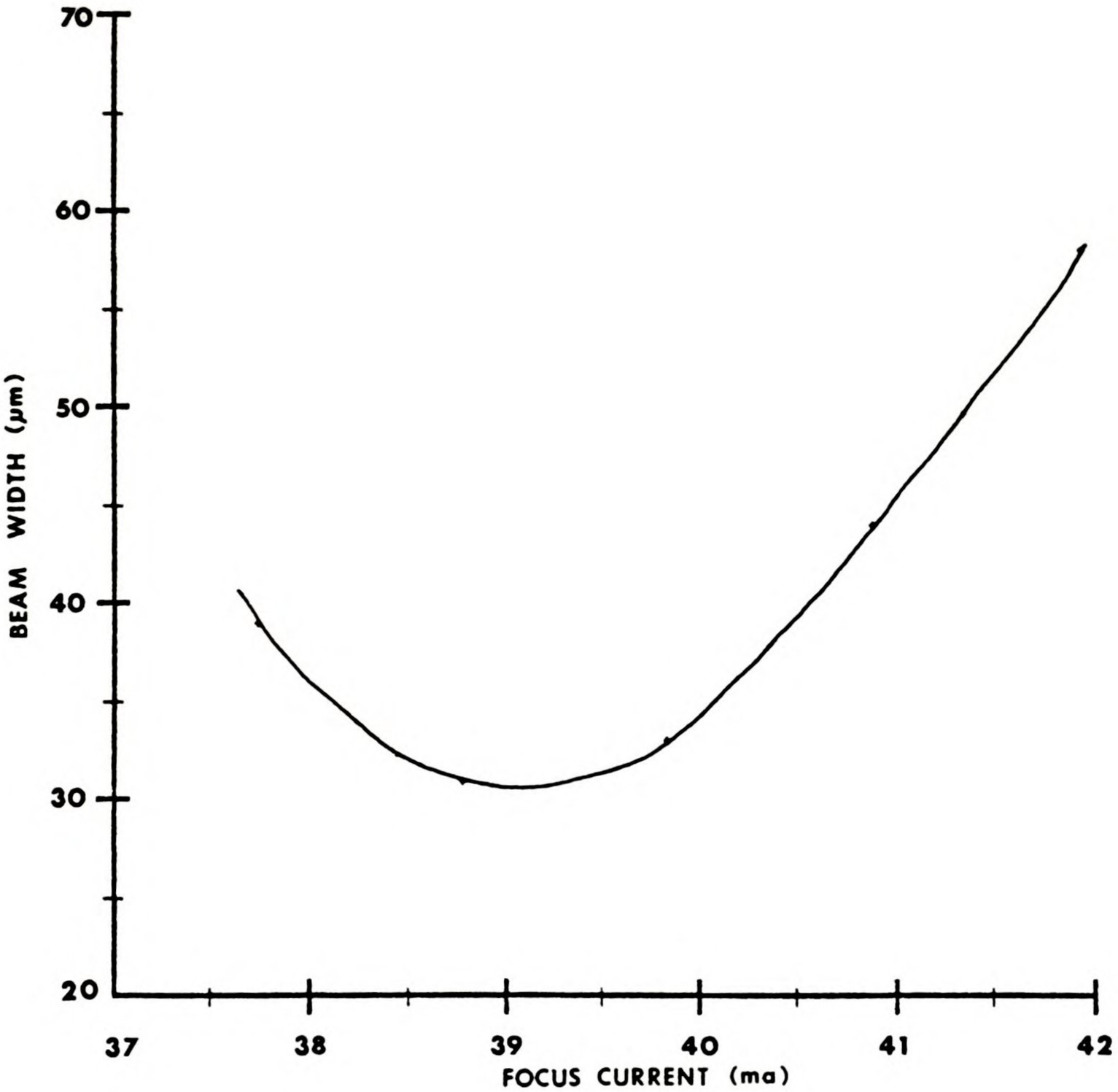


Figure 6-7. Beam Width Versus Focus Coil Current.

significantly from the 51 μm (58) and 62 μm (59) beam widths that were reported for similar systems. This indicates that most of the reading capability is concentrated in the center of the beam, while the diffuse edges of the beam make the actual beam width much larger. This new technique is nevertheless very important since it provides a very reproducible indication of the relative beam width, which is independent of the bandpass of the monochromator used in the system.

At a given vertical scan frequency, the length of time required to scan an entire frame depends on the number of channels per frame. This means that a trade-off between resolution and frame scan rate can be made. For example, it might be advantageous to select low resolution, high frame scan rate conditions in the early, rapid part of a reaction and then change to high resolution, low scan rate conditions toward the end of the reaction. To investigate the feasibility of this mode of operation, the vidicon target was illuminated by placing a red LED a few inches from the target. The vertical scan frequency was set at 25 kHz and the number of channels per frame was changed from 127 to 511. The results of this experiment are shown in Figure 6-8. 90% of the final steady state intensity was obtained within 4 to 5 frames. Similar results were obtained when the number of channels was changed from 511 to 127. The relatively rapid re-establishment of equilibrium indicates that this mode of operation would indeed be feasible, especially if the change in the resolution were not as drastic as that shown in Figure 6-8.

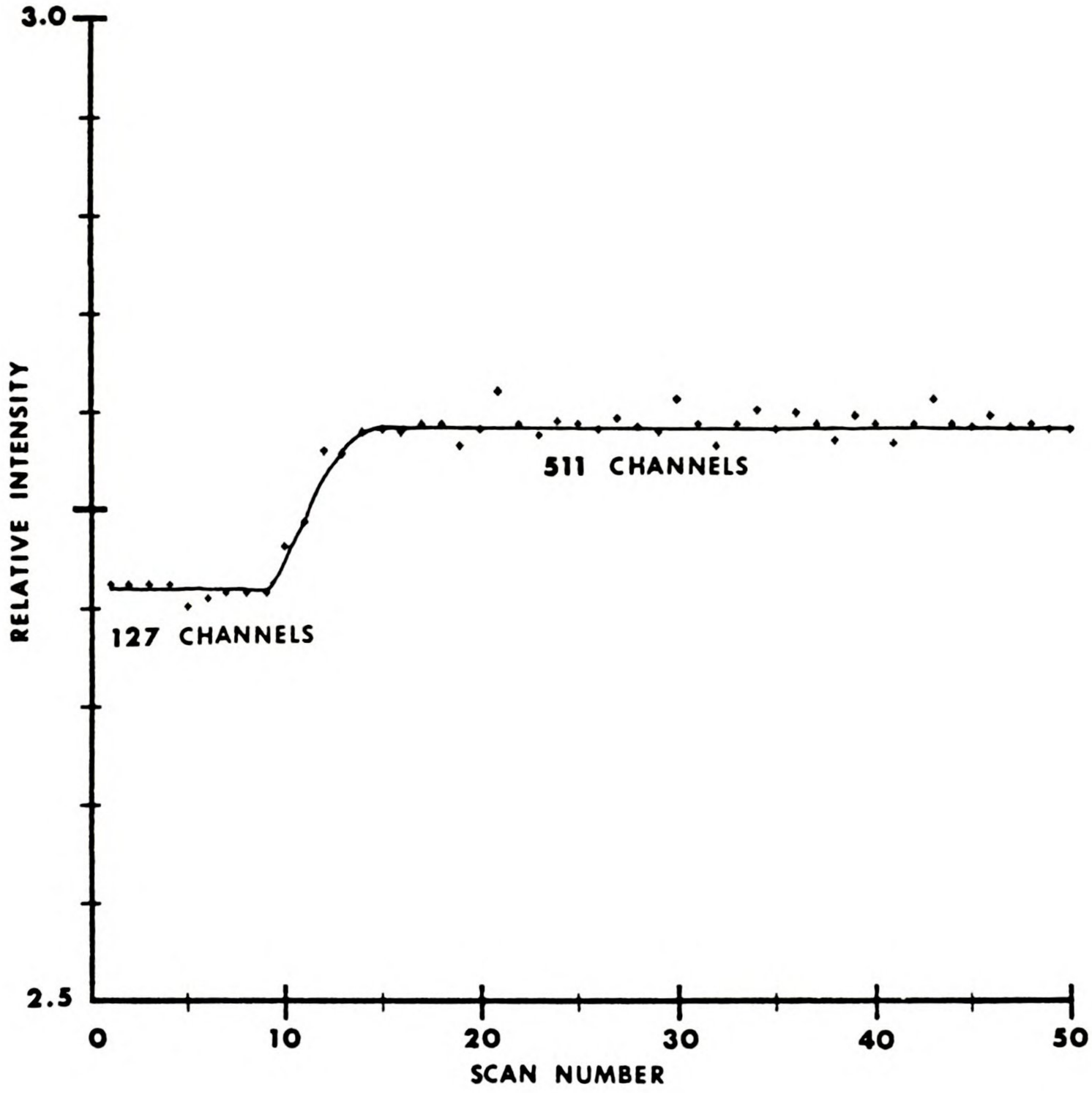


Figure 6-8. The Effect of Instantaneous Change in Resolution.

4. Electron Beam Current

The electron beam current is determined by the negative voltage which is applied to Grid 1 in the vidicon tube. The beam current is decreased by making the voltage on Grid 1 more negative. In the current system the grid voltage can be varied from about +4 V to -110 V. To determine the optimum beam current, the vidicon target was illuminated by a tungsten continuum, and the resolution was set at 511 channels per frame. The relationship between the intensity observed at the center of the target and the Grid 1 voltage at various vertical scan frequencies is shown in Figure 6-9. The rapid decrease in observed intensities at large grid voltages is due to the beam current being reduced to a level where it no longer suffices to read out all the charge stored on the target. As long as the beam current is larger than the minimum required to read out the charge, the observed intensity appears to be independent of beam current. If care is taken in selecting a beam current which is large enough to read out the largest signal expected, then the beam current should never have to be changed. This is quite fortunate, because even though the Datel AM301A high voltage amplifier used to generate the grid voltage is supposed to have a slew rate of 100 V/μsec, the actual observed slew rate of the grid voltage is only 15 V/sec.

5. Charge Integration

Charge integration is a signal enhancement technique which takes advantage of the integrating character of the silicon vidicon. In

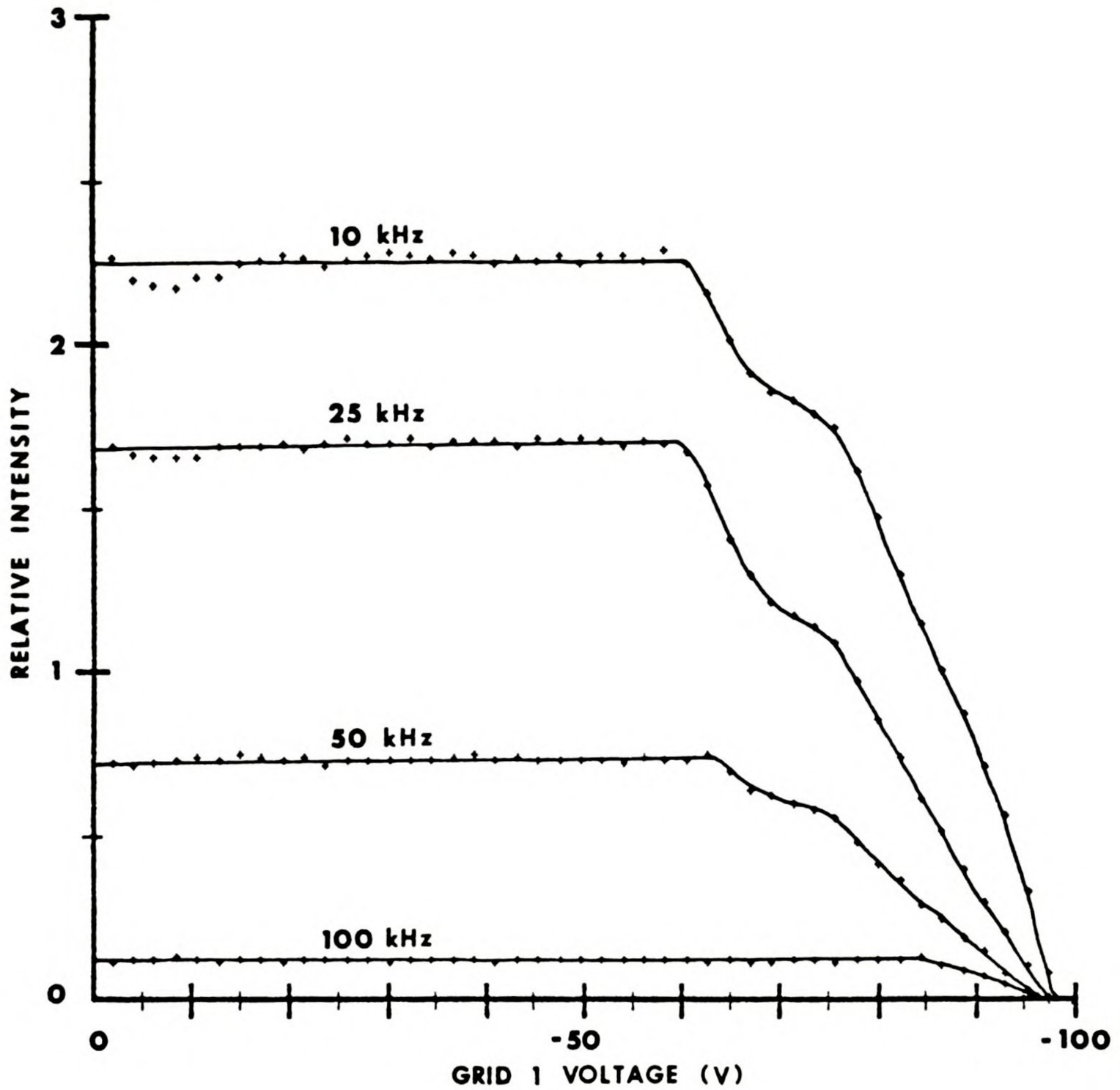


Figure 6-9. Relative Intensity Versus Grid 1 Voltage at Various Frequencies.

charge integration the electron beam is turned off for a number of scans and the photon-generated charge is allowed to accumulate. When the accumulated charge is finally read out an increase in the signal is observed which is linearly related to the integration time. Since the S/N (signal-to-noise ratio) increases linearly with the number of scans integrated, while the S/N increases only with the square root of the number of scans averaged, charge integration can be a very useful signal enhancement technique (38, 43, 60).

Signal enhancement by charge integration could be quite useful in kinetics experiments. If a constant integration time was used throughout a reaction, the signal level for the entire reaction could be increased. Even greater benefit could be derived if the integration time could be varied during the course of a reaction. For example, if the percent transmittance decreased dramatically during the course of a reaction, the signal level could be increased by using longer charge integration times. In order to determine if integration times could be rapidly changed, the following experiment was performed. The vidicon target was illuminated by a tungsten continuum and the resulting charge was read out at a vertical scan frequency of 25 kHz with a resolution of 101 channels per frame. The number of frames during which the beam was turned off was varied from 0 to 7. The results are shown in Figure 6-10. It is painfully apparent that integration times cannot be rapidly changed. At least 15 to 20 frames were required to reach 90% of the final steady state value. These results indicate that only constant integration times can be used for kinetics applications.

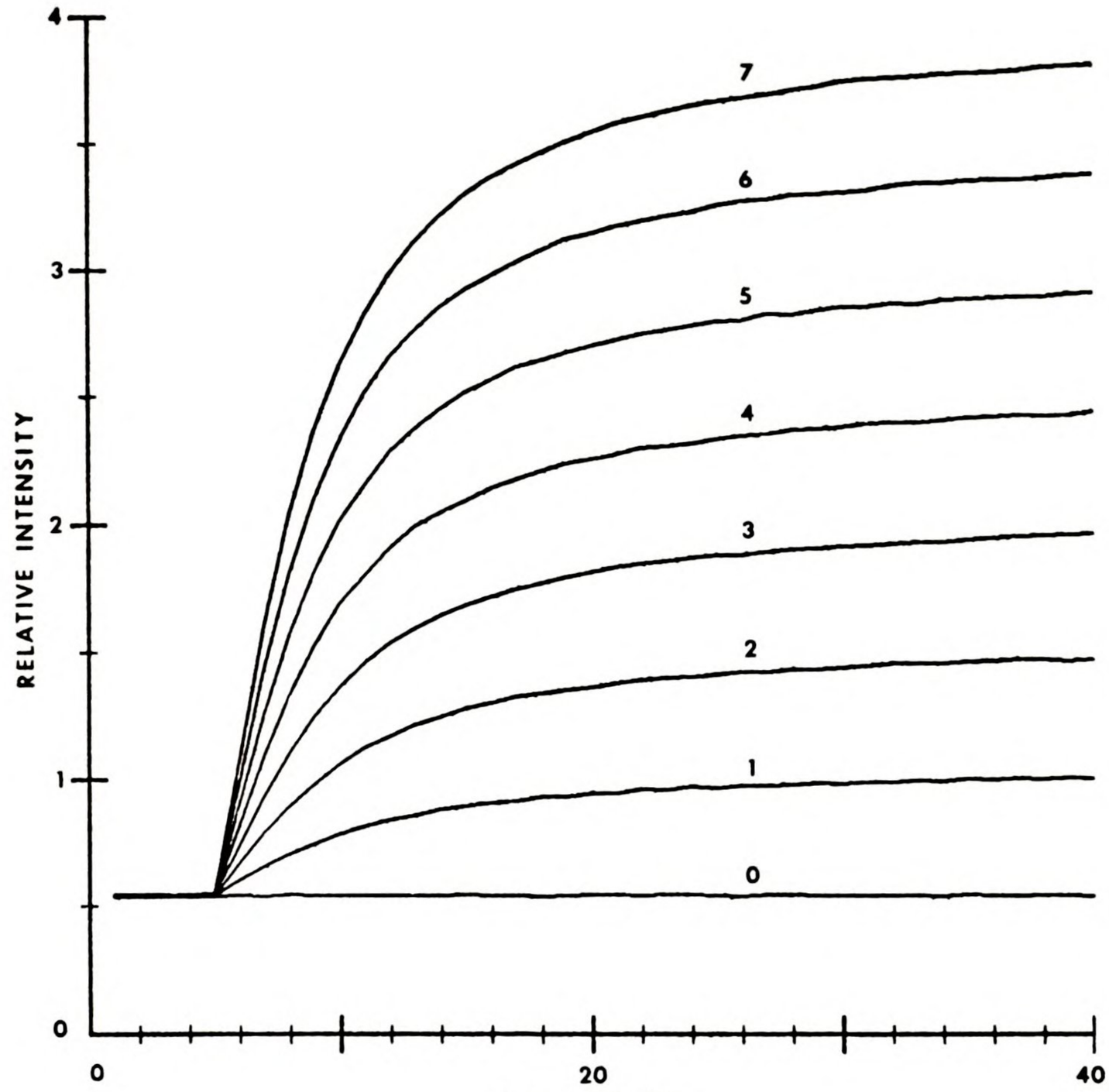


Figure 6-10. Effect of Rapid Changes in Charge Integration at Various Integration Levels.

6. Lag

As the electron beam travels across the vidicon target not all of the charge stored there is read out. The percentage of the total charge which is left behind is defined as the lag. Lag seriously degrades the response time of the vidicon detector and, therefore, it is the feature which most severely limits the applicability of vidicon detectors to fast kinetics. This section contains the results of a number of experiments which were designed to assess the severity of the lag in the vidicon spectrophotometer and to determine optimum conditions for limiting the effect of the lag.

Since it is virtually impossible to actually measure the amount of charge left behind on the target, techniques must be developed which provide an indication of the amount of charge left behind. This is most commonly done by continously scanning the vidicon target under illumination, then rapidly removing the illumination and observing the signal obtained after one or more complete frames have been scanned. All of the lag experiments reported in this section were performed using the same procedure. First the target was illuminated by placing a red LED a few inches from the target. A number of frames were scanned with the LED turned on. The LED was then turned off and the signal decay was observed until a constant signal was obtained. The lag was then defined in the following manner:

$$\% \text{ Lag} = \frac{S_1 - S_D}{S_{ILL} - S_D} \times 100$$

where S_1 is the signal measured one frame after the LED has been turned off, S_{ILL} is the average signal obtained while the LED is on and S_D is the average dark signal. Figure 6-11 shows a sample lag decay curve.

Comparison of the lag measured in different laboratories is often extremely difficult due to the varying experimental conditions under which the lag was measured (19, 27, 38). Often the conditions are not mentioned at all. The most commonly published data are measured at standard television scan rates. The lag decay curve shown in Figure 6-11 was obtained with the scan parameters which most closely approximate those used for standard television rates. The vertical scan frequency was 15.87 kHz and the resolution was 272 channels per frame. All other parameters were optimized for minimum lag. The resulting lag was 12.0%. This is in good agreement with one group (27) which reported a lag of 14% under similar conditions. Another source (38) listed a lag of 7% after three readout frames, whereas our system shows only 4.9% lag after three readout frames.

The remainder of this section is devoted to studies of how the various scan parameters affect the observed lag. The purpose of these studies is to determine how the parameters can be varied to minimize the lag. The reproducibility of the lag over periods of up to 6 hours was found to be fairly good, with 1 to 5 percent relative standard deviations being typical. When these same lag measurements were made over a period of a few weeks, relative precisions as poor as 50% were observed. The poor reproducibility of

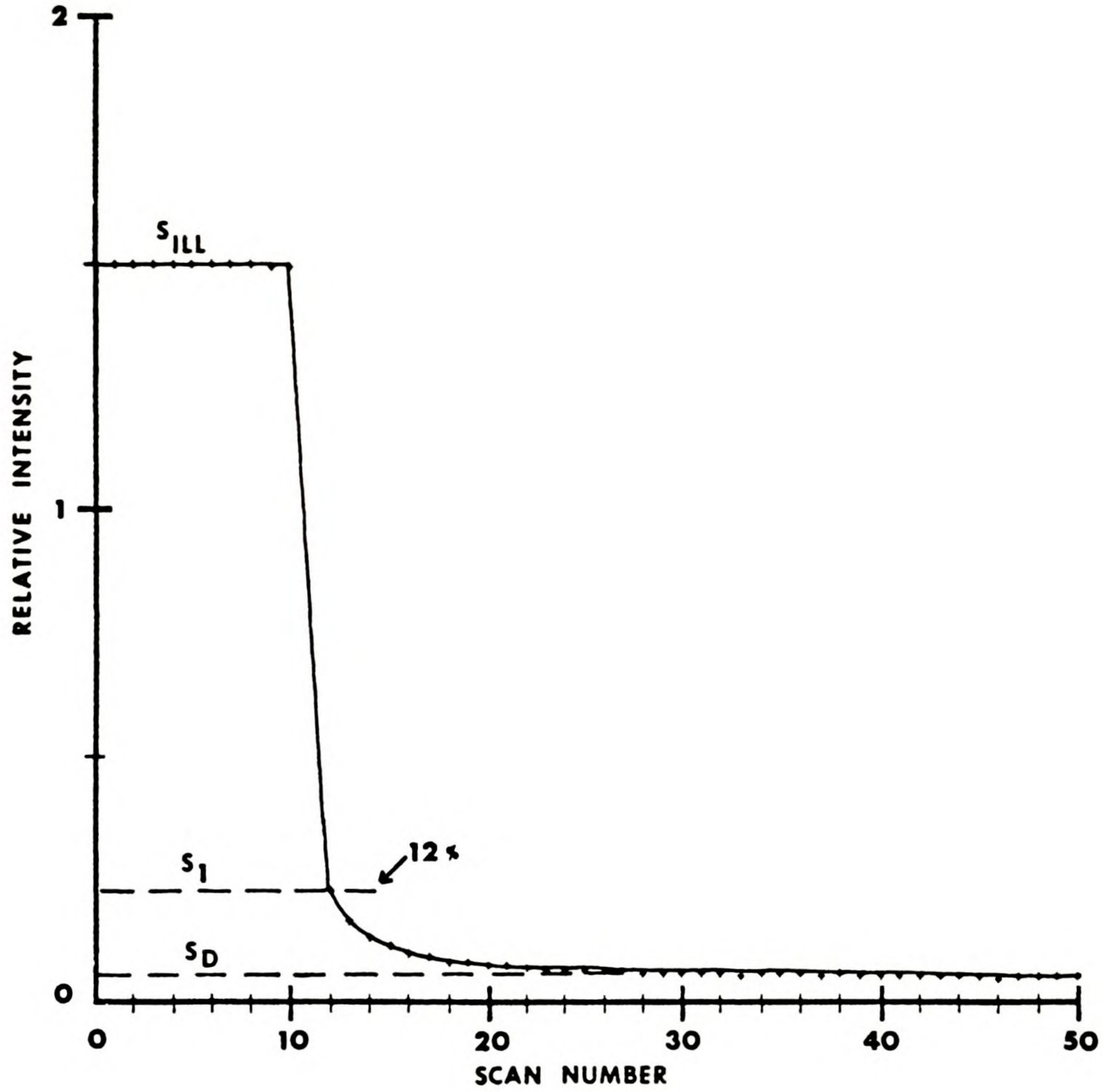


Figure 6-11. Signal Decay After Removal of Illumination.

the lag did not significantly affect the overall trends in the lag which are reported in this section. Since the actual lag values are not very reproducible, only the relative changes in the measured lag are reported in each of the following experiments.

Lag is dependent upon the magnitude of the signal read out from the vidicon target. Figure 6-12 shows the relationship between the relative lag and the signal magnitude. This experiment was performed at a vertical scan frequency of 25 kHz, a resolution of 255 channels per frame and a Grid 1 voltage of -23 V. Normally, lag tends to improve at higher signal levels. However, as the signal level approaches saturation, the lag increases rapidly. The increase in lag at low signal levels is due primarily to the target potential being discharged only slightly, resulting in a low potential difference between the cathode and the target. This results in a lowered beam landing efficiency and increased lag.

The resolution which is selected also appears to have a significant effect upon the observed lag. Figure 6-13 shows the relationship between relative lag and resolution at a vertical scan frequency of 25 kHz and a Grid 1 voltage of -23 V. The decrease in lag at higher resolutions is due primarily to the decrease in the channel separation. Decreasing the channel separation decreases the saturated area between the channels, therefore decreasing the relative lag.

The vertical scan frequency has a dramatic effect upon the observed lag. Figure 6-14 shows the frequency dependence of the relative lag at a resolution of 255 channels per frame and a Grid 1

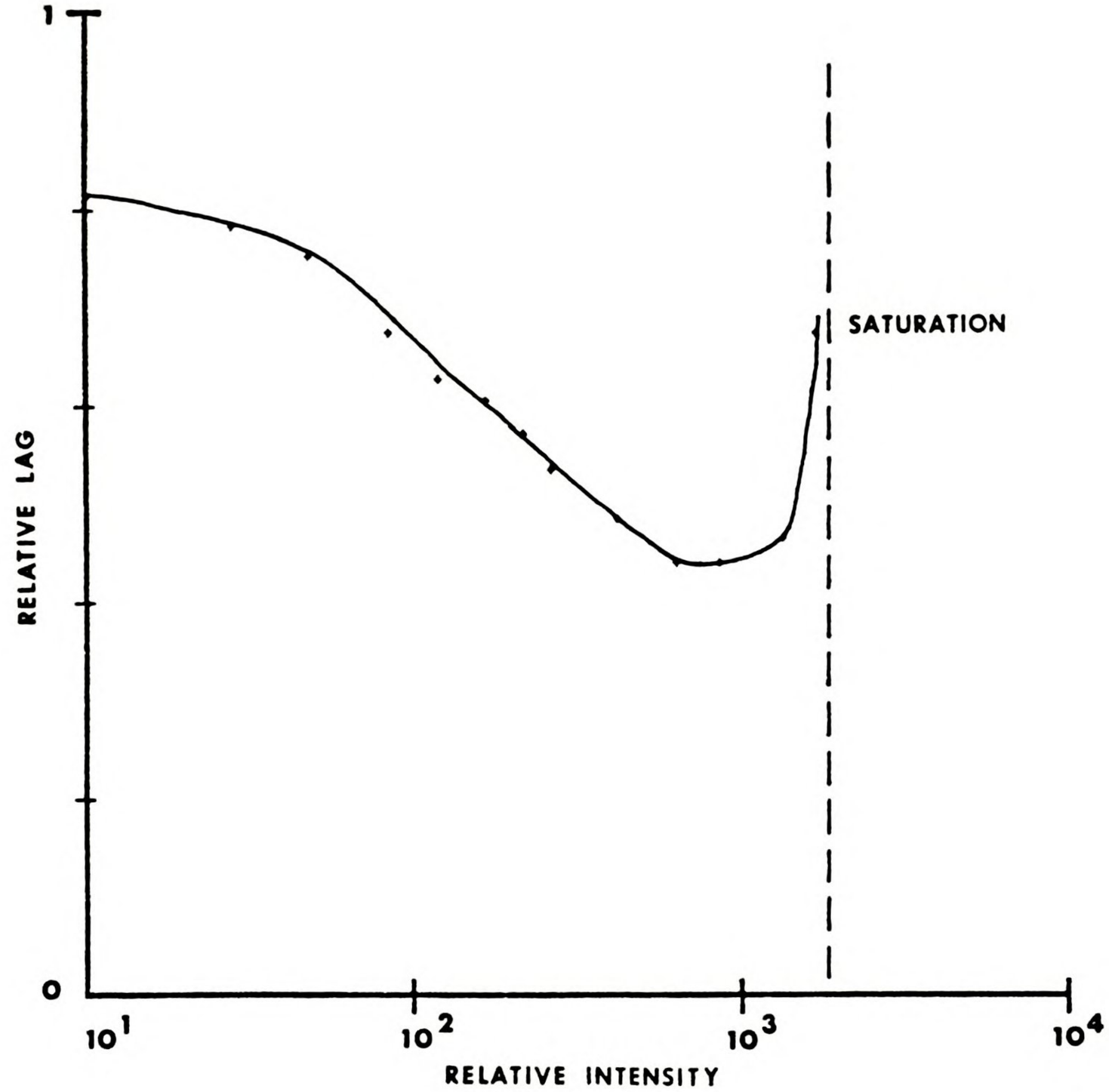


Figure 6-12. Relative Lag Versus Relative Intensity.

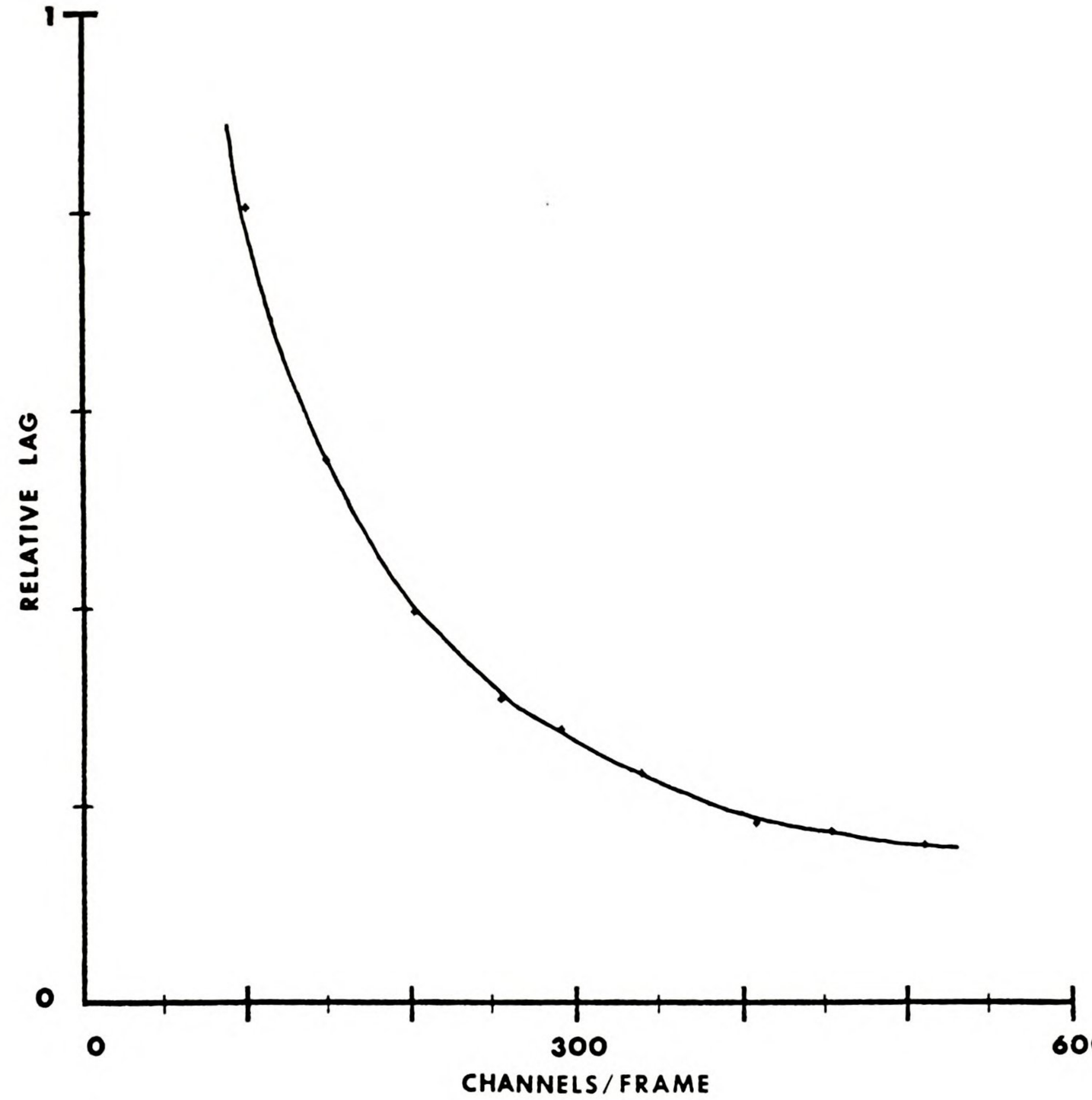


Figure 6-13. Relative Lag Versus Resolution (Frequency = 25 kHz).

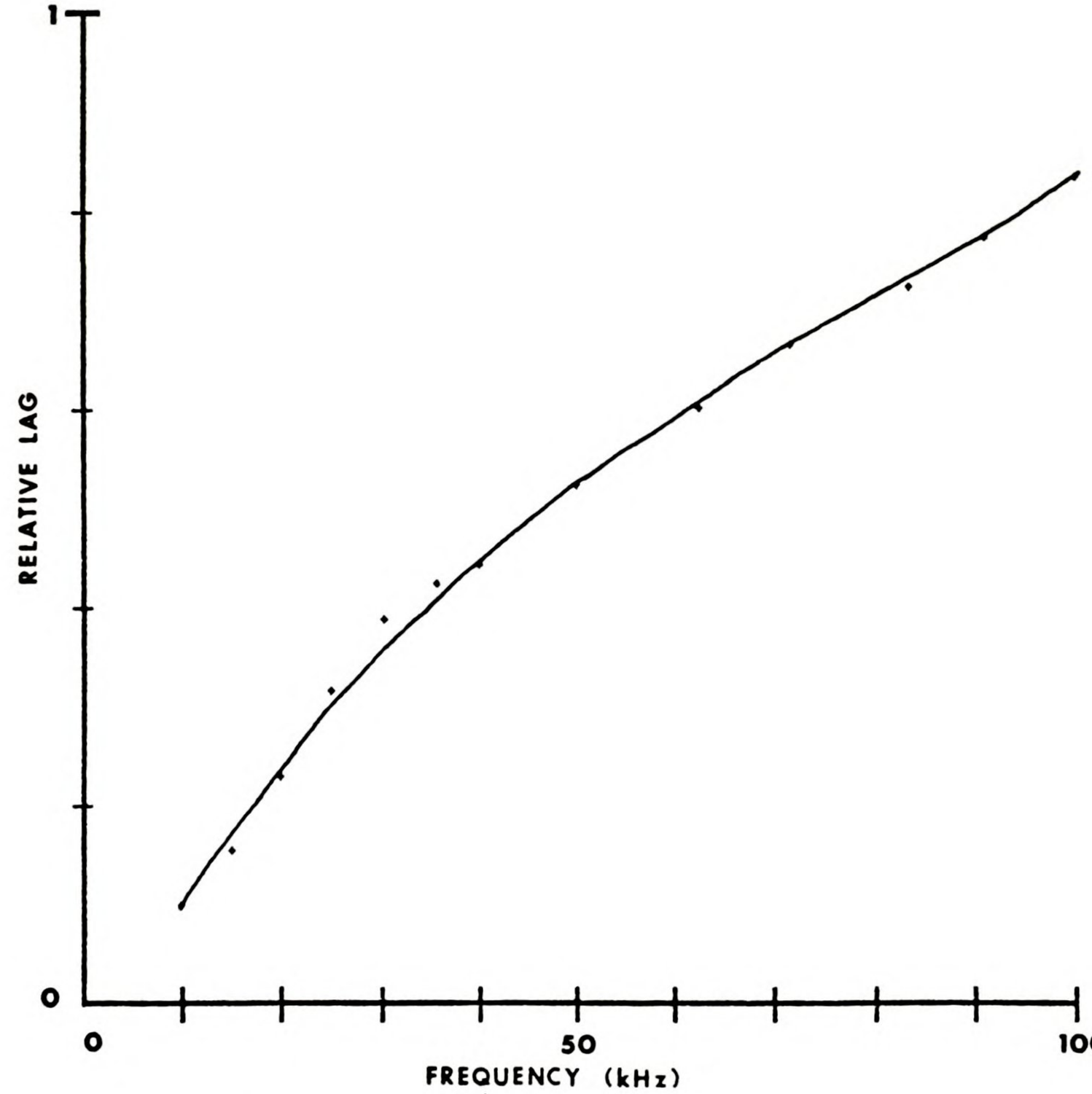


Figure 6-14. Relative Lag Versus Vertical Scan Frequency (Resolution = 255 Channels/Frame).

voltage of -23 V. The increase in the lag at higher vertical scan frequencies is due to a number of factors. Since the observed intensity decreases rapidly at higher frequencies, the intensity dependence of the lag results in an increase in the observed lag. The beam residency time decreases with increasing frequency, also causing an increase in the lag. Another cause of the increased lag at high frequencies is the incomplete scanning of the upper and lower edges of the target. As shown in Figure 6-3 the amplitude of the vertical scan varies significantly with frequency. The edges of the target are not scanned because it would be impractical to increase the electron beam; rectangular scanning pattern to cover the entire circular, active region of the target. Since the edges are in saturation, the observed lag is increased in the scanned portions of the target near the upper and lower edges. A limited amount of vertical resolution is possible in the vidicon spectrophotometer by varying the electronic delay and integration times in the amplifier chain (43). Figure 6-15 shows the dependence of the lag on the vertical target position at a resolution of 255 channels per frame. The vertical scan frequency was set to 25 kHz at which the vertical drive amplitude is near maximum. The edges of the target do show a significantly higher lag than the center of the target. (The very high lag at the left edge of the graph is due to the inclusion of the vertical retrace.)

The last parameter which was found to significantly affect lag is the beam current. Figure 6-16 shows the relationship between the relative lag and the voltage applied to Grid 1 (the higher the

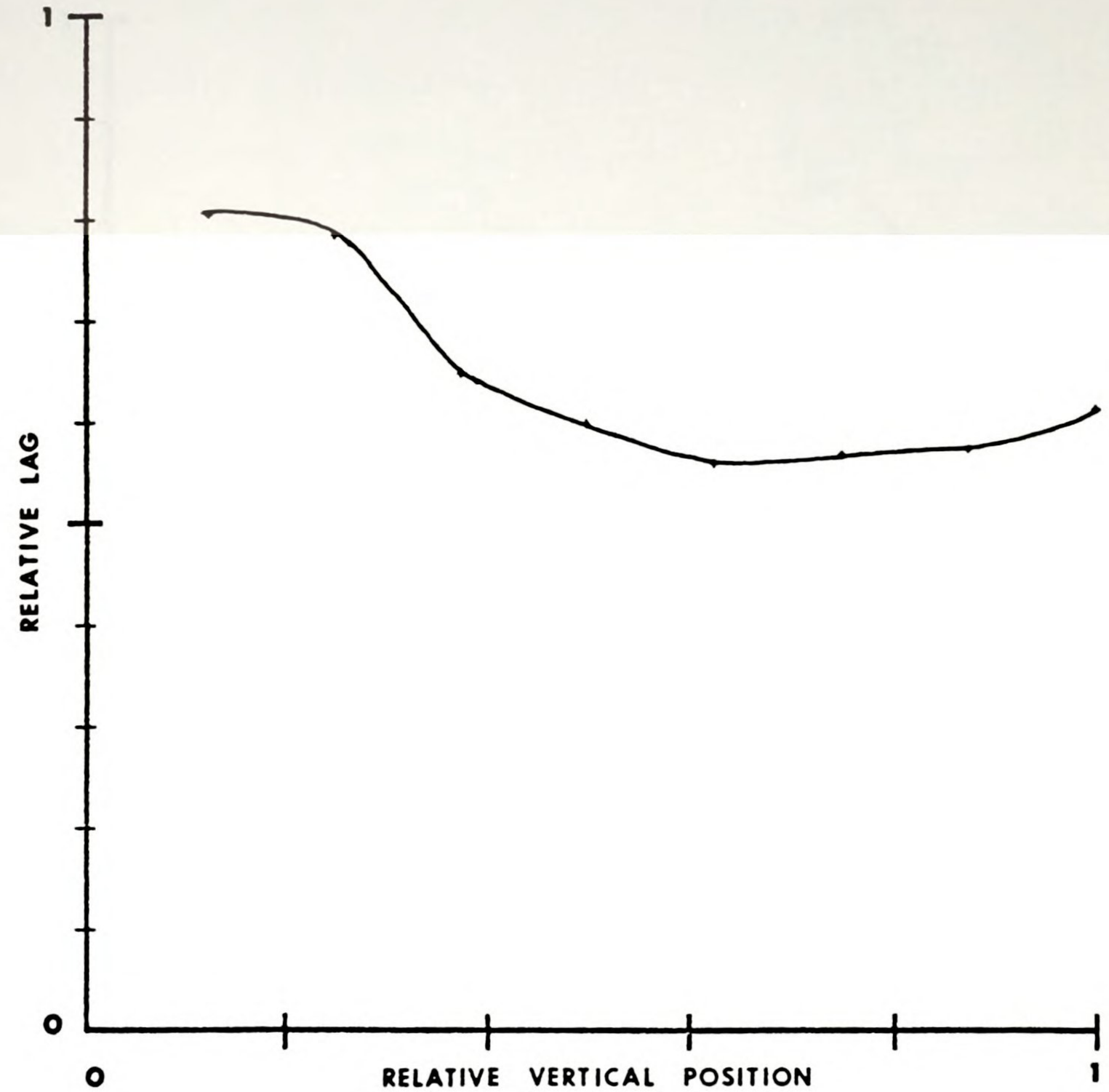


Figure 6-15. Relative Lag Versus Relative Vertical Target Position.

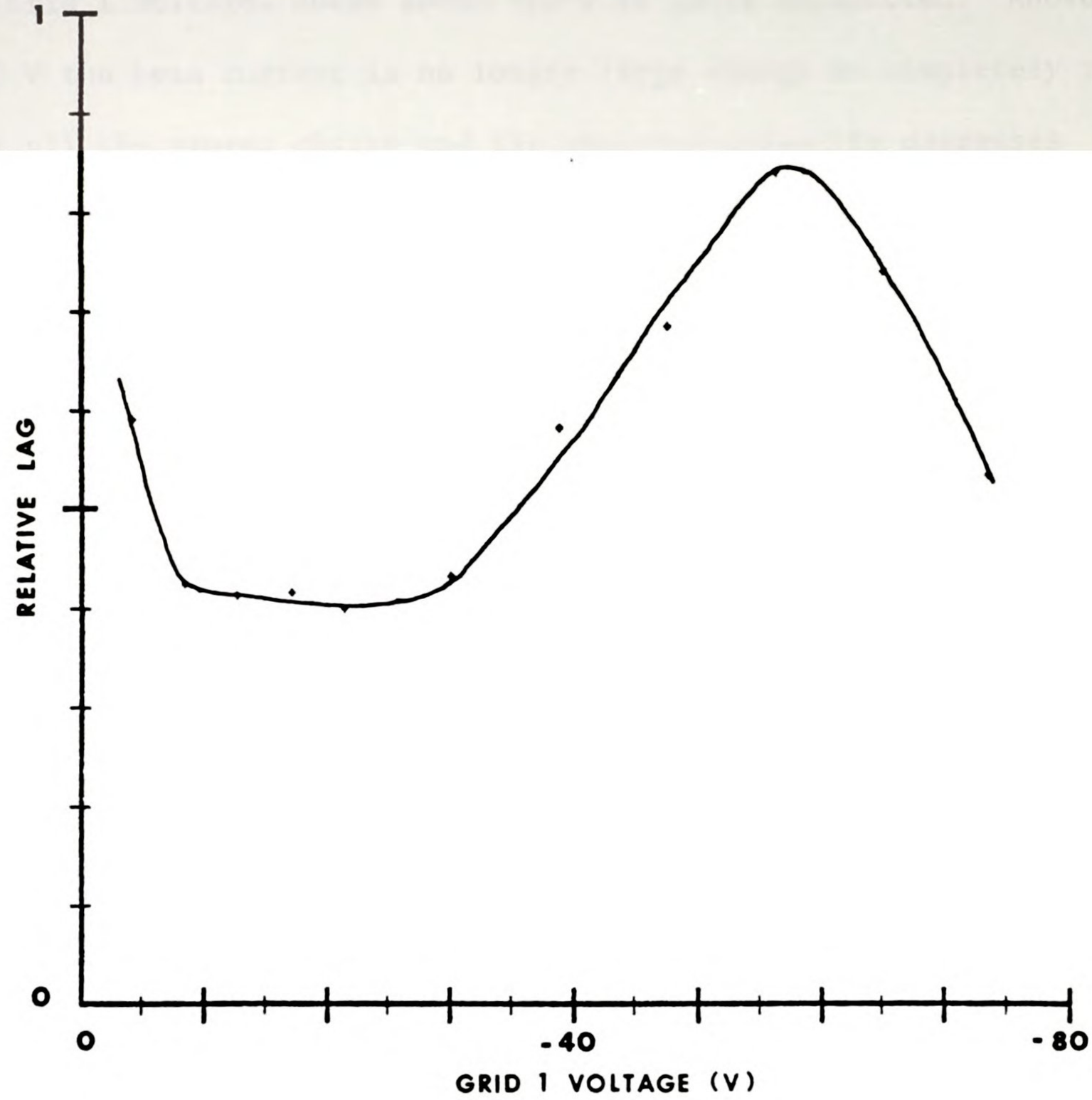


Figure 6-16. Relative Lag Versus Grid 1 Voltage.

voltage, the lower the beam current). This experiment was performed at a vertical scan frequency of 25 kHz and a resolution of 101 channels per frame. The lag is seen to reach a broad minimum at a Grid 1 voltage between about -10 V and -30 V. The decrease in lag at Grid 1 voltages above about -60 V is quite unexpected. Above -60 V the beam current is no longer large enough to completely read out all the stored charge and the observed intensity decreases rapidly (see Figure 6-9). Both of these factors should result in a dramatic increase in the lag. No matter what the reason is for the observed decrease in lag, the signal degradation at these voltages makes operation under these conditions impractical.

In this section it has been demonstrated that although lag is a serious problem in the vidicon spectrophotometer, its effects can be minimized by the careful selection of the scanning conditions. In general, the lag can be decreased by setting the Grid 1 voltage to about -18 V, decreasing the vertical scan frequency and/or increasing the number of channels per scan. Adjustment of the experimental conditions so that the signal intensity is about 50% of saturation will also decrease the observed lag.

C. Characteristics of the Stopped-Flow Mixing System

The purpose of the stopped-flow mixing system is to mix rapidly together two solutions and deliver them to an observation cell where any subsequent reactions can be monitored by the vidicon spectrophotometer. In order to be able to investigate relatively rapid reactions the mixing system must be capable of mixing the two

solutions both rapidly and reproducibly. The mixing characteristics and other important characteristics of the stopped-flow mixing system developed in this work will be presented in this section.

The stopped-flow mixing characteristics are highly dependent upon the solution flow velocity. In the current system, the solution flow velocity is monitored by a transducer which consists of a transparent ruling (with 50 lines/inch) mounted on the stopping syringe, and an opto-interrupter module. (A complete description of this transducer can be found in Chapter 5.) The transducer allows the measurement of the length of time in μsec, between the lines as they pass the opto-interrupter. Figure 6-17 shows the flow velocity output for 5 separate stopped-flow pushes. Line number 47 represents the point at which the flow comes to a stop. The erratic output observed after this line, is due to mechanical vibrations in the flow transducer. (Further experiments showed that no vibration or cavitation effects occured in the flow cell.) Curve A of Figure 6-18 shows the distance traveled by the stopping syringe during a typical run. The entire distance is covered in about 0.03 s. Curve B of Figure 6-18 shows the syringe velocity profile. The velocity profile was calculated from the lower curve by performing a linear least squares fit to 4 points at a time. The slope of this tangent line was then taken as the velocity at the mid-point.

The single most important characteristic of a stopped-flow mixing system is its dead time. The dead time can be defined as the length of time between the theoretical start of a reaction and the earliest time at which the reaction can be monitored. In a properly

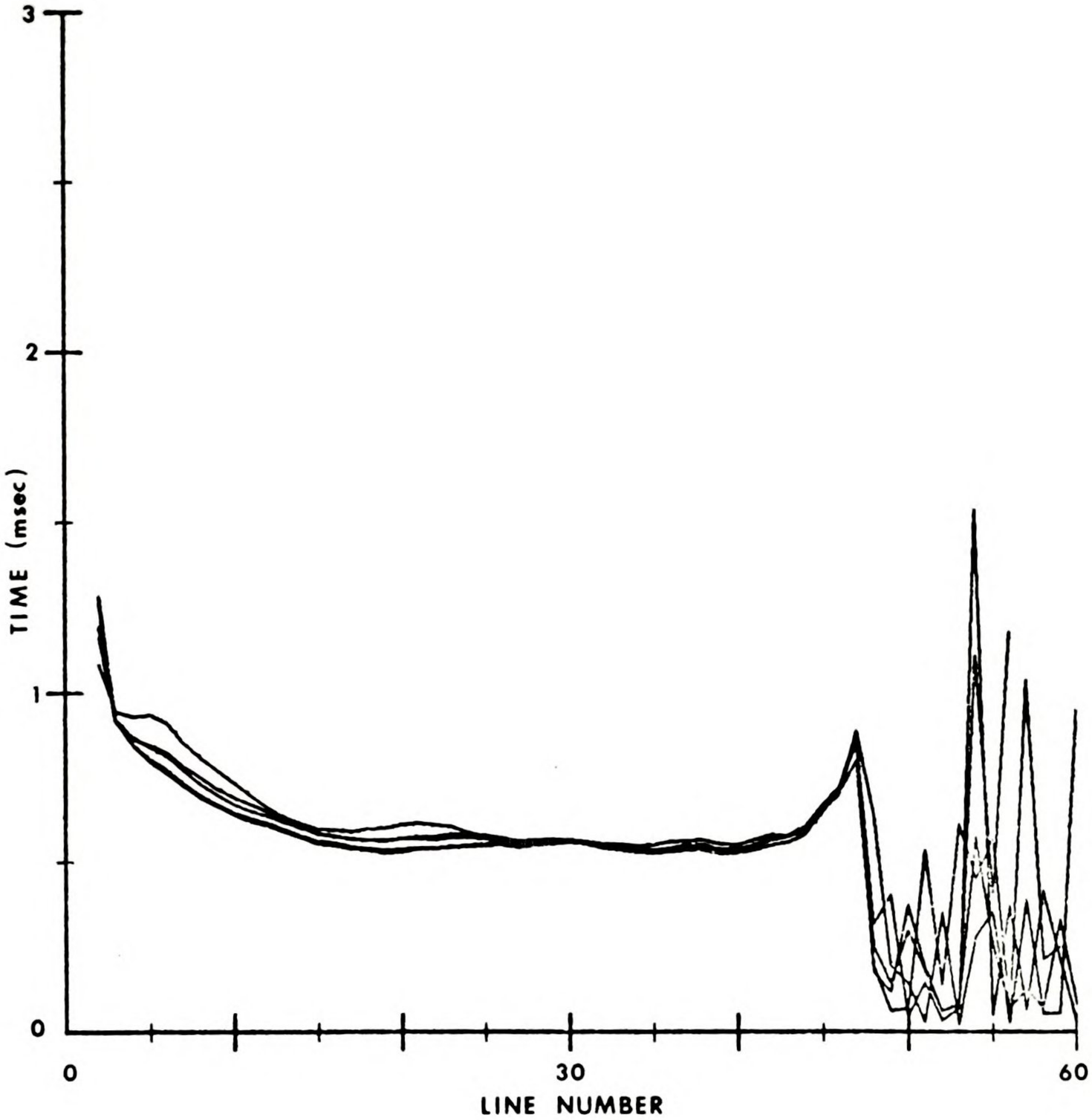


Figure 6-17. Flow Velocity Output for 5 Separate Runs.

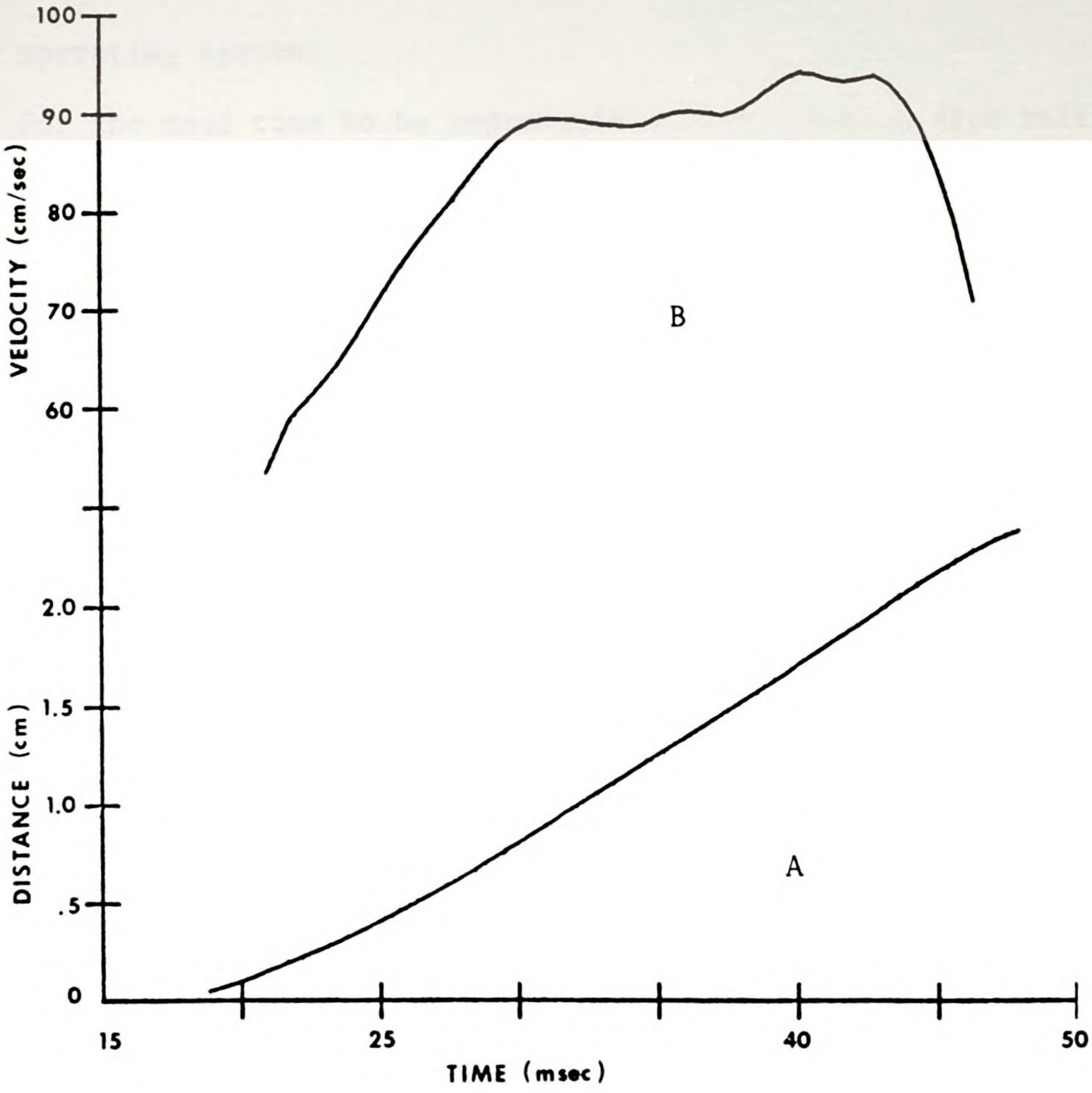


Figure 6-18. A - Syringe Distance Traveled Versus Time.
B - Syringe Velocity Versus Time.

operating stopped-flow, dead time is determined primarily by the length of time required for the solution to flow from the point of first mixing to the center of the observation cell. The actual dead time can be longer than this due to slow mixing, cavitation and other problems; however these problems should be minimal in a properly operating system.

For the dead time to be reproducible, the solution flow rate must be reproducible during the final portion of the flow. Figure 6-17 shows that the flow profile is quite reproducible from about line number 26 to the end of the flow. This corresponds to about the last 12 msec of the solution flow. The average syringe velocity over this section is 87.8 ± 0.9 cm/s, which yields a solution flow rate of 13.6 ± 0.1 ml/s with the current stopping syringe. This means that 0.163 ± 0.002 ml of solution flow through the cell during this time. Since the combined volume of the mixing chamber and the observation cell is 0.134 ± 0.0007 ml, the entire observation cell should be completely flushed during the most reproducible portion of the flow. This should result in a very reproducible dead time.

The dead time of the system was investigated by two methods. The length of time required for the solution to flow from the point of first mixing to the center of the observation cell can be determined by the following equation:

$$t_d = \frac{V_d}{v_s A_s}$$

where t_d is the dead time, V_d is the dead volume (the volume between

the point of first mixing and the center of the cell), v_s is the syringe velocity and A_s is the cross sectional area of the syringe. The value of v_s used should be the average syringe velocity just prior to the stopping of the flow. It should be measured over a time approximately equal to the dead time. The average syringe velocity over this period was 85.4 ± 1.1 cm/s. Calculations based on the cell specifications yielded a value of 0.103 ± 0.005 ml for V_d and 0.155 ± 0.002 cm^2 for A_s. This results in a dead time of 7.7 ± 0.4 msec. The uncertainty in the dead time is primarily due to the uncertainty in the dead volume. This uncertainty could be improved by determining a more precise value for V_d.

The above method for calculating the dead time does not include any effects from mixing, cavitation and other problems which can affect the performance of the stopped-flow mixing system. The traditional method for determining dead times is the extrapolation technique (61), whereby the total dead time including mixing and other problems is measured. In this technique the absorbance of a chemical reaction in the stopped-flow is recorded with time, normally on an oscilloscope. The experimental curve is then extrapolated back to zero absorbance. The dead time is the time between the extrapolated point of zero absorbance and the time at which the extrapolated and experimental curves coincide. This technique, with a photomultiplier tube employed as a detector was also used to determine the dead time. The dead time was found to be 8 ± 1 msec. The large uncertainty is due to the difficulty of performing the extrapolations. The difference between the two dead time values is not significant.

Two solutions were used to determine whether equal volumes are delivered from the two drive syringes. One syringe was filled with 0.0017 M $K_2Cr_2O_7$ in 0.2 M H_2SO_4 and the other syringe was filled with 0.2 M H_2SO_4. The absorbance of this mixture at 435 nm was determined, the solutions switched and the new absorbance determined. The absorbance which was found when the $K_2Cr_2O_7$ solution was in the left syringe was found to be larger than the absorbance when it was in the right syringe. The ratio of the absorbances was found to be 1.03 ± 0.02. The ratio of the volumes of the syringes is also equal to this value. Whenever quantitative measurements are performed with this stopped-flow apparatus, the above difference in syringe volumes must be compensated for in the calculations.

The pathlength of the stopped-flow observation cell was designed to be 2.0 cm. The actual pathlength was determined by comparing the absorbance of a solution in a cuvette having a calibrated pathlength and the same solution in the stopped-flow cell. The pathlength of the cell was found to be 1.96 ± 0.04 cm.

When solutions to be used in stopped-flow experiments are changed, the drive syringes must be flushed a number of times. The number of times that the syringes must be flushed to ensure complete removal of the old solutions varies with the type of solutions used. In most cases at least 30 to 40 flushes of the drive syringes are required. Since about 0.42 ml of solution is required for each flush, the total amount of solution required is about 12 to 17 ml.

D. The Monitoring of a Relatively Rapid Reaction

The capabilities of the entire system were determined by monitoring a relatively rapid reaction which has a large change in absorbance. The large change in absorbance causes a drastic change in the amount of light falling on the vidicon detector. If the lag of the vidicon is going to cause any errors when fast reactions are followed, these errors should show up in this type of application. The reaction which was selected was the iron-thiocyanate reaction (62), carried out with an excess of iron. Under these conditions the equilibrium equation involved is:

$$FeSO_4^{+} + SCN^{-} \underset{k_2}{\overset{k_1}{\rightleftharpoons}} Fe(SCN)^{2+} + SO_4^{2-}$$

and the rate law is (62):

$$\frac{d\,[Fe(SCN)^{2+}]}{dt} = k_1\,[FeSO_4^{+}]\,[SCN^{-}] - k_2\,[Fe(SCN)^{2+}].$$

A solution of 0.0800 M Fe^{3+} was mixed with a 0.0102 M SCN^{-} solution. The reaction was carried out in 0.02 M H_2SO_4. The vertical scan frequency for the vidicon spectrophotometer was set to 25 kHz, 102 vertical channels per scan were acquired, and the Grid 1 voltage was set to -21.5 V. The resulting kinetics data were evaluated by using a curve-fitting routine called KINFIT-4 (63) to determine the rate constant. This was done at two different wavelengths. At 583 nm the average forward rate constant (k) was found to be 96 M^{-1} sec^{-1}, while at 658 nm a value of 196 M^{-1} sec^{-1} was determined.

Previously reported values for this rate constant are 208 M^{-1} sec^{-1} (62), 228 M^{-1} sec^{-1} (54) and 212 M^{-1} sec^{-1} (36). The rate constant of 196 M^{-1} sec^{-1} is in about the right range; however, the value determined at 583 nm is far too low. The reason for this is lag problems in the vidicon detector. At 583 nm the light intensity from the tungsten source is much lower than at 658 nm. Also, under the conditions of this experiment the absorbance change at 583 nm is greater (total absorbance change of 1.68) than at 658 nm (total absorbance change of 0.31). Both of these factors result in a dramatic increase in the detector lag at 583 nm.

The results of this study indicate that although the stopped-flow system has a far slower response time than a standard single wavelength instrument, it can still be used to monitor fairly rapid reactions provided care is taken in selecting the conditions under which the reactions are studied. The light intensity falling on the vidicon detector should be kept as high as possible. Also the absorbance changes which are studied must be kept small.

CHAPTER 7

AN INVESTIGATION OF THE BIURET REACTION

A. Background

The biuret reaction is the basis for the standard clinical chemistry technique for determining the total protein content of blood serum (64), which can be a very useful aid in the diagnosis of a number of diseases. The normal total serum protein level in a healthy adult is between 6.0 and 8.5 g/100 ml (64). A number of diseases show symptoms of hypoproteinemia, in which the total serum protein level drops below 6.0 g/100 ml. An elevation in total serum protein level is encountered in multiple myeloma and in any disease which results in dehydration.

The biuret reaction is a complexation reaction between the cupric ion (Cu^{2+}) and the peptide bonds of the protein which occurs in alkaline solution. Although the exact structure of the complex is unknown, the Cu^{2+} is believed to interact with the carbonyl (-C=O) and imine (=N-H) groups of the peptide bond. The Cu^{2+} is normally linked to between four and six peptide bonds (64).

When the biuret reaction was first introduced, the procedure was relatively complex and the reagents were unstable, resulting in poor reproducibility. Kingsley (65) simplified the technique and made it reproducible enough for routine clinical use. Since that time a number of variations and improvements on the basic technique have appeared (66, 67). In the standard biuret technique, the reaction is allowed to go to completion and then the absorbance of the complex is

measured at 550 nm. This technique tends to be rather slow since the reaction, though very rapid over the first few seconds, does not go to completion for at least 20 to 30 minutes.

Koch, et al. (68) reported a kinetic method for determining total serum protein using the biuret reaction. In this study a centrifugal analyzer, the CentrifiChem Fast Analyzer (Union Carbide Corp.), was used. They found that reasonable results could be obtained by this method in as little as 18 s. They reported a precision of 2.9% for replicate runs and a 3.4% precision for day-to-day results. One limitation of this technique is that assayed human blood serum has to be used for standardizations. In the standard biuret technique, assayed bovine serum albumin can be used as a standard. Bovine albumin was found to react much more rapidly in the early part of the reaction than human serum. They also found that bovine albumin could be used as a standard if the analysis time was lengthened to at least 93 s. Savory, et al. (69) tried to use this kinetic method, but found it to be relatively imprecise and subject to analysis errors in patients with certain diseases.

Recently, Wai-Tak Law and S.R. Crouch of the Michigan State University Department of Chemistry have been investigating a more rapid rate method for total serum protein using the biuret reaction in a stopped-flow spectrophotometer (70). The reaction is monitored by following the growth of the absorbance of the copper-protein complex at 550 nm. Good results have been obtained in as little as 0.1 s. The average relative standard deviation for replicate runs was 3.1%; when the analysis time was increased to 10 s the precision improved

to 0.83%. The experimental rate curve for this reaction is very complex. After an induction period of about 20 msec, the reaction rate increases rapidly until it reaches a peak at just over 100 msec. After this peak the rate falls off irregularly over a long time period. Most of the reaction occurs within the first 30 s, but the reaction doesn't go to completion until after 20 to 30 minutes. Just as Koch had discovered earlier (68), bovine protein samples react more rapidly than human protein samples and therefore cannot be used as standards.

To gain a better understanding of the biuret reaction, Wai-Tak Law and I decided to investigate the reaction by using the vidicon rapid scanning stopped-flow spectrophotometer. The first goal of this investigation was to study the very unusual rate behavior of the biuret reaction with human serum protein. The second goal was to study the variations in the reaction rates observed in the biuret reaction with bovine serum proteins and human serum protein. We felt that these studies would help to characterize the new rate method and also serve as a basis for determining future directions in characterizing the new rate method.

B. Experimental

The biuret reagent used in this work was prepared by dissolving 24.0 g of sodium hydroxide, 4.0 g of sodium potassium tartrate and 0.5 g of potassium iodide in approximately 800 ml of water. When the reagents were completely dissolved, 1.2 g of cupric sulfate was added and the solution diluted to 1 liter. After standing for a few

hours, the solution was filtered. The sodium potassium tartrate is used to complex the copper and keep it in solution. The biuret blank reagent was prepared in the same way as the biuret reagent except that no cupric sulfate was added.

The various serum protein samples were prepared by making an approximately 1 to 30 dilution of the serum samples with 1.9% saline solution. The human serum samples were Moni-Trol I (Dade Division American Hospital Supply Corp.). The bovine serum protein samples were bovine serum albumin and bovine serum globulin (Sigma Chemical Co.).

The following vidicon scan parameters were used throughout this study: the vertical scan frequency was set at 25 kHz with 102 channels per scan frame. This results in a scan time of 4.08 msec/frame. The beam current was set to about -23 V. No charge integration was performed.

The following general procedure was used for all of the stopped-flow data presented in this chapter: the stopped-flow system was first flushed approximately 30 times with the appropriate serum sample and the biuret blank reagent. With the shutter open, 16 scans from the vidicon detector were averaged to yield the reference (100% transmittance) spectrum. The shutter was then closed and another 16 scans were averaged to form the dark current spectrum. Next, the blank reagent was replaced by the biuret reagent and the stopped-flow system flushed another 30 times. The system was always flushed 1 or 2 more times immediately before each stopped-flow run. The reaction was monitored by collecting successive spectra, each of which was the

average of 4 frames. A total of 25 frames were skipped between the acquisition of successive spectra. This results in the acquisition of one spectrum every 118.32 msec. A total of 40 spectra are acquired during each push. To gain better precision, four separate pushes were always averaged together.

The reaction was initially investigated by centering the wavelength window of the vidicon detector at the absorbance maximum of the copper-protein complex (550 nm). It was soon discovered that an isobestic point existed at about 690 nm. The wavelength window was then moved to cover the region from about 550 nm to 850 nm.

Figure 7-1 shows selected spectra from the biuret reaction with human serum protein. Two isobestic points are observed, at about 690 nm and 802 nm. The sharp irregularities that occur periodically in the spectra (e.g. 640 nm, 730 nm, 810 nm, etc.) are due to slight shifts in the vidicon's electron beam retrace pattern which are not completely corrected for when the absorbances are calculated.

Figure 7-2 and 7-3 show the absorbance change with time which occurs at wavelengths of 575 nm, 752 nm and 839 nm (corresponding to the vertical lines in Figure 7-1).

Figure 7-4 shows selected spectra from the biuret reaction with bovine serum albumin. In this case only a single isobestic point was found, at 690 nm. For this reason, the reaction profile has a dramatically different appearance from the human serum. Figure 7-5 shows the absorbance changes which occur at 586 nm (a) and 779 nm (b).

The final serum sample investigated with the biuret reaction was bovine serum globulin; the results are shown in Figure 7-6. The

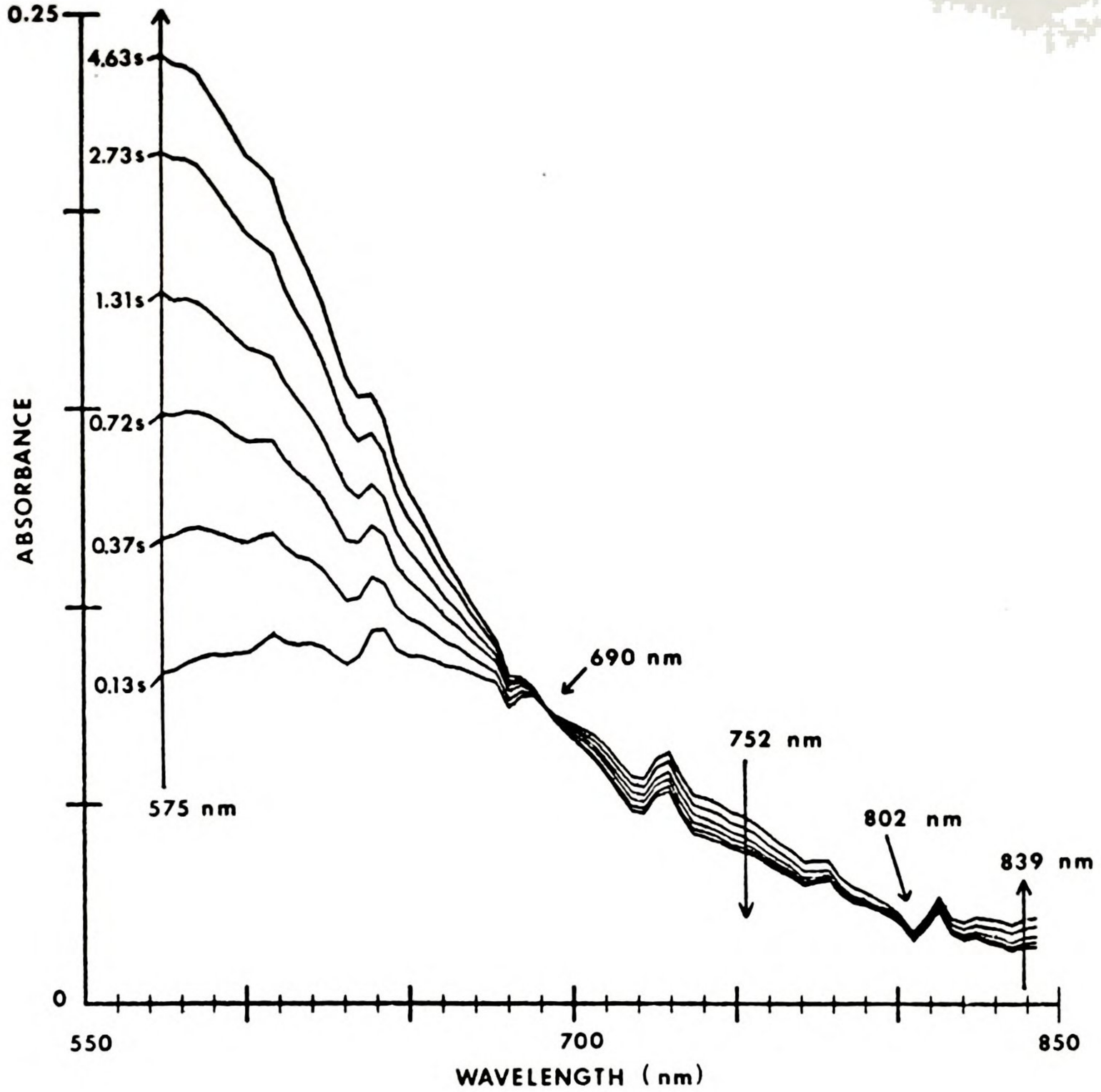
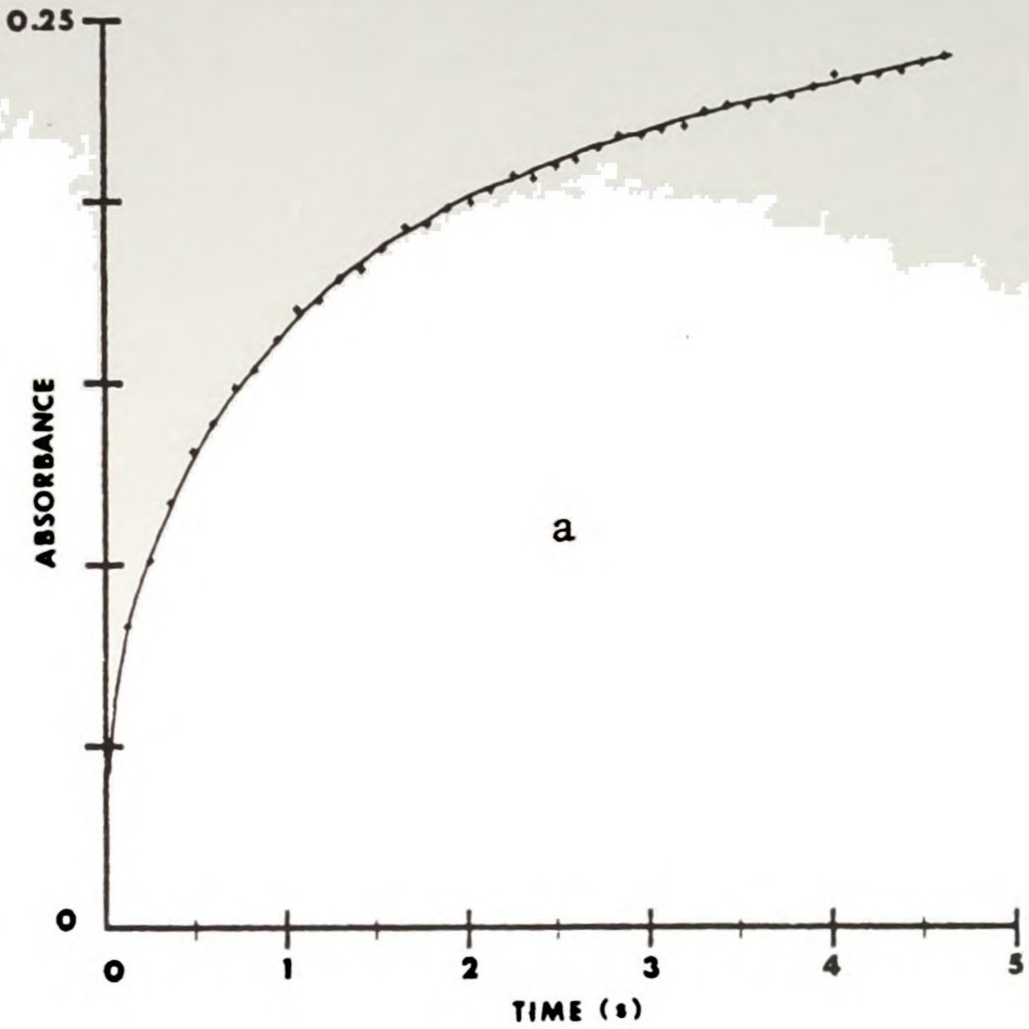
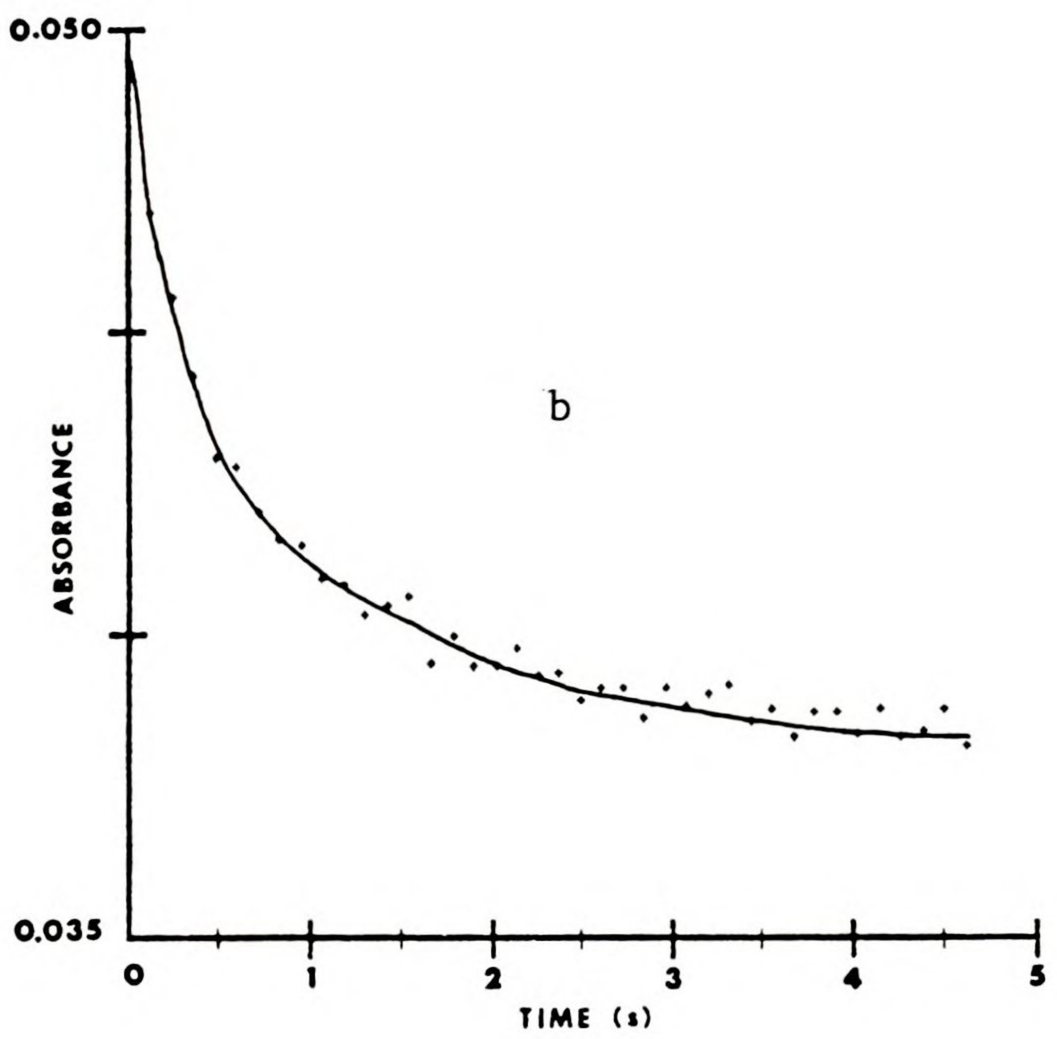


Figure 7-1. Absorption Spectra at Various Times During the Biuret Reaction with Human Serum (2.2 g/L).



a



b

Figure 7-2. Absorbance Versus Time for the Biuret Reaction with Human Serum. (a) 575 nm; (b) 752 nm.

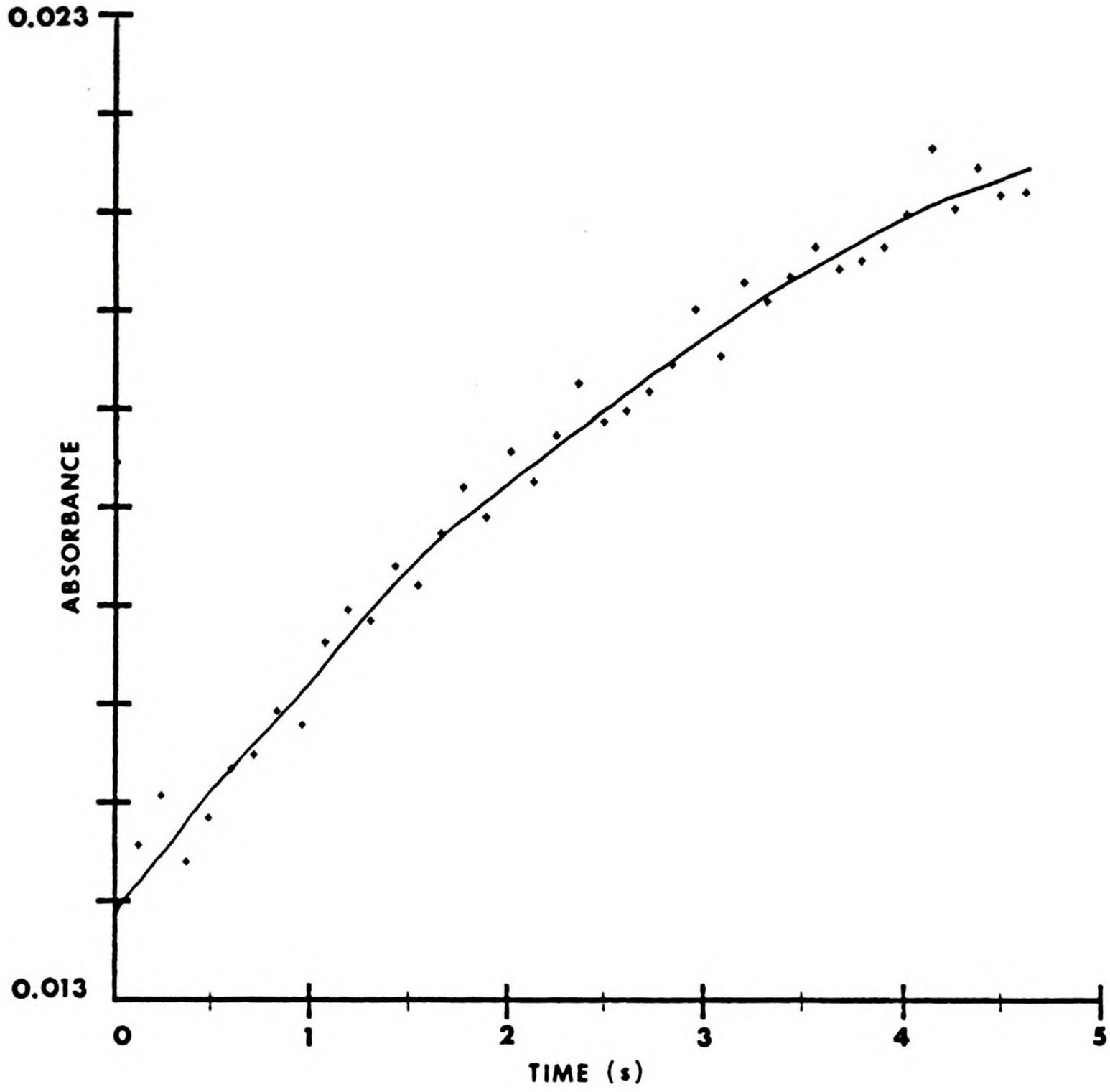


Figure 7-3. Absorbance (839 nm) Versus Time for the Biuret Reaction with Human Serum.

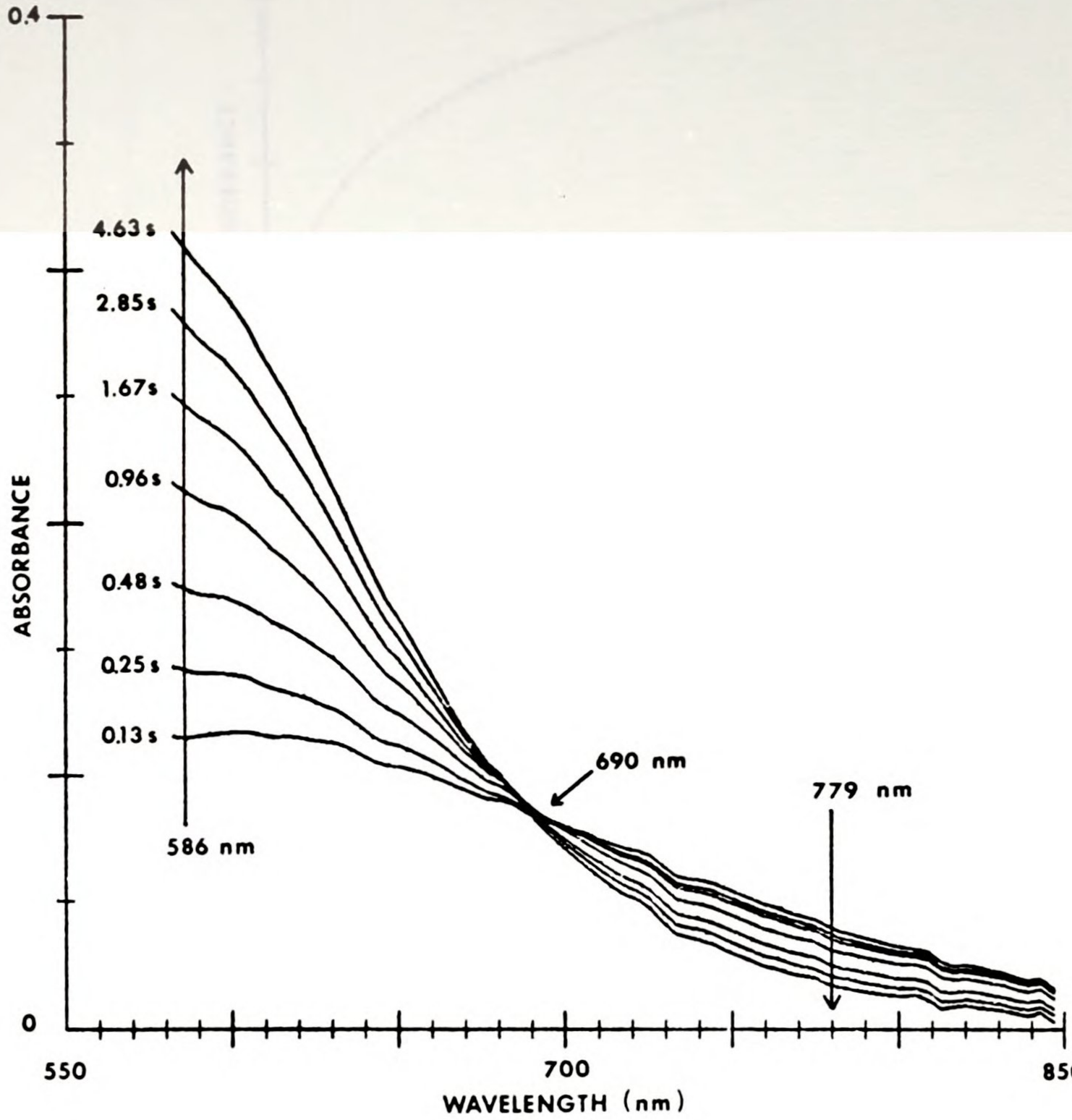
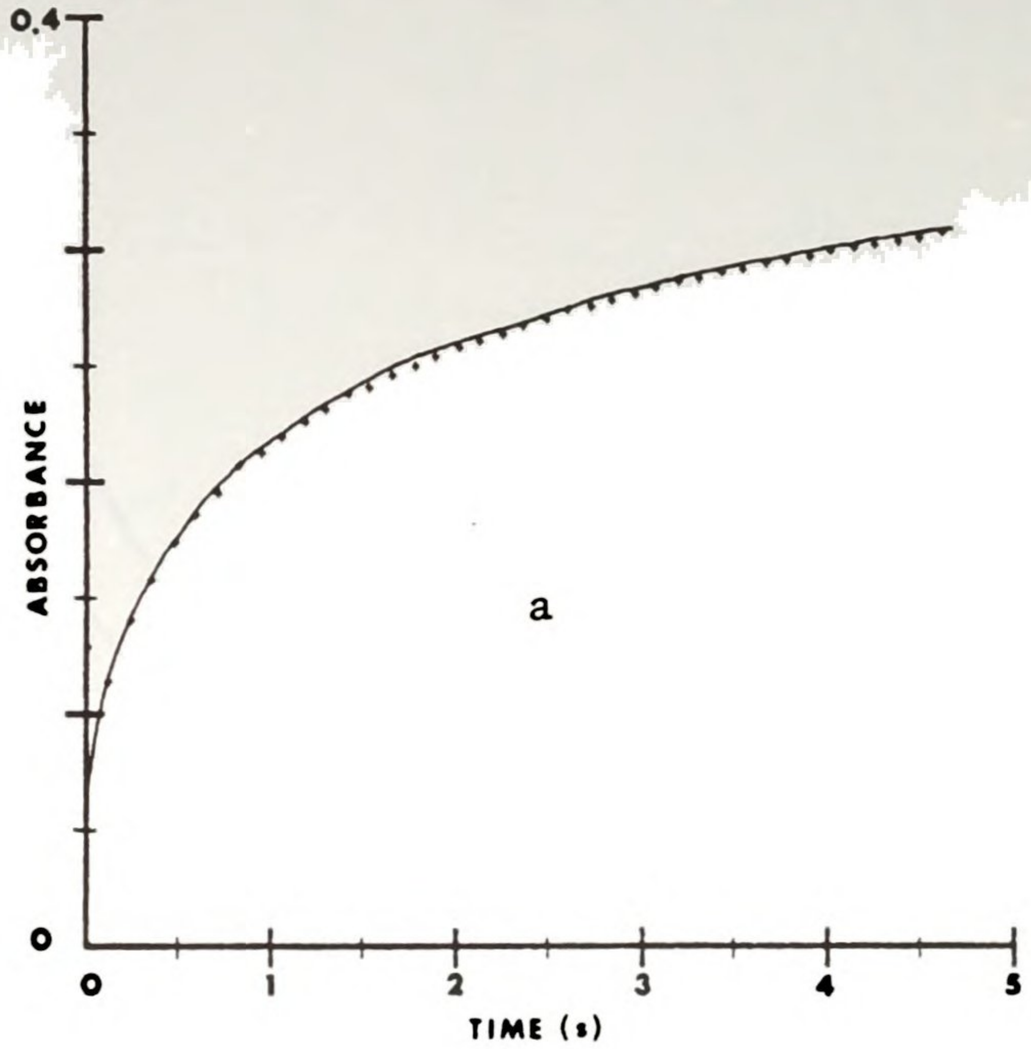


Figure 7-4. Absorption Spectra at Various Times During the Biuret Reaction with Bovine Serum Albumin (3.3 g/L).



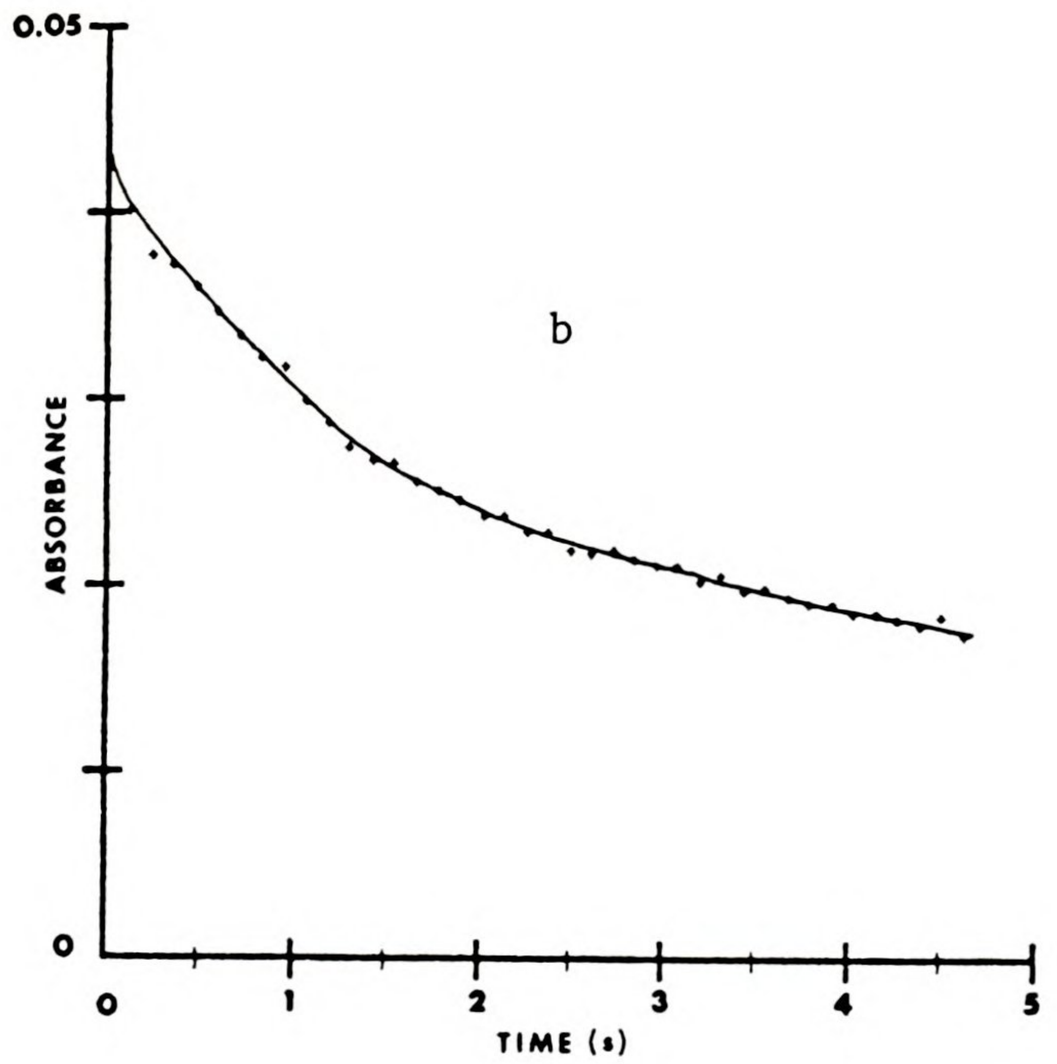


Figure 7-5. Absorbance Versus Time for the Biuret Reaction with Bovine Serum Albumin. (a) 586 nm; (b) 779 nm.

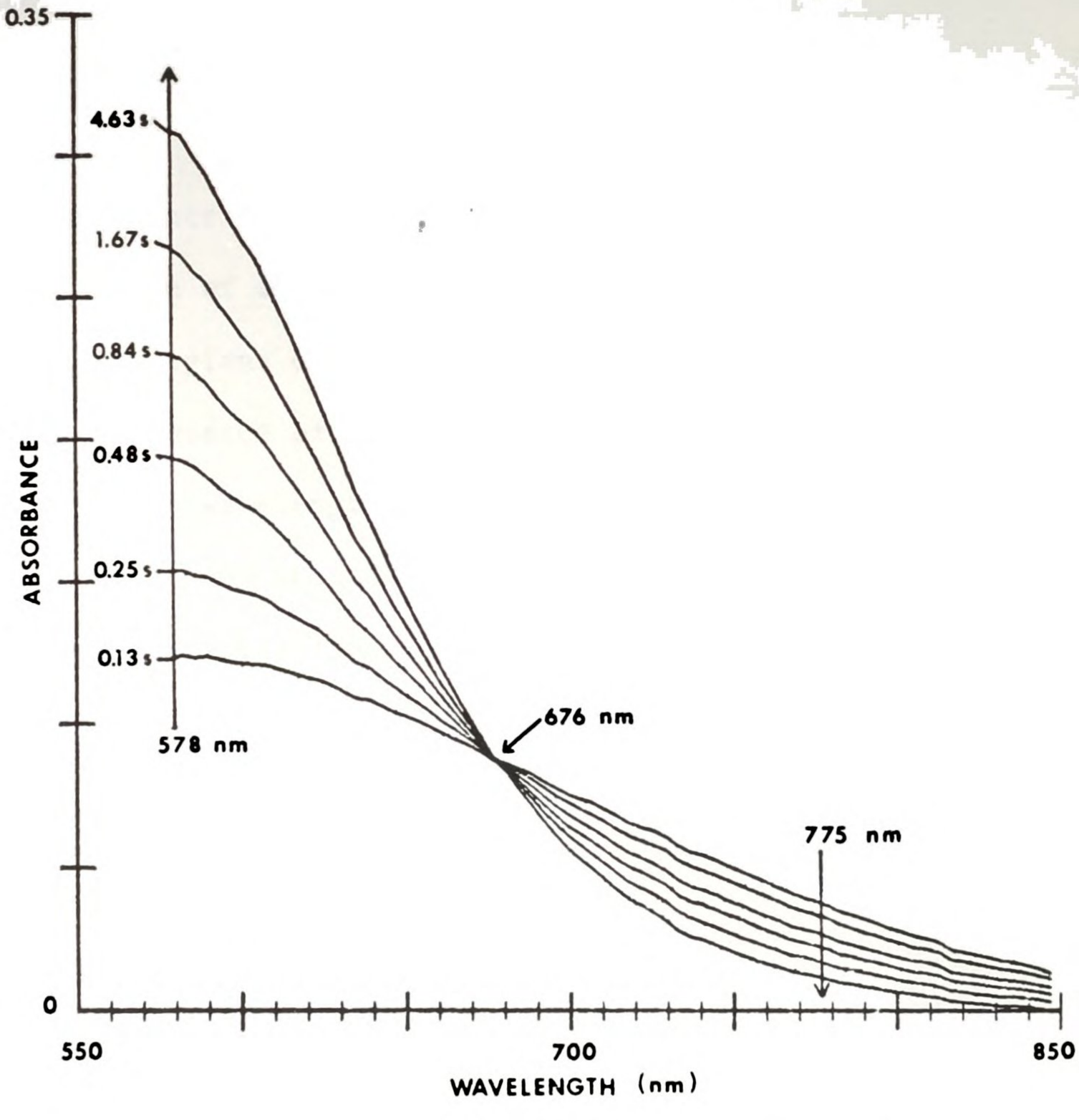


Figure 7-6. Absorption Spectra at Various Times During the Biuret Reaction with Bovine Serum Globulin (3.3 g/L).

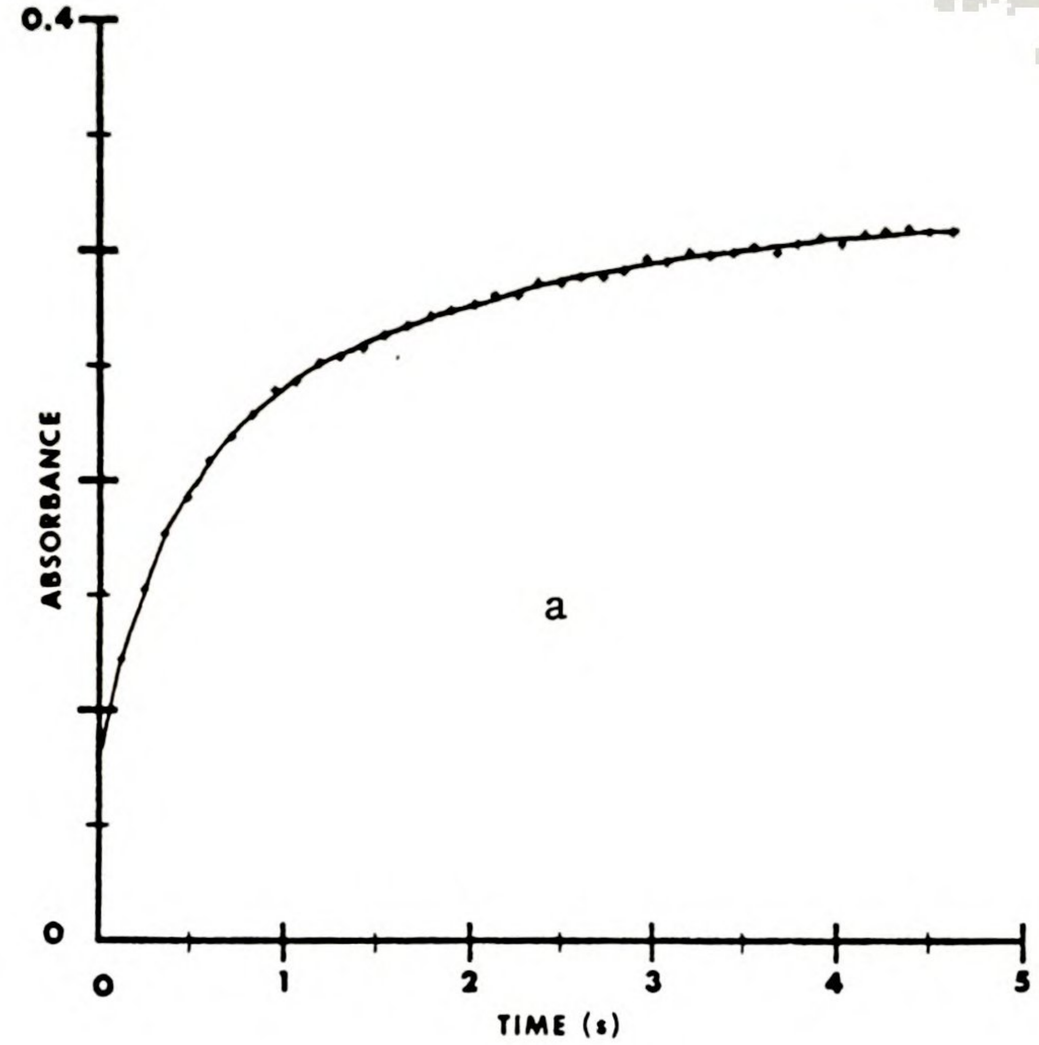
reaction profile is quite similar to the bovine serum albumin reaction. The single isobestic point is shifted slightly to 676 nm. The absorbance changes which occur at 578 nm (a) and 775 nm (b) are shown in Figure 7-7.

C. Analysis of the Biuret Reaction Spectra

The absorbance of the copper-protein complex should increase during the course of the reaction. The existence of isobestic points and wavelength regions over which the absorbance decreases with time indicates the presence of at least one interfering species which absorbs over this wavelength region. No interference had been previously reported, and this discovery is the most important result of the current investigation.

To determine the extent of this interference the identity of the new species and the spectral regions over which all of the species present during the reaction absorb, must be determined. Figure 7-8 shows the absorption spectrum of the copper complex with human serum protein. It was obtained by letting the biuret reaction go to completion (about 25 min.). A comparison of this spectrum and the spectra in Figure 7-1 indicates that there is only a single interfering species which absorbs over this spectral region. The increase in absorbance with time at wavelengths below 690 nm and above 800 nm is due to the copper-protein complex.

The cupric tartrate complex present in the biuret reagent was identified as the interfering species. Figure 7-9 shows the absorption spectrum of the cupric tartrate complex, with the biuret



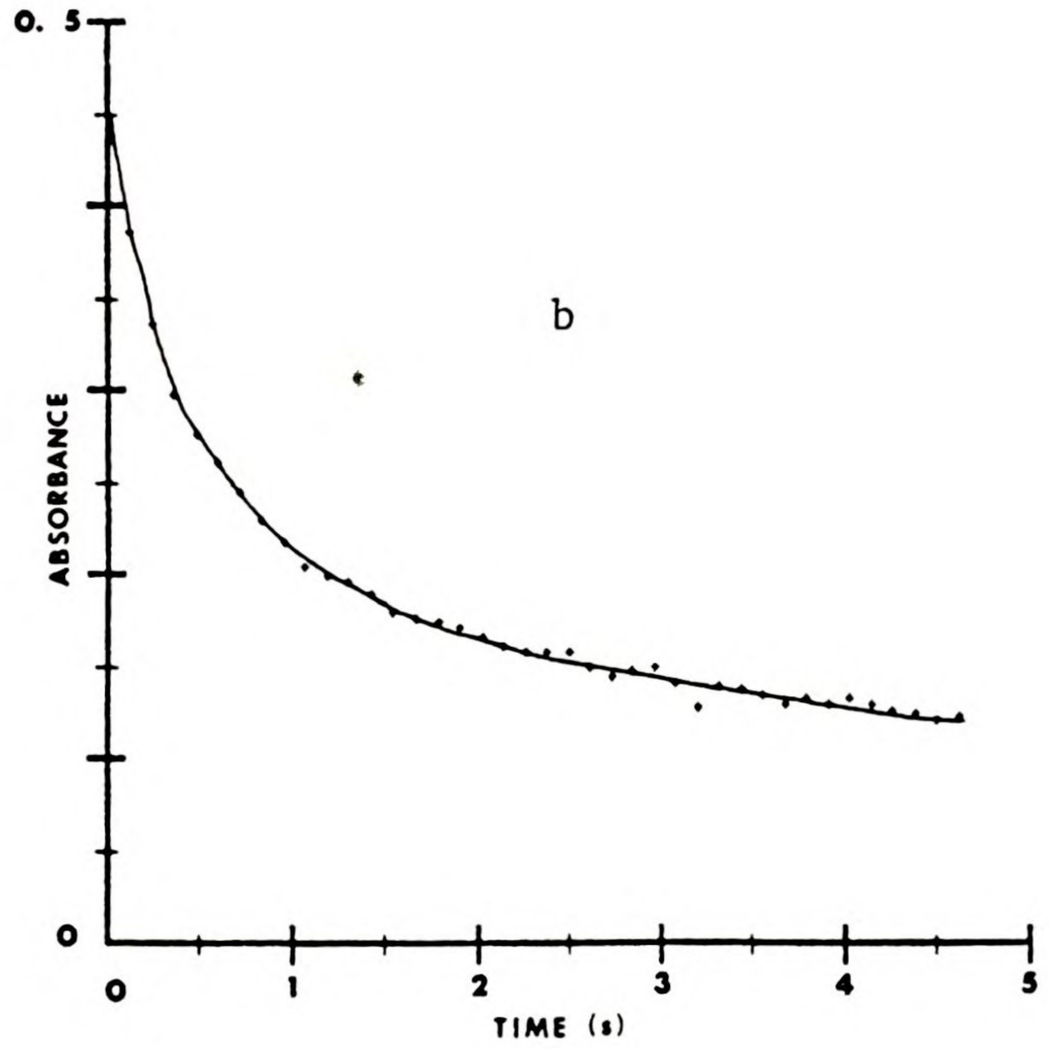


Figure 7-7. Absorbance Versus Time for the Biuret Reaction with Bovine Serum Globulin. (a) 578 nm; (b) 775 nm.

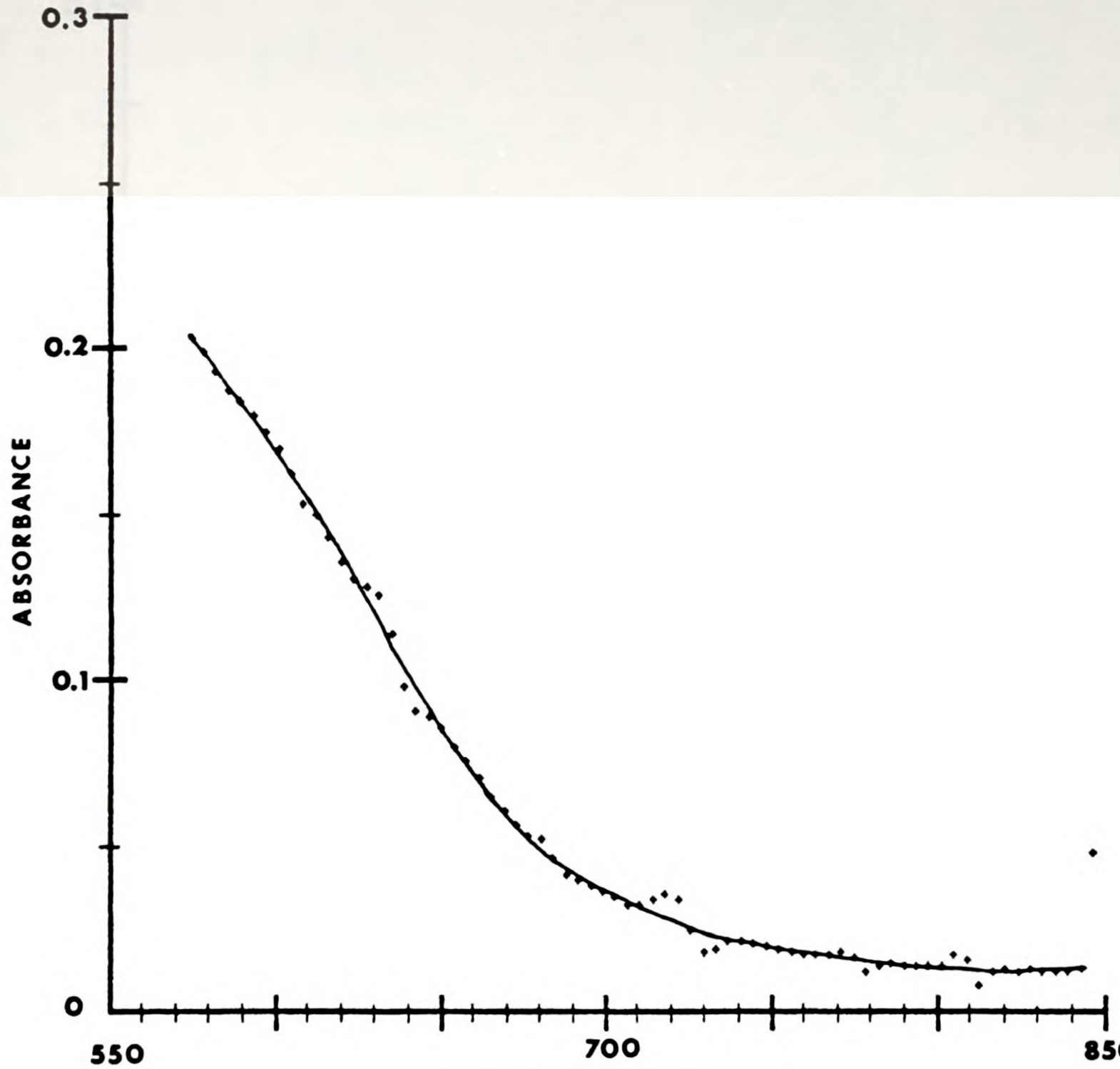


Figure 7-8. Absorption Spectrum of the Copper-Protein Complex from the Biuret Reaction with Human Serum After About 25 Minutes.

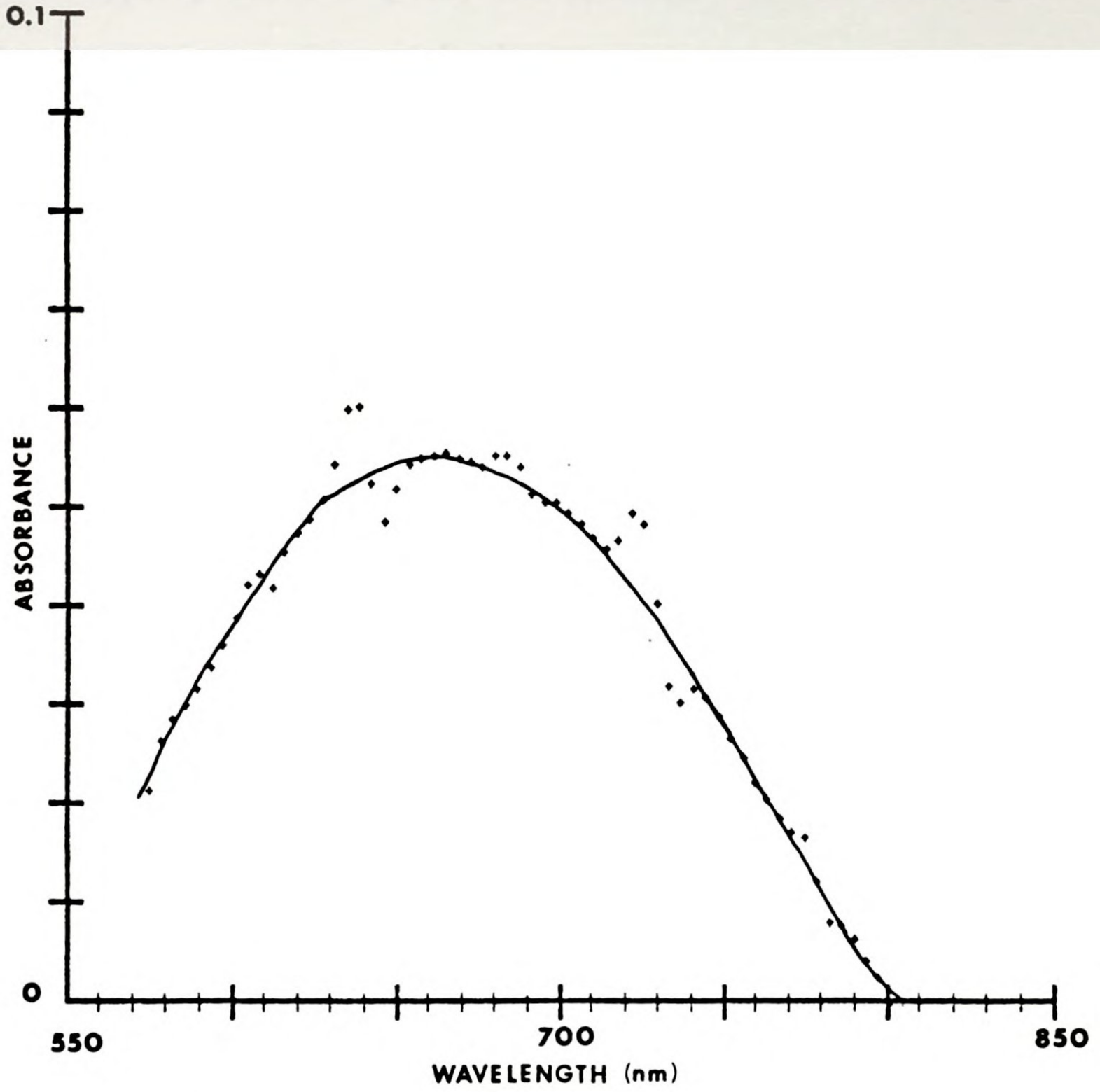


Figure 7-9. Absorption Spectrum of the Biuret Reagent.

blank used as a reference. The cupric tartrate complex shows an absorption maximum at about 660 nm and it still has significant absorbance at 550 nm. The interference from the cupric tartrate complex causes a decrease in the observed rate of absorbance change for the copper-protein complex, and accounts for the wavelength shift of the absorption maximum with time.

Since the cupric tartrate complex does not absorb above 800 nm, it does not explain the relatively large absorbance changes observed for both bovine serum samples between 800 nm and 850 nm (see Figure 7-4 and 7-6). Figure 7-10 shows the absorbance spectrum of the copper-protein complex from bovine serum albumin after about 25 minutes of reaction. Again, there is essentially no absorbance above 800 nm. Both of the bovine serum samples also showed slight shifts in the absorption maxima with time; however the shifts were far less dramatic than those found in the human serum samples. These observations indicate that the absorbance from interfering species in the biuret reaction with the bovine serum samples has a maximum at a larger wavelength than the cupric tartrate complex. One possible explanation for this is that a new copper complex of unknown composition is formed almost immediately after the bovine samples and the biuret reagent are mixed. The decrease in the concentration of the cupric tartrate complex reduces its interference at 550 nm. Also, since the absorption maximum of the new interfering species occurs at longer wavelengths that the cupric tartrate complex, it is likely that its absorbance at 550 nm is significantly less than the absorbance of the cupric tartrate complex. This lessened interference

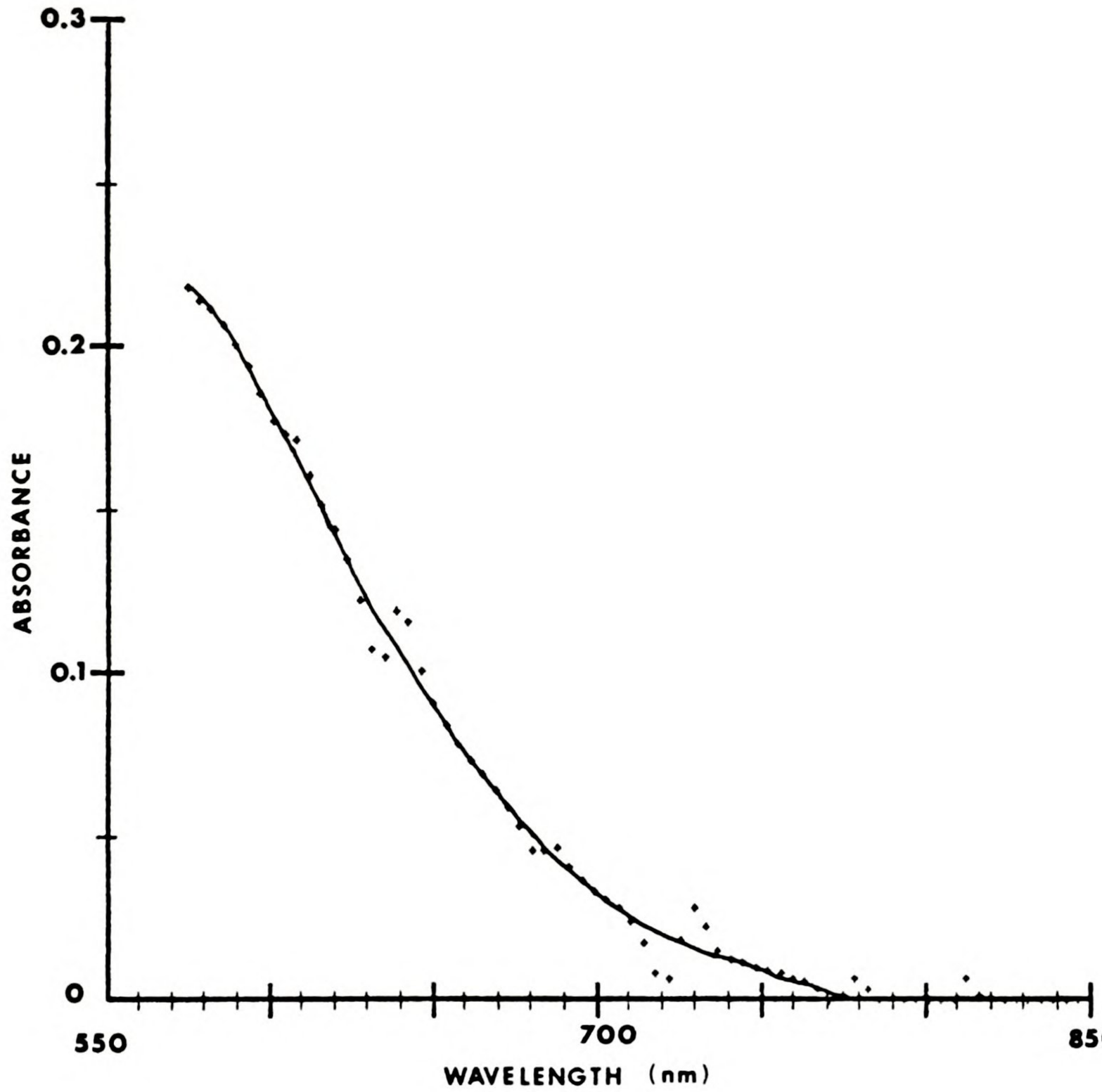


Figure 7-10. Absorption Spectrum of the Copper-Protein Complex from the Biuret Reaction with Bovine Serum Albumin After About 25 Minutes.

in the bovine samples at 550 nm could account for the higher rate of absorbance change observed in these samples.

Two important discoveries have resulted from this investigation of the biuret reaction. The first major result was the discovery of a chemical species in the reaction which absorbed over the same wavelength region as the copper-protein complex. This discovery enabled the new species to be identified as the cupric tartrate complex and the extent of its interference on the observed rate of absorbance change for the copper-protein complex could be determined. The second important finding was the discovery that the interfering species in the biuret reaction with bovine serum samples are different, and that the absorption maximum of the interference lies at longer wavelengths than that of the cupric tartrate complex. On the basis of these results it is possible to suggest further studies which could help characterize the biuret reaction. The rate curve should be investigated at wavelengths below 550 nm. In this region the absorbance of the cupric tartrate complex should be relatively small. This study would help to quantify the effect of the cupric tartrate complex on the reaction rate at 550 nm. Also, comparisons with the bovine serum samples would indicate if the differences in the rates were due to the cupric tartrate complex interference. (At the present time this study could not be performed on the vidicon system, since a problem with the monochromator dramatically reduced the sensitivity of the system below 550 nm.)

Another important study would be to identify the interfering copper complex in the biuret reaction with the bovine serum samples.

Once identified, the spectral region over which it absorbs could be determined and its effect on the rate curve at 550 nm could be more precisely determined.

One critical question which has not been addressed at all is whether the many different proteins found in human serum samples react with the biuret reagent at different rates. The irregular decrease in the rate curve for the biuret reaction could be due to these varying rates. This is a very important problem, since certain diseases cause increases or decreases in only a single type of protein (64), and kinetic analysis may be able to differentiate the different types.

CHAPTER 8

CONVERS-M

A USER-DEFINABLE LANGUAGE

FOR COMPUTERIZED INSTRUMENTATION

A. The Role of Software in Computerized Instrumentation

As the use of computers in the sciences becomes more and more common, their role is rapidly shifting. Throughout most of their history computers have been large and expensive, forcing them to be used primarily as remote, off-line data processors. In this type of application the principal task of the computer was to perform data file manipulations and complex mathematical processing of already existing data. As computers became smaller and less expensive, examples of computers also being used for data acquisition and real-time instrumental control became more common. With the recent introduction of small and inexpensive microprocessors, computers are now being found in a rapidly expanding variety of scientific instrumentation. This has caused a dramatic shift in the types of tasks that a computer is commonly required to perform. Sequencing, formatting, self-checking, operator interaction, monitoring, communications, and many other modes of automation are increasingly common in dedicated systems.

Since most of the common computer languages were written long before the rapid increase in computerized instrumentation, a re-evaluation in terms of current applicability is in order. It is necessary to set up some criteria by which to compare the various

computer languages. A large percentage of the most important criteria can be summed up with the conclusion that the computer language must enable the programmer to create the required software as rapidly as possible. The speed with which the programmer completes the necessary software is primarily determined by the number of lines of code that he is required to write. Therefore, the language must be versatile enough and of a high enough level that most tasks can be accomplished with only a few lines of code. Also, it is important that there be a library of commonly used routines (e.g. least squares fit, etc.) so that the programmer need not create them all himself. Obviously, if the completed software does not execute rapidly enough to accomplish the desired task, than it does not matter how few lines of code were necessary to create the software. Therefore, the execution speed of the language is very important. It is always important that the completed software take up as little computer memory as possible and this is especially critical when the computerized instrument is to be mass produced. The above criteria (program level, routine libraries, execution time and memory requirements) will be applied to the evaluation of the languages most commonly used in computerized instrumentation.

The most basic type of computer language is machine code, which is the numeric code used directly by the computer and therefore is the only code upon which the computer actually operates. All other computer languages must first have their commands translated into machine code before they can be executed. Routine computer programming is seldom done in machine code for many reasons. Since the

language is entirely numeric, programs in machine code are extremely lengthy and very tedious to write. Also, each type of computer has its own specific machine code, which makes the development of standard routine libraries impossible.

The programming level of assembly language is only slightly higher than that of machine code. The real advantage of assembly language lies in its use of English mnemonics to replace the numeric codes of machine language. Before any assembly language program can be used, all of the English mnemonics must be converted to their machine code equivalents. This is most commonly done through the use of a program called an "assembler". Assembly language is extensively used in computerized instrumentation because it is the most common computer language now in general use which can be applied to instrumental control and data acquisition. Well written assembly language programs require less memory and execute faster than programs written in any other type of code; however, it shares with machine code a number of distinct disadvantages. Since it is a low level language, even simple tasks require many lines of assembly code. Each type of computer has its own assembly language, resulting in the establishment of very few common routine libraries. The mathematical capabilities of different assembly languages vary with the type of computer which is used, but at best they include only integer arithmetic, with multiply and divide capabilities being rather uncommon.

FORTRAN and BASIC are the two high level computer languages which are most often used in computerized instrumentation. Their command structures enable them to handle file manipulations and

complex mathematics that would be extremely difficult and lengthy if done with assembly language. As with any high level language, a single line of FORTRAN or BASIC code can perform operations which would require many lines of code if written in assembly language. Before any FORTRAN or BASIC program can be executed, it must be converted into machine code. All FORTRAN languages and a few types of BASIC accomplish this conversion through the use of a "compiler" program. The compiler creates an executable machine language program which is equivalent to the original FORTRAN or BASIC program. Most types of BASIC use an "interpreter" program instead of a compiler. The lines of BASIC code are interpreted and executed one line at a time. Since most computers have compiler and/or interpreter programs available, FORTRAN and BASIC programs can be run on almost any type of computer. BASIC, and especially FORTRAN, have been used long enough that extensive libraries or common routines are in existence. Unfortunately, FORTRAN, and especially BASIC, routines tend to be quite slow during actual execution and their memory requirements are high. In addition, standard FORTRAN and BASIC contain no commands which could be used for data acquisition or instrumental control. The way in which FORTRAN and BASIC are adapted for such use is by programming in assembly language. Assembly language subroutines are written for the various modes of instrumental control and data acquisition. These routines are then linked together with the FORTRAN or BASIC routines which handle the data processing.

It is obvious that there is no single computer language currently in general use which can easily solve all the problems normally found

in computerized instrumentation. There are very few languages which have any high level "instrumental" commands for instrumental control or data acquisition. Furthermore, it would probably be impossible to create a computer language with a set of "instrumental" commands versatile enough to handle all the extremely varied tasks found in the different types of computerized instrumentation.

There is a possible solution to this dilemma. Recently a new class of computer language has begun to develop. CONVERS-M, a language developed at Michigan State University, is one version of this new type of computer language. The basic concepts of CONVERS-M were based on the language CONVERS developed at the University of Arizona (71). CONVERS itself was based on FORTH (72), the most widely distributed language of this type. CONVERS-M does not have the rigid command structure characteristic of most other languages. In fact, its major advantage lies in the ease with which new commands can be created and used within its structure. The result is a language which is not restricted to a small range of programming levels but, as shown in Figure 8-1, can be written on any level from machine code to commands more powerful than FORTRAN commands. This makes it possible to create "instrumental" commands tailored to the specific application. The execution speed of CONVERS-M depends upon the level at which it is written. For most instrumental control applications the speed is the same as an equivalent machine code routine. Its major disadvantage is that it is a new language and therefore software libraries have not yet been established.

Table 8-1 summarizes the advantages and disadvantages of the

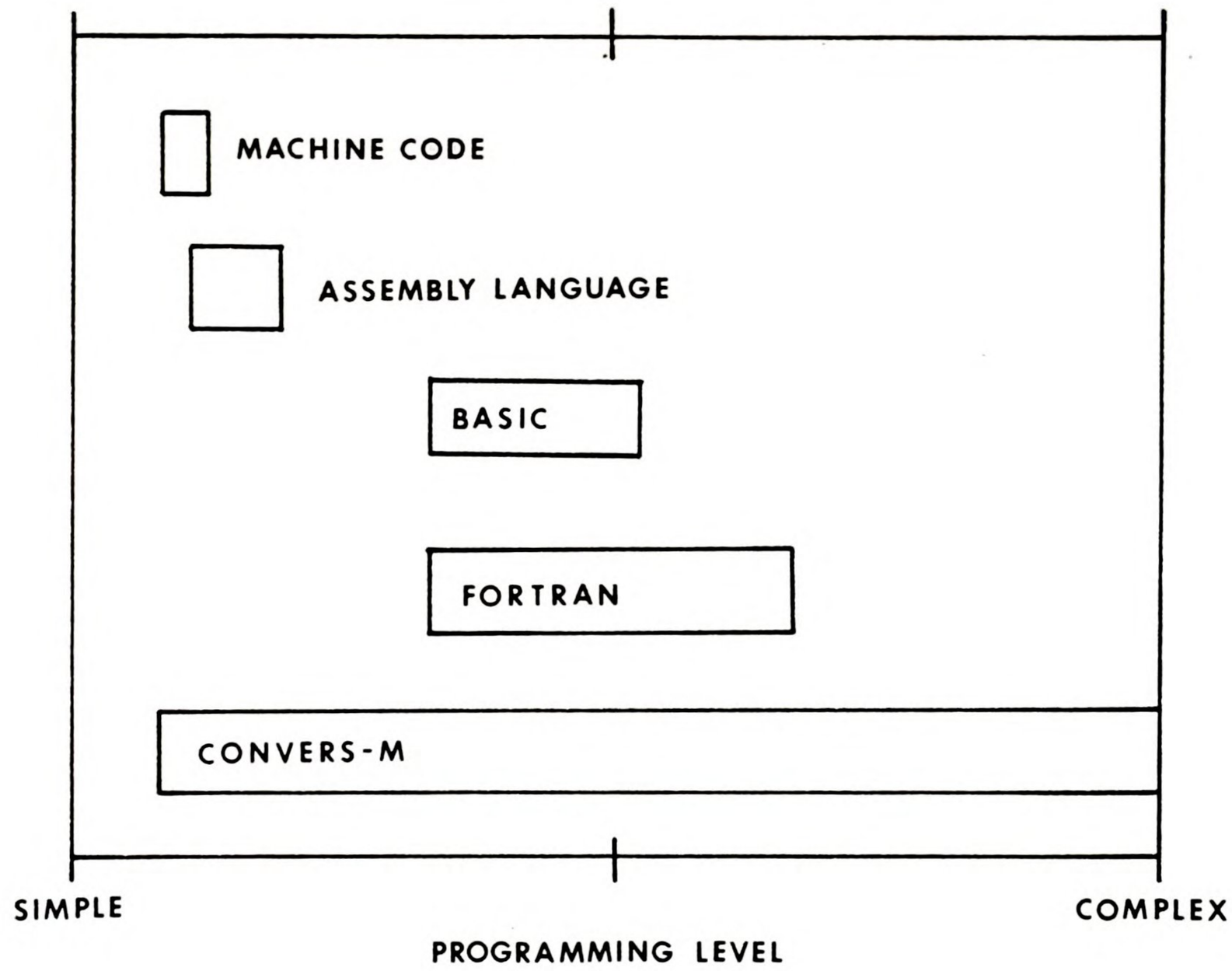


Figure 8-1. Programming Levels of Various Computer Languages.

Table 8-1. Comparison of Common Programming Languages.

Language	Programming Level	Standard Libraries Available	Execution Time	Core Requirements
Machine	Lowest	No	Fast	Small
Assembly	Low	Few	Fast	Small
BASIC	High	Yes	Very Slow	Large
FORTRAN	High	Yes	Slow	Large
CONVERS-M	Any Level	No	Fairly Fast	Medium

various languages examined. It shows that although CONVERS-M is not the ultimate solution, it is at least a start in the right direction.

B. Programming with CONVERS-M

CONVERS-M represents a style of programming which is totally different from any other programming technique. Whereas most programming languages are based on a single, predefined set of commands which can only be altered or expanded with difficulty, programming with CONVERS-M consists almost entirely of creating new commands. This type of dynamic command structure is best illustrated through the use of examples.

One task of the computer system in the vidicon rapid scanning stopped-flow spectrophotometer is to open a valve, thereby enabling two chemicals to rapidly mix and flow into an observation cell. Since this is a fairly common operation in the system, it was decided to create a command called FLOW which would cause this to happen. The simplest way to create this command is to type, at the terminal, terminal, the following set of instructions:

```
<<  FLOW
    PUSH    PSW
    MVI     A,4
    STA     16000
    POP     PSW
    >>
```

The instruction '<<' tells the CONVERS-M monitor that a new command

named FLOW is about to be defined. The next four lines are assembly language instructions which, when executed, interact with the hardware and cause the valve to open. The final instructin, '>>', signals the monitor that the definition has been completed.

The completed command can be used in two different ways. When only the command is typed at the terminal it is being used in the run mode. In this mode the command is executed, causing the valve to open. This allows testing of the new command and, later on, provides a simple method of testing the valve and associated electronics in case of a hardware failure. The other mode in which FLOW can be used is in the creation of new commands. This mode, defined as the compile mode, will be illustrated through the use of another example. A more complex task that must sometimes be performed in the stopped-flow system is to empty the sample and reagent syringes. The syringes can be partially emptied by opening the flow valve, then closing it and allowing the reacted chemicals to be expelled to the waste container. If this is repeated a number of times, the syringes will be emptied. A command called EMPTY was created to accomplish this task by typing the following:

```
<<  EMPTY
    PUSH    PSW
    MVI     A,4
:10 WAIT
    FLOW
    WAIT
    WASTE
```

```
DCR     A
JNZ     .10
POP     PSW
>>
```

The definition of the EMPTY command differs in only two ways from the simple definition of the command FLOW. First there are three commands, including FLOW, which are not assembly language commands. The other two non-assembly level commands are WAIT, which causes the system to wait until the valves have switched and the solutions have stopped moving, and WASTE, which causes the solution to be emptied to the waste container. Both of these commands have been previously defined, just as FLOW was. In this mode the FLOW command is not executed immediately. Rather, a subroutine call to the routine FLOW is linked into the EMPTY command.

The other new technique which is illustrated in the EMPTY command is the use of branching within a command definition. In this case it is desired to execute the command sequence, WAIT, FLOW, WAIT and WASTE, four times. This is accomplished by three assembly language instructions. First the instruction 'MVI A,4' loads the accumulator register in the computer with 4. The command sequence is then executed once, the 'DCR A' instruction causes the accumulator to be decremented by one and if the result is non-zero the 'JNZ .10' instruction causes the system to branch back to the WAIT command referenced by the ':10' label. In general, any location within a command definition can be labeled with an octal number between 1 and

177, preceeded by a colon (e.g. :10, :4, :103, etc.). This location can then be referenced by repeating the octal number, this time preceeded by a decimal point (e.g. .10, .4, .103, etc.).

In all the preceeding examples, when a command was used in the compile mode, a subroutine call to the specified command was linked into the definition. This is the simplest and most common type of command. A more powerful and versatile type of command, the macro command, is any command which causes more than a simple subroutine call to be linked into the definition. One way in which a macro command might be used would be to speed up the execution of a command by linking the actual assembly code for the command into the definition instead of linking in a subroutine call to the command. For example, forty percent of the execution time of the FLOW command is involved in the subroutine call and the return from the subroutine. If these steps were eliminated, the command would execute much more rapidly. The FLOW command could be made into a macro command by defining it in the following way:

```
<<  FLOW
MODE    FLOW    300
MAC     7
PUSH    PSW
MVI     A,4
STA     16000
POP     PSW
>>
```

Only two new lines of code are required to make FLOW into a macro command. The first new line of code, starting with MODE, informs the monitor that the command FLOW should execute even in the compile mode. The second new line of code starting with MAC, causes the four assembly language instructions (7 bytes of code) to be linked into a definition when the FLOW command is used.

CONVERS-M was designed to operate in the multi-microcomputer system described in Chapter 3. CONVERS-M enables most of the details of the interaction between the microcomputers to be kept transparent to the programmer. Each microcomputer is assigned a number with the coordinator designated as microcomputer 0. To create commands in any of the microcomputers, it is first necessary to specify in which microcomputer the commands are to be created. For example, the FLOW command described earlier must reside in the stopped-flow microcomputer which is microcomputer 4. By typing 'MICRO 4' at the terminal, any commands, including FLOW, which are now created will reside in microcomputer 4. Once the commands are created in the slave microcomputers, it is necessary to be able to create routines in the coordinator which are able to cause these routines in the slaves to be executed. This is accomplished through the use of four macro commands. An example of a program segment, which would cause the command FLOW to be executed by the stopped-flow slave, is the following:

```
        SETUP   4,FLOW
        RUN     4
:10     FLAG    4
        JZ      .10
```

```
STOP    4
```

The first line specifies that the command FLOW on slave 4 is to be executed. The next line creates a code which will cause slave 4 to begin execution of the FLOW command. The next two lines form a loop which checks the 'task completed flag' of slave 4 to determine when the FLOW command has completed execution. This final line causes slave 4 to halt any further program execution.

For further details on programming with CONVERS-M, refer to Appendix A (The CONVERS-M User's Guide) and Appendix B (CONVERS-M Basic Command List).

C. The Structure of CONVERS-M

The command library of CONVERS-M is structured to allow for simple expansion. As new commands are created, they are entered into the library. Figure 8-2 shows the way in which a typical library command is stored in memory. The body of the command definition is preceeded by the command header. This header contains the command name, a two byte link address used during library searching, and a mode byte which specifies the mode in which the command is used.

Whenever a command is used, it is necessary to search the library in order to find the command. The search of the library is accomplished through the use of the link addresses stored with each command. Figure 8-3 shows how the library search is performed. The library is searched beginning with the last command that was entered. If that command is not the one sought, then the search is transferred to the address stored in the link address following the command name.

Storage Location	Contents	Command Definition	
0	# Characters	<<	FLOW
1	F		
2	L		
3	O		
4	W		
5	Low Byte Link Address		
6	High Byte Link Address		
7	Mode		
8	Assembly Code	PUSH	PSW
9	↓	MVI	A,4
10	↓		
11	↓	STA	16000
12	↓		
13	↓		
14	↓	POP	PSW
15	Return	>>	

Figure 8-2. The Structure of a Command Definition.

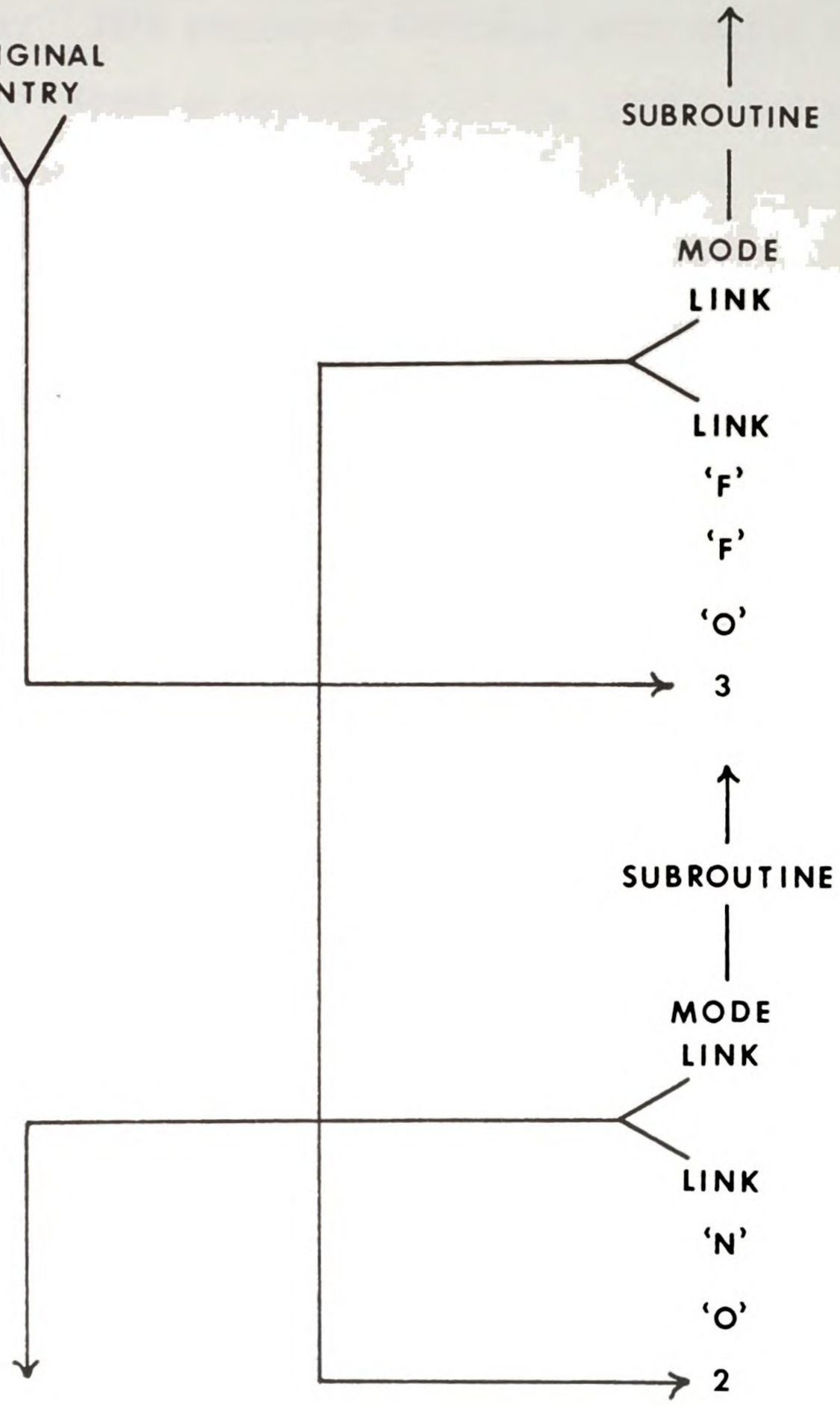


Figure 8-3. CONVERS-M Library Searching Procedure.

The link address points to the command name which is the next lowest in the library. This process is continued until either the command being sought is found or the entire library is searched with no match being found. The end of the library is found when the link address points to a command name whose length is zero. If the character string being searched for is not found in the library, then the attempt to interpret the character string continues on through a number of levels. For example, in the compile mode, the character string is compared to assembly language instructions to determine if there is a match. The entire process of character string interpretation is shown, in detail, in Appendix C (The CONVERS-M Flow Chart).

When CONVERS-M operates in the multi-microcomputer system each microcomputer contains its own separate library of commands. The coordinator microcomputer contains almost all of the CONVERS-M monitor overhead for the entire system. The basic memory allocations for CONVERS-M in the coordinator and a typical slave microcomputer are illustrated in Figure 8-4. The first 1 K (the first 1024 memory locations) in the coordinator contain the CONVERS-M system routines. These are routines that are not stored in the normal library format. This is the heart of the CONVERS-M monitor. The first section of code found here is the system initialization routine, which is executed only when the entire system is reset. Next comes the character string interpreter, followed by the assembler routines, the label decoding routines, and finally, the assembly language conversion tables. The next 1 K of memory contains the minimal CONVERS-M library. Here are found routines, stored in the normal library format, which

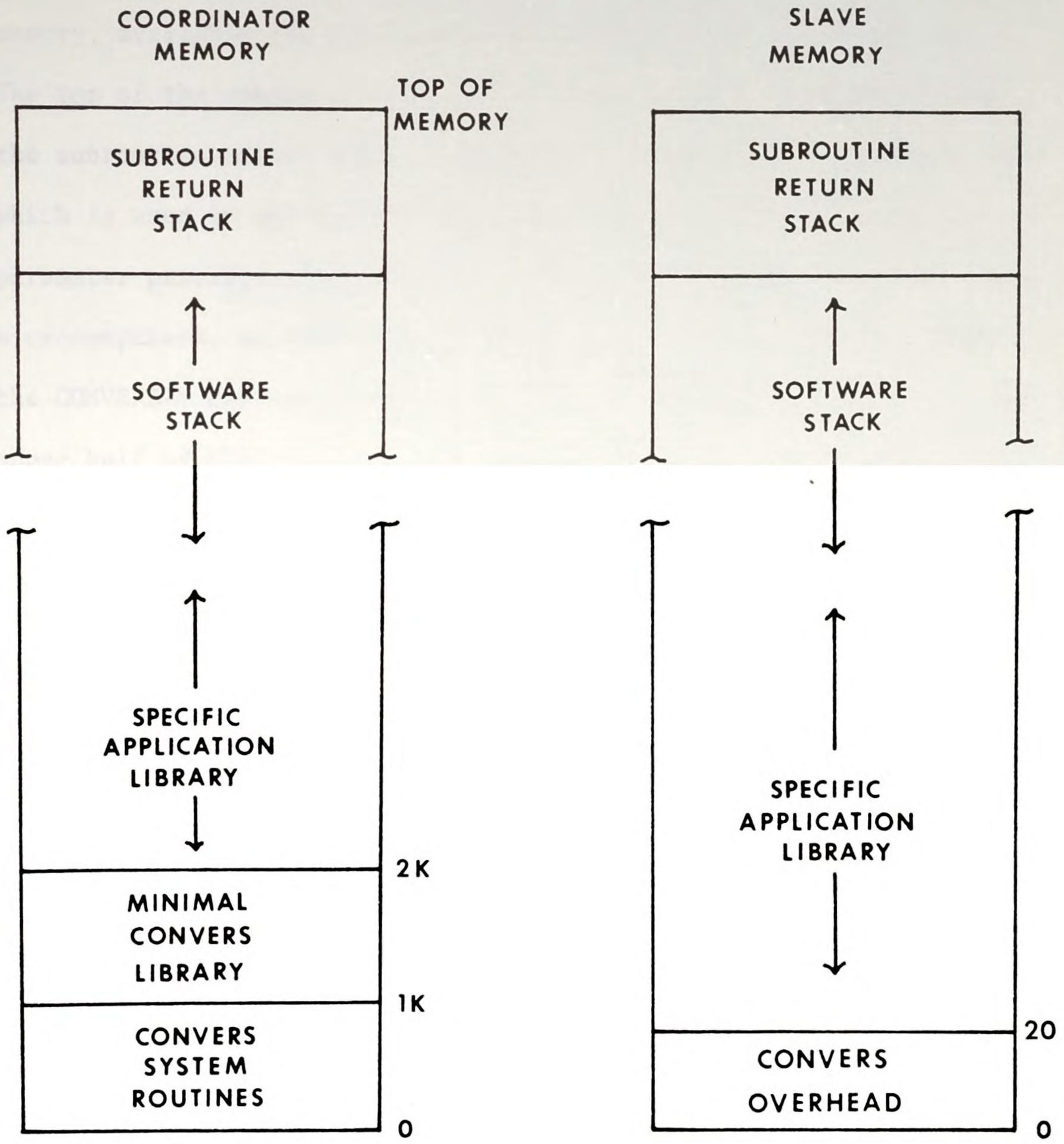


Figure 8-4. CONVERS-M Memory Allocation.

are essential to the operation of the monitor. Above this is free memory, available for the storage of specific application libraries. The top of the memory in each of the microcomputers is reserved for the subroutine return stack. Below this is found the software stack which is used in one type of parameter passing. (See the section on parameter passing in Appendix A). The memory allocation in the slave microcomputers, as indicated in Figure 8-4, is much simpler. Here the CONVERS-M monitor overhead requires only about 20 locations. The upper half of this area contains monitor parameter storage locations. In the lower half of this area reside the four to six assembly language instructions which are required to enable any routine in the slave to be executed. These instructions first load the hardware stack pointer and then perform a subroutine call to the desired routine. Upon return from this routine, the slave signals the coordinator that it has completed the routine and then halts. In this way the coordinator can cause any routine in the slave to be executed by simply placing the address of that routine in the subroutine call and then running that microcomputer. Since the monitor overhead is so small in the slave microcomputers, most of the memory is left free for the specific application libraries for the slaves.

APPENDICES

APPENDIX A

CONVERS-M USER'S GUIDE

APPENDIX A

CONVERS-M USER'S GUIDE

A. Introduction

This Appendix is only designed to give details on some of the programming conventions used in CONVERS-M and is, therefore, not a complete manual for the use of CONVERS-M. The structure of CONVERS-M and some examples of its use can be found in Chapter 8. Appendix B contains a listing of all the commands which are found in the minimum CONVERS-M library and a flow chart of the operation of the CONVERS-M monitor appears in Appendix C.

B. Special Characters

Most operating systems input an entire line of characters before actually beginning to interpret the character string. CONVERS-M operates on character strings which are smaller than this. Here, a character string is defined as a group of characters followed by a separator which is either a comma, space, tab, or carriage return. The monitor does not differentiate among the four separators, therefore, in any specific application, it does not matter which of the four separators is used. The following two program segments differ only in the separators which are used and, therefore, are compiled identically by the monitor.

```
        MVI   A,10
:10     DCR   A
```

```
                    JNZ     .10
                    -
            MVI,A   10,:10  DCR
            A,JNZ   .10
```

The semicolon is a special character which is used to indicate a comment. Comments are character strings which appear on the terminal, but are ignored by the CONVERS-M monitor. A comment begins with a semicolon and is terminated by a carriage return. There are no restrictions on the characters which may appear within a comment.

```
; **
; Examples of Comments
; **
            MVI     A,10    ; At the end of a line
; Between Lines
: 10        DCR     A
            JNZ     .10
```

C. Labels

Labels are character strings which are used to define or reference locations within a command definition. They are always octal numbers between 1 and 177, preceeded by either a period or a colon (e.g. :10, .26, :170, etc.). If the label starts with a colon, then it is defining a location. A single octal number can define only a single location. If a second label definition is attempted with the same octal number, then the monitor will signal an error by outputting a 'P', indicating that the number was used in a previous definition. These defined

locations can now be referenced by using the same octal number, this time preceeded by a period.

When a command is being created, a temporary library of the label definitions and references is stored in memory. This temporary library is located just below the software stack. Each time a label is defined, the memory address at which the compiler will next store code is entered into the temporary library. Whenever that label is referenced, this address is stored in the next two bytes of the command definition. An example of the use of labels is shown in the simple timing loop routine shown below.

```
        LDA    .20
:10     DCR    A
        JNZ    .10
        RET
:20     10
```

In this example the 'A' register is loaded with the octal value 10. This register is decremented until it is zero, and then command is returned to the calling routine.

D. Parameter Passing

It is often necessary to transfer data or parameters from one routine to another. There are a number of ways that this can be accomplished. The most common method is to use the software stack. The software stack is a last-in-first-out data buffer located near the top of memory below the subroutine return stack. The address, at which

the data is currently being stored, is kept in the stack pointer which occupies two bytes of storage in the monitor. As this data buffer fills, the data is stored in successively lower memory addresses. This leaves the bottom of memory free for the user's software. There are four commands in the basic monitor which simplify the use of the software stack. To move a single byte of data onto the software stack, the 'ON' command is used. This command moves a single byte of data from the 'A' register onto the stack. To retrieve this byte, the 'OFF' command is used. This command removes a byte from this stack and places it in the 'A' register. In the following example, the first program segment places the octal number 10 onto the stack and the second segment retrieves that number, and moves it to the 'B' register.

```
MVI     A,10
ON


OFF
MOV     B,A
```

It is also possible to use the software stack for two byte data. In this case the command 'PUSH2" puts the two bytes contained in the 'H' and 'L' registers on the stack. The command 'POP2' removes two bytes of data from the stack and places it in the 'H' and 'L' registers.

Another method of passing data to another routine, is to store the data directly after the call to that routine. An example of an assembly language routine which could accept this type of data transfer, is the following:

```
<<      WAIT
        XTHL
        MOV     A,M
        INX     H
        XTHL
:10     DCR     A
        JNZ     .10
```

This routine causes a time delay whose length is determined by the byte of data following a call to this routine. For example, if the line 'WAIT 6' appeared in a command definition, then the internal loop in the 'WAIT' routine would be executed six times. The actual transfer of the data to the 'WAIT' routine is accomplished by the assembly language instructions which form the second through fifth lines of the definition. These four lines of code move the byte into the 'A' register and also increment the return address to insure that the return will occur to the location after the data byte. This type of parameter passing is used primarily when the data is predefined and will never change. The above 'WAIT' routine would be a good example in an application where a number of software delays, each of a different predetermined length, were needed. Variable data is best handled through the software stack.

When it is necessary to pass data very rapidly between routines, the internal registers of the computer can be used. Another technique which is almost as fast is to use a predetermined address in memory to store the common data. These last two methods should only be used when no other method is fast enough, since it requires the programmer to

remember in which registers or memory addresses the data must be stored.

E. Mode Word

Each command definition has a one byte mode word stored with it. The default mode specifies that when the monitor is in the run state the routine will execute only if the coordinator microcomputer's library is currently loaded and the routine is found in that library. If the monitor is in the compile state, a call to the routine will be compiled only if the routine is found in the currently loaded library. Bit 6 and bit 7 in the mode word are the 'run only' bits. They are used to override the default actions and cause routines to be executed when they normally would not be. Since the monitor can only execute routines found in the coordinator's library, these mode bits only affect routines found in that library. Bit 7 is the 'run only' bit for the coordinator's library. If the coordinator's library is currently loaded and bit 7 is set, the routine will execute regardless of whether the monitor is in the run or compile state. Bit 6 is the 'run only' bit for a slave's library. If a slave's library is currently loaded and the routine is found in the coordinator's library, the routine will execute regardless of the state of the monitor. If both bits are set, then the routine will always run.

F. Using the Floppy Disk Drive

Mass data storage for the multi-microcomputer system is accomplished through the use of floppy disks. They are used for storing binary programs, command definitions and data. Most of the programmer's

interaction with the floppy disk drive is handled by the 'FLOPY' routine. The floppy disk drive is interfaced through the Persci Model 1070 Intelligent Diskette Controller. Since the controller contains a powerful set of commands for manipulating files on the disks, the 'FLOPY' routine simply serves as a means of passing these commands from the user to the controller. A complete description of the controller's commands can be found in the product specifications for the Model 1070 controller (73). The 'FLOPY' routine interacts in a special way with two of the controller's commands. The controller's 'LOAD' (L) command causes the contents of the specified file to be listed on the terminal. The 'SAVE' (S) command can be used to create files on the disk. The process of creating a file is started when the 'SAVE' command is entered. The file name specified with the command is entered on the disk as the name of the new file which is being created. From this point on, everything that is typed at the terminal will be entered into a data buffer. Each time a carriage return is typed, the contents of the buffer are written into the disk file. When a control/C is typed, the input is terminated and the disk file is closed. An example of the creation of the disk file, WAIT.CON, is shown below.

```
FLOPY
* S     WAIT.CON
<<      WAIT
        MVI     A,10
:10     DCR     A
        JNZ     .10
```

```
>>
(CTRL/C)
*
```

In the above example the asterisks (*) are output by the 'FLOPY' routine whenever it is inputting command strings to be sent to the controller. To terminate the 'FLOPY' routine and return to the monitor, control/C is typed at the terminal.

G. Assembly (Compilation of stored routines)

It would be extremely tedious if the programmer had to enter, at the keyboard, the definitions for all the commands not already found in the monitor, which he planned to use during that session. To alleviate this, it is possible to enter command definitions that were previously stored on floppy disks. This is done through the use of the 'ASSM' routine. The 'ASSM' routine interacts with the floppy disk controller in much the same manner as the 'FLOPY' routine described in the preceeding section. In this case, when the controller's 'LOAD' (L) command is used, the contents of the specified file are compiled into the library which is currently loaded. For example, to compile the 'WAIT' routine, which was used as an example in the last section, the following would be typed:

```
ASSM
# L WAIT.CON
```

The number sign (#) is output by the 'ASSM' routine to indicate that it

is ready for a controller command. As the 'WAIT' routine is read from the disk, it is output to the terminal along with any output (normally error messages) from the monitor itself. As soon as the end of the disk file is encountered, the system returns to the monitor.

Since the entire disk file is output to the terminal, it is often difficult to notice any error messages which the monitor might output. This problem can be overcome by the 'DCRT' command which allows only output from the monitor to be sent to the terminal when the 'ASSM' routine is used. The 'DCRT' command inputs an octal number from the terminal. If the number is 6, then all input from the disk will be sent to the terminal also. If the number is not 6, the terminal output is disabled.

The 'ASSM' routine functions by forcing all input to the monitor to come from the floppy disk instead of the terminal as it normally does. Upon encountering the end of file from the disk, monitor input once again comes only from the terminal. For this reason, almost any commands which are allowed at the terminal can be input from floppy disk as well. The only exceptions are commands which also use the floppy controller. This means that batch streams can be stored on the disks and executed at any time.

H. Communication with the PDP 11/40

It is often necessary to send files to and from the PDP 11/40 minicomputer. This communication is accomplished by the 'PIP' routine in the multi-microcomputer system. (This routine is not found in the basic CONVERS-M library, it must be loaded from floppy disk). When

this routine is run, it first outputs a dollar sign ($) indicating that it is requesting the mode of the current transfer. This mode that must be input is made up of two characters. The first character indicates the destination of the transfer. If the character which is typed is a 'U', then the destination of the transfer is the floppy disk in the multi-microcomputer system. If this character is an 'M', then the destination is the PDP 11/40. The second character defines the type of transfer which will take place. If the character is an 'N', it indicates that the transfer will be a 'normal' transfer. This means that an exact copy of the file will be transferred. The character 'A' indicates that the transfer will be an ASCII transfer. This is necessary since the CONVERS-M system uses only a carriage return to terminate a line, whereas, the PDP 11/40 uses both a carriage return and a line feed to terminate a line. This problem is corrected during an ASCII transfer.

When the 'PIP' routine outputs a number sign (#), it indicates that it is ready to input a standard file specification for the PDP 11/40. After that has been input, the routine will output an asterisk (*), this time requesting a file name to be used on the floppy disk in the multi-microcomputer system. Once this file name has been input, the transfer will take place and the routine will be ready to input the next transfer mode from the user. As example of an ASCII transfer from a floppy disk on the PDP 11/40 to the multi-microcomputer system is shown below.

```
$ UA
# FL:WAIT.CON
```

```
* WAIT.CON
$
```

In the above example the file 'WAIT.CON' is transferred from the PDP 11/40 to the microcomputer system and placed in a file also named 'WAIT.CON'. To terminate the 'PIP' routine and return to the monitor control/C is typed at the terminal directly after the dollar sign ($).

I. Error Codes

Due to the necessity of keeping the CONVERS-M monitor 'short and simple', the error codes that are output by the system are limited. The monitor itself has only three different error codes. By far the most common error code is the question mark (?). This error can mean nearly anything. Some typical examples are: input of an unknown character string, illegal micro number, illegal label, illegal register during assembly, attempt to input something besides an octal number when an octal number is required, etc. At times the system can appear to 'lock up' and output question marks no matter what is typed. This can be caused by one of two problems. First, there are some routines that require the input of an octal number and they will continue to output question marks until an octal number is typed at the terminal. The second problem which can cause the system to 'lock up' is when an assembly language instruction is used which requires the use of a register code (e.g. MOV A,B etc.). The system will output question marks until a correct register code is input.

The other two error codes which the monitor uses involve improper use of labels or command names. The three error codes used by the

monitor are listed in Table A-1.

More detailed error codes are often encountered when using any of the routines which interact with the floppy disk controller. These error codes are part of the software which comes with the controller and all are self-explanatory (2).

Table A-1. Monitor Error Codes.

Error Code	Error
?	Nearly anything
P	Indicates a previously defined command name or label definition
nnnU	Indicates an undefined label with 'nnn' being the label in question

Sometimes detailed error codes are also found when using the 'PIP' routine to transfer routines to and from the PDP 11/40. The detailed error codes are sent by the PDP 11/40 when it encounters an error in the transfer. The 'PIP' routine also outputs a number of error codes, all of which begin with the letter 'F'. These 'PIP' error messages are listed in Table A-2.

Table A-2. 'PIP' Error Codes.

Error Code	Error
F0	Impossible? error code
F1	No ACK/EOT from PDP 11/40 on start up
F2	NAK from the PDP 11/40 (11/40 found an error)
F3	Checksum error during transfer to PDP 11/40
F4	Checksum error during transfer from PDP 11/40
F5	Unknown code sent from PDP 11/40

APPENDIX B

CONVERS-M BASIC COMMAND LIST

CONVERS-M BASIC COMMAND LIST

A. Coordinator's Basic Command List

ASCII — Performs octal to ASCII or ASCII to octal conversions. (i.e. adds or subtracts 60 octal to or from the data.)

The top of the stack must contain a mode byte.

Mode = 0: ASCII to Octal

Mode = 1: Octal to ASCII

	INPUT	OUTPUT
Top of Stack:	Mode	# Char.
	# Char.	Etc.
	Etc.	

ASSM — Is used to compile commands which are stored on floppy disk. Details of its use are given in section G of Appendix A.

BININ — Reads an octal number (between 0 and 177777) from the terminal. The octal number is converted to binary and the 2-byte result is stored on the stack.

Top of Stack: Low Byte

High Byte

If the input from the terminal is not octal, a question mark is output and a new number is accepted.

BINOCT Converts a 2-byte binary value on the stack to octal and puts the resultant octal digits on the stack.

	INPUT	OUTPUT
Top of Stack:	Low Byte	# of Digits
	High Byte	Least Sig. Digit
		Etc.
		Most Sig. Digit

Leading zeros are suppressed on output.

BINOUT Converts a 2-byte binary number found on the stack to octal and outputs it to the terminal with leading zeros suppressed.

Top of Stack: Low Byte
High Byte

CLEAN Is used to remove unwanted data strings from the stack. The top of the stack must contain the number of bytes to be removed.

CRLF Causes a carriage return / line feed to be output to the terminal.

DEP (Run only mode) - is used to alter a location in memory.

DEP AAA DDD

AAA - Octal address in memory

DDD - Octal data to store in AAA

EXM (Run only mode) - is used to examine a location in memory.

EXM AAA

AAA - Octal address to examine

The octal contents of AAA will be output to the terminal.

FLAG Is a macro routine which creates a code segment to check the task completed flag of the specified slave microcomputer. If the task is completed the internal zero flag will be reset.

FLOPY Handles file manipulations on the floppy disks. It is described in section F of Appendix A.

LP Inputs an octal number from the terminal. If it is a 5, then the output of the 'FLOPY' routine is sent to the TTY instead of the terminal. If any other number is input, then the 'FLOPY' routine functions normally.

MAC Simplifies the creation of macro routines. When run, it enables strings of code to be linked into a definition.

MAC 3, CALL 1106

The above example causes the three byte instruction, CALL

1106, to be linked into the definition.

MCHG Loads the CONVERS-M library of the micro corresponding to the 1-byte number on the top of the stack, (i.e. 0,1,2,3, etc.)

MESSAG Causes a message to be written on the terminal.

	INPUT	EXAMPLE
Top of Stack:	# Char.	4
	Last char. sent	'2'
	Etc.	'T'
	First char. sent	'U'
		'O'

Output of example: OUT2

MICRO Loads the CONVERS-M library of the micro corresponding to the octal number input from the terminal.

MICRO DD

DD - Octal micro number between 0 and NMICRO inclusive

NMICRO - Current number of micros in the system

This value is stored at octal location 52.

MODE Allows the mode word of a routine to be changed.

MODE NAME DATA

NAME = Name of the routine

DATA = The new mode

OCTBIN Converts the octal digits (0-7) found on the stack to a single 2-byte binary word.

	INPUT	OUTPUT*
Top of Stack:	# Char.	Low Byte
	Most Sig. Digit	High Byte
	Etc.	
	Least Sig. Digit	

* If there are too many digits or if the data is not octal, the carry flag is set (otherwise clear) and the output is:

	OUTPUT
Top of Stack:	# Char. left
	Etc.

OFF Pops one byte off of the software stack and puts it into the 'A' register.

A < = ((Stack))

(Stack) < = (Stack)+1

ON Pushes one byte from the 'A' register onto the software stack.

((Stack)-1) < = A

(Stack) < = (Stack)-1

POP2 Pops two bytes off the software stack and puts them in the 'H and L' registers.

L < = ((Stack))

H < = ((Stack)+1)

(Stack) < = (Stack)+2

PUSH2 Pushes two bytes from the 'H and L' registers onto the software stack.

((Stack)-1 < = H

((Stack)-2 < = L

(Stack) < = (Stack)-2

RUN Is a macro routine for creating program segments which reset and run the specified slave micro.

RUN 2

This example will create a program segment which will reset and run slave micro number 2.

SEARCH Searches the loaded CONVERS-M library for a match to the character string found on the stack. If a match is found the carry flag is set, otherwise it is cleared.

	INPUT	OUTPUT
Top of Stack:	# Char.	No Change
	First char. of name	
	Etc.	

SETUP Is a macro routine for creating program segments which enable specified routines on slave micros to be executed when the 'RUN' command is used.

```
SETUP    1,INIT
```

In the above example, the library of micro number 1 is searched and the address of the routine, 'INIT', is compiled into a program segment which, when run, will load that address into the initial call of micro number 1.

STRING Inputs a character string from the terminal.

	OUTPUT	EXAMPLE
Top of Stack:	# Char.	4
	First char. in	'O'
	Etc.	'U'
		'T'
		'2'

Input from terminal for example: OUT2

WHAT Outputs a question mark to the terminal.

/DI (Run only mode) - lists all the names of the commands in the CONVERS-M library which is currently loaded. The first command listed is the last command that was entered.

/FR (Run only mode) - outputs the difference between the address of the bottom of the stack and LASTAD. This gives a rough idea of the free core available. (Since the bottom of the stack will decrease during program execution, this value is larger than the actual core available.)

/KI (Run only mode) - removes the last entered command from the CONVERS-M library.

.= (Run only mode) - outputs the current value of LASTAD to the terminal. (i.e. the address at which code will next be stored.)

<< (Run only mode) - is used to begin the creation of a new command. It switches the mode to compile mode and requires the input of a command name which has not been previously defined. The command name is then entered and linked into the CONVERS-M library.

<< NAME

NAME - Command name of any length.

>> (Run only mode) - terminates the creation of an instruction by placing a return at the end of the routine and flags any undefined labels as errors. The system then returns to the run state.

B. Communicator's Basic Command List

ASM Is used by the 'ASSM' routine in the coordinator's library and has no use outside of this context.

CRT Outputs a single byte, from the 'A' register, to the terminal. It also handles the special characters: carriage return, tab, delete and page. The contents of the 'A' register are unchanged upon return.

FCMD Causes the command stored on the software stack to be sent to the floppy disk controller. If an error is encountered the carry will be set. The 'A' register indicates the state of the controller after accepting the command.

'A' Reg.	State
0	Command completed.
3	Controller ready to accept data to place in a file.
5	Controller ready to output data from a file.

FLOP1 Is used by the coordinator's 'FLOPY' routine and is of no other use to the programmer.

INIT Initializes the communicator micro. The keyboard is cleared, the floppy disk drive motor is disabled, both drives are write protected, the terminal is paged and the software stack is reset.

INPUT Is the character string input routine for the monitor. It inputs characters until it finds one of the separator characters (space, comma, tab or carriage return).

OUTPUT

Top of Stack: # Char.
Last char. typed
Etc.
First char. typed

LINE Inputs an entire line of characters from the terminal.

OUTPUT

Top of Stack: # Char.
First char. typed
Etc.
Last char. typed

If a control/C character is encountered, the software stack is cleared and the carry bit is set.

OFF See the Coordinator's Basic Command List.

ON See the Coordinator's Basic Command List.

PRINT Causes a character string to be output to the terminal.

INPUT

Top of Stack: # Char.
First char. printed
Etc.
Last char. printed

READ Reads a byte from the keyboard and places it in the 'A' register.

TOUT Sends a byte from the 'A' register to the teletype. The contents of the 'A' register are unchanged upon return.

TTY Outputs the 'A' register to the teletype. It also handles the tab and carriage return special characters.

APPENDIX C

CONVERS-M FLOW CHART

APPENDIX C

CONVERS-M FLOW CHART

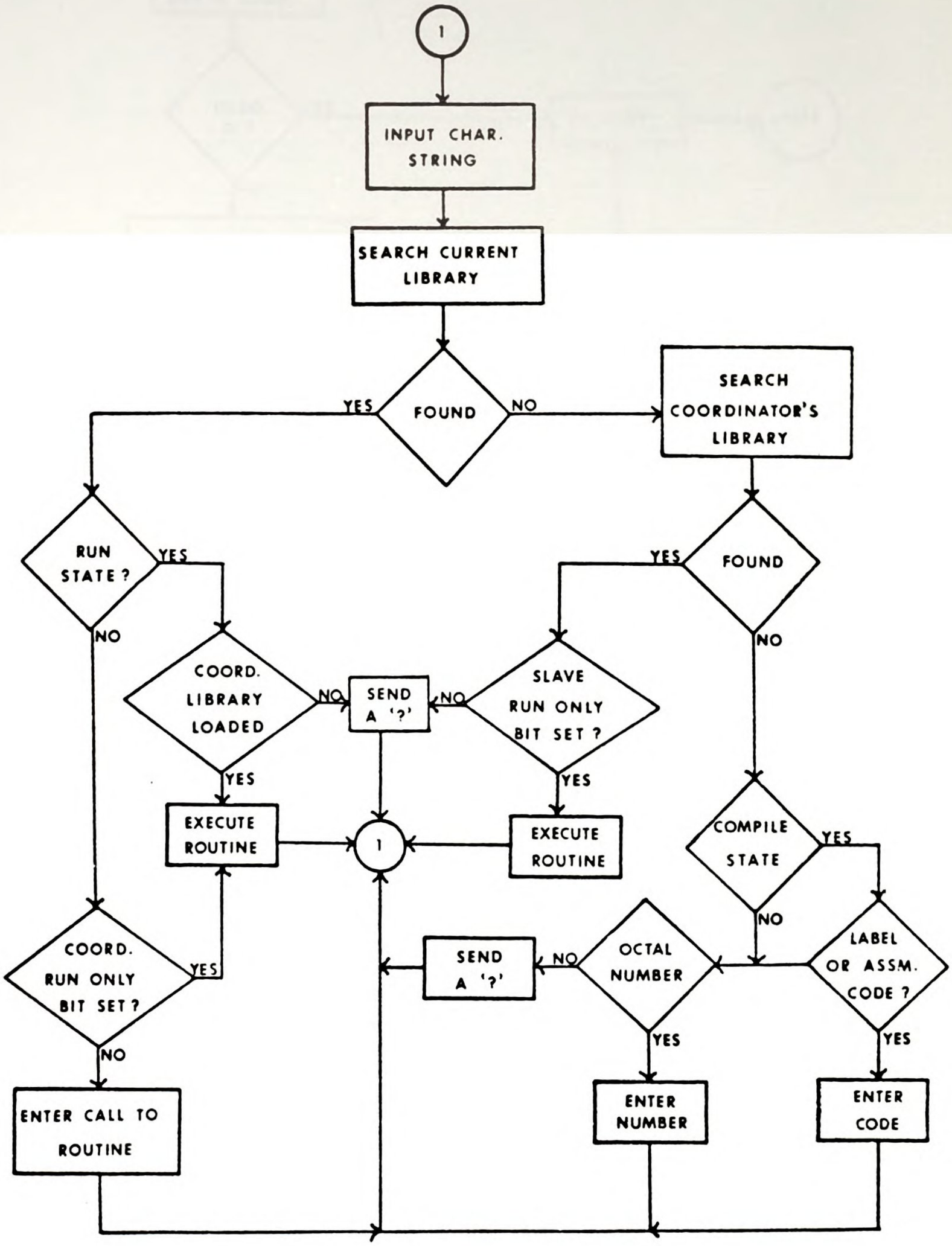


Figure C-1. Flow Chart of the CONVERS-M Command Decoding Software.

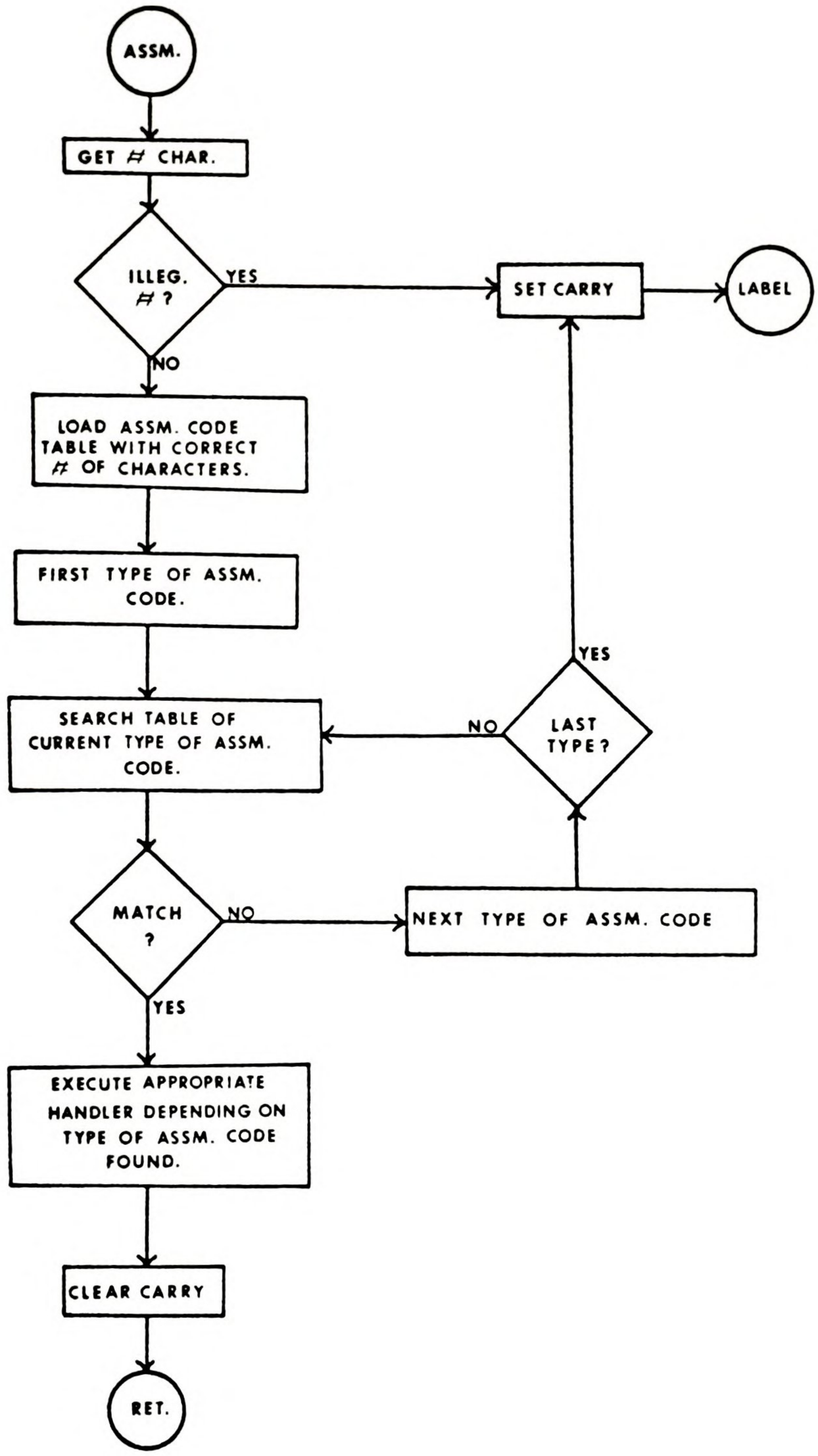


Figure C-2. Flow Chart of the CONVERS-M Assembler.

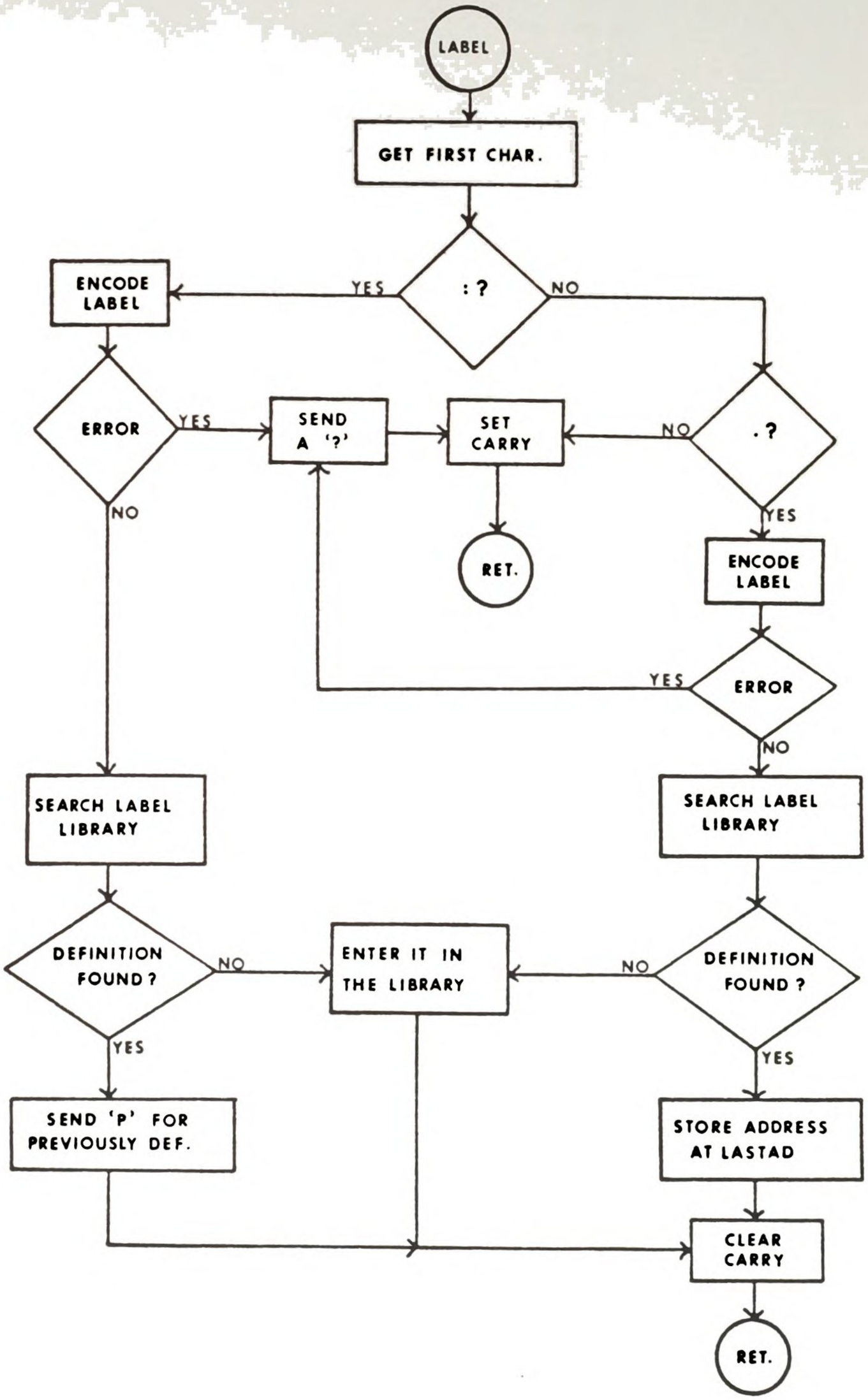


Figure C-3. Flow Chart of the Label Decoding Software.

APPENDIX D

PROGRAM LISTINGS

APPENDIX D

A. Microcomputer Programs

```
;
;
; ##############################
;
        MICRO   0
;
; ##############################
;
;
;BATCH STREAM TO INITIALIZE MICRO 2
;
;LOAD THE ROUTINE
;
<< BOOT2                                        ;BOOT MICRO 2
        MVI     A,2                             ;TWO MICROS
        STA     52
        MVI     B,106                           ;SEND 100 BYTES
        LXI     H,40000                         ;START OF MICRO 2
        LXI     D,.30                           ;START OF TABLE
:10     LDAX    D
        MOV     M,A                             ;STORE IT
        INX     H
        INX     D
        DCR     B
        JNZ     .10
        RET
;
;MICRO 2 CONVERS-M OVERHEAD
;
:30     LXI     SP,11777                        ;MICRO # 2
        CALL    0
        RST     2
        HLT
        40120                                   ;LASTAD
        40016                                   ;LINKAD
        11500                                   ;SOFTWARE STACK
        0,0
        MVI     A,1
        STA     120003                          ;ENABLE HOLD
        STA     120002                          ;SEND THE FLAG
        HLT
;
;MICRO 2 PARAMETER STORAGE AREA
;
        0                       ;33 -       - DATA0
        0,0                     ;32 - DPTS  - TOTAL POINTS IN ONE D
        1,0                     ;34 - AVGS  - NUMBER OF SCAN AVERAGES
        50,0                    ;36 - PTS   - HORIZONTAL SCAN INC.
        0,0                     ;40 - INT   - NUMBER OF INTEGRATIONS
        71,0                    ;42 - FREQ  - VERTICAL SCAN FREQUENCY
        144,0                   ;44 - BEAM  - BEAM CURRENT
        161,0                   ;46 - DELY  - ELECT. DELAY AND INT.
        0,0                     ;50 - #D    - NUMBER OF DATASETS
        0,0                     ;52 - #C    - NUMBER OF CUTS
        0,0                     ;54 - GPTS  - NUMBER OF GRAPH POINTS
        1,0                     ;56 - AVGA  - NUMBER OF ARRAY AVERAGES
        0,0                     ;60 -       - ERROR STATUS
        1,0                     ;62 - SCNS  - NUMBER OF SCANS
        0                       ;64 -       - DATA1 (POST-INT FOR LAG)
        0                       ;65 -       - DATA2 (PRE-SCANS FOR LAG
```

```
        0                       ;66 -        - DATA3 (POST-SCANS
        0                       ;67 -        - DATA4
        0,0                     ;70 - CPTS   - TOTAL POINTS IN ONE CUT
        0,0                     ;72 - OVLP   - OVERLAP POINTS PER DATA
        0,0                     ;74 - WAV    - CURRENT MONOCHROMATOR
        0,0                     ;76 -        - NEW MONO. WAVELENGHT
        51000                   ;100-        - FREE ADDRESS FOR CUTS
        0,0                     ;102- MIN    - MINIMUM
        0,0                     ;104- MAX    - MAXIMUM
        >>
;
;RUN THE ROUTINE
;
BOOT2
;
;REMOVE THE ROUTINE
;
/KI
;
;
;
;############################
;
        MICRO   2
;
;############################
;
;
<< VSET                                         ;INITIALIZE VIDICON
        PUSH    H
        PUSH    PSW
        MVI     A,1
        STA     120010                  ;2 BYTES OF PRECISION
        STA     120013                  ;SET GAIN & TURN F.A. OFF
        STA     120014                  ;INIT. F.A. COUNTERS
        LDA     44                      ;BEAM
        STA     140000                  ;BEAM CURRENT
        LDA     46                      ;DELY
        STA     140001                  ;ELECT. DELAY AND INT.
        MVI     A,13
        STA     140002                  ;TURN ON THE BEAM
        LDA     42                      ;FREQ
        STA     140004                  ;VERTICAL SCAN FREQUENCY
        STA     140003                  ;RESET FLOPS
        LDA     36                      ;PTS
        STA     140005                  ;HORIZONTAL SCAN INC.
        POP     PSW
        POP     H
        >>
<< SFLAG                                ;RETURN ON SCAN FLAG
        PUSH    PSW                     ;44 TO 58 USEC. TOTAL
:10     LDA     140011                  ;SCAN FLAG
        RAL
        JNC     .10
        STA     140010                  ;CLEAR SCAN FLAG
        POP     PSW                     ;30 TO 44 USEC. TO RETURN
        >>
<< ADDR                                         ;LOAD DATASET MAP
        PUSH    PSW
        LHLD    10                      ;LASTAD
        MOV     A,H
        ANI     77                      ;CLEAR MICRO BIT
        MOV     H,A
        POP     PSW
        >>
<< BSCAN                                ;SETUP SCAN
        PUSH    PSW
        SFLAG                           ;CLEAR SCAN FLAG
        SFLAG                           ;SYNCHRONIZE WITH SCAN
        MVI     A,21
        STA     120013                  ;SET F.A. SCAN BIT
```

```
        POP     PSW
        >>
;**********************
;
;ROUTINE TO TAKE MULTIPLE SCANS FROM THE
;VIDICON.  ALL PARAMETERS PASSED THROUGH
;REGISTERS FOR SPEED.  ON INPUT:
;
;       H&L  = SCAN ADDRESS POINTER
;       B    = SCAN AVG.
;       C    = SCAN INT.
;       A    = NUMBER OF SCANS TO TAKE
;
;ON RETURN:
;
;       H&L  = NEW SCAN ADDRESS POINTER
;       B&C  = UNCHANGED
;       D&E  = UNCHANGED
;       A    = CLEARED
;
;**********************
;
<< MSCAN                                ;TAKE MULTIPLE SCANS
        PUSH    D
:10     STA     .50                     ;NUMBER OF DATASETS
        XRA     A
        MOV     D,M                     ;HIGH BYTE OF ADDRESS
        CMP     D                       ;IF 0 - END OF MAP
        JZ      .60
        INX     H
        MOV     E,M                     ;LOW BYTE OF ADDRESS
        INX     H
        XCHG
        SHLD    120011                  ;DATASET ADDRESS
        MOV     L,C                     ;SAVE # INT.
        MOV     H,B                     ;SAVE # AVG.
:20     XRA     A
        CMP     C                       ;TEST IF INT.
        JZ      .40                     ;JUMP IF NO INT.
        MVI     A,1
        STA     120013                  ;CLEAR F.A.  SCAN BIT
        MVI     A,12
        STA     140002                  ;CLEAR BEAM SCAN BIT
:30     SFLAG                           ;INT. STARTS ON FLAG
        DCR     C                       ;TEST IF LAST INT
        JNZ     .30
        MVI     A,13
        STA     140002                  ;SET BEAM SCAN BIT
        STA     120013                  ;SET F.A.  SCAN BIT
        MOV     C,L                     ;REPLACE # OF INT
:40     SFLAG                           ;SCAN STARTS ON FLAG
        DCR     B                       ;TEST IF LAST AVG.
        JNZ     .20
        MOV     B,H                     ;REPLACE # AVG.
        XCHG
        76                              ;MVI     A,
:50     0                               ;# SCANS
        DCR     A                       ;TEST IF LAST SCAN
        JNZ     .10
:60     POP     D
        >>
<< ENDS                                         ;ROUTINE TO TERMINATE
        PUSH    PSW
        MVI     A,1
        STA     120013                  ;CLEAR F.A.  SCAN BIT
        SFLAG                           ;TERMINATES ON FLAG
        LDA     120016                  ;GET ERROR STATUS
        ANI     3
        STA     60                      ;STORE IN ERROR STATUS
        STA     120017                  ;CLEAR ERROR FLAG
```

```
        POP     PSW
        >>
<< OPEN2                                ;OPEN THE SHUTTER
        PUSH    PSW
:10     LDA     120004                  ;TEST IF ALREADY OPEN
        ANI     2
        JNZ     .20
        STA     120007
        JMP     .10
:20     POP     PSW
        >>
<< SHUT2                                ;CLOSE THE SHUTTER
        PUSH    PSW
:10     LDA     120004                  ;TEST IF ALREADY SHUT
        ANI     2
        JZ      .20
        STA     120007
        JMP     .10
:20     POP     PSW
        >>
<< SCAN2                                ;SETUP AND TAKE MULTIPLE SCAN
        LDA     56                      ;AVGA (ARRAY AVG.)
:10     PUSH    PSW
        LDA     34                      ;AVGS (SCAN AVG.)
        MOV     B,A
        LDA     40                      ;INT
        MOV     C,A
        LDA     62                      ;SCANS
        STA     120017                  ;CLEAR ERROR FLAG
        ADDR                            ;LOAD DATASET MAP
        BSCAN                           ;SETUP SCANS
        MSCAN                           ;TAKE THE SCANS
        ENDS                            ;END SCANS
        POP     PSW
        DCR     A                       ;CHECK IF LAST ARRAY AVG.
        JNZ     .10
        >>
<< LAG2                                         ;STUDY VIDICON LAG
        LDA     56                      ;AVGA (ARRAY AVG.)
:10     STA     .20
        LDA     34                      ;AVGS (SCAN AVG.)
        MOV     B,A
        LDA     40                      ;INT (PRE-INT)
        MOV     C,A
        LDA     64                      ;POST-INT
        MOV     D,A
        LDA     66                      ;POST-SCANS
        MOV     E,A
        LDA     65                      ;PRE-SCANS
        STA     120017                  ;CLEAR ERROR FLAG
        ADDR                            ;LOAD SCAN TABLE
        BSCAN                           ;SETUP SCANS
        MSCAN                           ;TAKE INIT. SCANS
        MOV     C,D                     ;POST-INT
        MOV     A,E                     ;POST-SCANS
        MSCAN                           ;TAKE FINAL SCANS
        ENDS                            ;END SCANS
        76                              ;MVI   A,
:20     0                               ;AVGA
        DCR     A
        JNZ     .10
        >>
;
;
; ##############################
;
        MICRO   1
;
; ##############################
;
;
```

```
<< R1140
:10     LDA       16000
        ANI       2
        JNZ       .10
        LDA       16300
        >>
<< W1140
        PUSH      PSW
:10     LDA       16000
        RRC
        JNC       .10
        POP       PSW
        STA       16300
        >>
;**********
;THIS ROUTINE OPENS FLOPPY FILES FOR I/O
;USING THE FLOPPY LOAD OR SAVE COMMANDS.
;AN ASCII 'S' IN THE AC OPENS THE FILE
;FOR OUTPUT.  AN ASCII 'L' IN THE AC
;OPENS THE FILE FOR INPUT.  (I.E. THE FILE
;NAME IS INPUT FROM THE KEYBOARD , THE
;AC IS PUT IN FRONT AND THE STRING IS
;SENT TO THE FLOPPY.
;**********
<< FILE
        PUSH      B
        MOV       C,A                   ;SAVE THE CHAR.
        MVI       A,52                  ;'*'
        CRT
        LINE
        JC        .50
        OFF                             ;REMOVE # CHAR.
        MOV       B,A
        MOV       A,C                   ;PUT COMMAND ON
        ON
        MOV       A,B
        INR       A
        ON
        FCMD                            ;SEND & DECODE IT
:50     POP       B
        >>
;**********
;THIS ROUTINE DECODES MESSAGES FROM THE
;11/40.  THIS RESULTS ARE:
;
;CARRY          ACC      MESSAGE FROM 11/40
;-----   ---    ---------------------
;  CLR 0        ACK/EOT - COMMAND RECEIVED & COMPLETED
;  CLR 1        ENQ/EOT - 11/40 EXPECTS DATA FROM MICRO
;  CLR 2        SOH     - 11/40 SENDING DATA TO MICRO
;  CLR 3        BEL/EOT - 11/40 EXPECTS FILESPEC FROM MICRO
;  SET 25       NAK/EOT - ERROR MESSAGE - ANY CHARS.  BETWEEN
;                         NAK & EOT ARE SENT TO THE CRT.
;-----          ---      --------------------
;
;ANY CHARS.  NOT MATCHING THE ABOVE ARE ASSUMED TO BE A MESS
;AND ARE SENT TO THE CRT, UNTIL ONE OF THE ABOVE CODES
;ARE FOUND.
;**********
<< MDEC
        PUSH      B
:10     R1140                           ;GET A CHAR.
        MVI       B,0                   ;INIT. RETURN CODE
        CPI       6                     ;'ACK'
        JZ        .20
        CPI       25                    ;'NAK'
        JZ        .50                   ;**ERROR**
        INR       B
        CPI       5                     ;'ENQ'
        JZ        .20
        INR       B
```

```
        CPI     1               ;'SOH'
        JZ      .30
        INR     B
        CPI     7               ;'BEL'
        JZ      .20
        CRT
        JMP     .10
:20     R1140                   ;NEXT CHAR.
        CPI     4               ;'EOT'
        JNZ     .60             ;**ERROR**
:30     MOV     A,B
        POP     B
        STC
        CMC
        RET
:50     R1140                   ;GET NEXT CHAR.
        CPI     4               ;'EOT'
        JZ      .60
        CRT
        JMP     .50
:60     POP     B
        MVI     A,25
        STC
        >>
;**********
;THIS ROUTINE SENDS THE CHAR STRING
;ON THE STACK TO THE 11/40 AND
;USES MDEC TO DECODE THE REPLY
;**********
<< MCMD
        PUSH    B
        OFF
        MOV     B,A
:10     OFF                     ;GET THE DATA
        W1140                   ;SEND IT
        DCR     B
        JNZ     .10
        MVI     A,4             ;'EOT'
        W1140
        POP     B
        MDEC
        >>
;**********
;THIS ROUTINE SENDS THE B REG.  TO THE 11/40
;THE 11/40'S REPLY IS DECODED AND THE
;CARRY SET IF IT IS NOT ACK/EOT
;**********
<< CKSUM
        MOV     A,B
        W1140                   ;SEND IT
        MDEC                    ;DECODE ANSWER
        ORA     A               ;ACK/EOT? (ZERO)
        RZ
        STC
        >>
;**********
;THIS IS A ROUTINE WHICH HANDLES TRANSFERS
;BETWEEN A MICRO FLOPPY AND THE 11/40.
;IT SHOULD BE LOADED INTO THE COMMUNICATOR
;**********
<< PIP
:10     MVI     A,44            ;'$'
        CRT
        LINE                    ;GET A 2 LETTER COMMAND
        RC
        OFF
        CPI     2               ;2 LETTERS?
        JZ      .20
        MOV     B,A             ;REMOVE IF NOT 2 LETTERS
:14     OFF
        DCR     B
```

```
        JNZ     .14
        MVI     A,77            ;'?'
        CRT
        JMP     .10
:20     ON                      ;REPLACE # CHAR
        MVI     A,4             ;SETUP COMMUNICATION
        STA     16000
        MVI     A,6             ;'ACK'
        W1140
        MDEC                    ;WHAT RESPONSE?
        ORA     A               ;ZERO?
        CNZ     .150            ;**ERROR** NO ACK/EOT ON STAR
:30     MCMD                    ;SEND COMMAND &DECODE RESP
        CC      .150            ;**ERROR** NAK FROM 11/40
:151    CPI     0               ;ACK/EOT ?
        JZ      .10             ;DONE
        CPI     1               ;11/40 EXPECTS DATA
        JZ      .40
        CPI     2               ;11/40 SENDING DATA
        JZ      .60
        CPI     3               ;11/40 WANTS FILESPEC
        JZ      .100
        CALL    .150            ;**ERROR** UNKNOWN CODE
:153    JMP     .10
;**********
;DATA FROM FLOPPY TO 11/40
;**********
:40     MVI     A,114           ;'L'
        FILE
        JC      .40             ;IF ERROR TRY AGAIN
:41     LXI     B,1             ;CHECKSUM=0,BYTE COUNT=1
        LXI     H,177750        ;-30
        DAD     SP              ;FIND ADDRESS FOR DATA BUFFER
        SHLD    .46
:42     CALL    1106            ;DINP
        JC      .44             ;EOF
        DCX     H
        MOV     M,A             ;STORE DATA
        INR     C
        JNZ     .42             ;377 MAX
:44     MOV     E,C
        DCR     C
        JZ      .50             ;ALL DONE
        MOV     A,C
        W1140                   ;SEND BYTE COUNT
        41
:46     0,0
        DCX     H
:47     MOV     A,M
        W1140                   ;SEND DATA
        DCX     H
        XRA     B               ;FORM CHECKSUM
        MOV     B,A
        DCR     C               ;DONE?
        JNZ     .47
        CKSUM                   ;SEND & CHECK CHECKSUM
        CC      .150            ;**ERROR** CHECHSUM ERROR
:152    XRA     A
        CMP     E               ;ZERO FOR CONTINUE
        JZ      .41
:50     XRA     A
        W1140                   ;ZERO BYTE COUNT TO TERMINATE
        JMP     .10
;**********
;DATA FROM 11/40 TO FLOPPY
;**********
:60     MVI     A,123           ;'S'
        FILE
        JC      .60             ;TRY AGAIN
:62     R1140                   ;GET BYTE COUNT
        ORA     A               ;ZERO?
```

```
        JZ      .70                     ;ALL DONE
        MOV     C,A
        MVI     B,0                     ;INITIALIZE CHECKSUM
:64     R1140                           ;GET THE DATA
        CALL    1123                    ;DOUT
        XRA     B                       ;FORM CHECKSUM
        MOV     B,A
        DCR     C                       ;DONE?
        JNZ     .64
        R1140                           ;GET CHECKSUM
        CMP     B                       ;MATCH?
        JZ      .66
        MVI     A,25                    ;DON'T MATCH!
        W1140                           ;SEND 'NAK'
        MVI     A,4                     ;'EOT'
        W1140
        CALL    1171                    ;EOTS - CLOSE FILE
        CALL    .150                    ;**ERROR** CHECKSUM ERROR
:66     MVI     A,6                     ;'ACK'
        W1140
        MVI     A,4                     ;'EOT'
        W1140
        JMP     .62                     ;NEXT
:70     CALL    1171                    ;EOTS
        JMP     .10
;**********
;FILESPEC FOR 11/40
;**********
:100    MVI     A,43                    ;'#'
        CRT
        LINE
        JC      .100
        JMP     .30                     ;GO SEND IT
;**********
;ERROR CODE ROUTINE
;**********
:150    LXI     H,.160                  ;ADDRESS TABLE
        POP     D                       ;RETURN ADDRESS
        MVI     B,5                     ;MAX. # OF ERROR CODES
:156    MOV     A,M                     ;GET LOW BYTE ADDRESS
        CMP     E                       ;MATCH?
        JZ      .157
        INX     H
        INX     H                       ;POINT TO NEXT ADDRESS
        DCR     B
        JNZ     .156
:157    MVI     A,106                   ;'F'
        CRT
        MOV     A,B
        ADI     60                      ;CONVERT TO ASCII
        CRT
        JMP     .10
:160    .153,.66,.152,.151,.30
        >>
<< DFILE                                ;OPEN DATA FILE
        PUSH    PSW
:10     MVI     A,123                   ;'S'
        FILE                            ;OPEN THE FILE
        JC      .10                     ;ERROR
        POP     PSW
        >>
<< DATAF                                ;SEND BYTE TO FLOPY
        PUSH    PSW
        LDA     7
        CALL    1123                    ;DOUT
        POP     PSW
        >>
<< AXIS                                         ;ROUTINE TO DRAW AXIS
        PUSH    PSW
        PUSH    H
        PUSH    B
```

```
        PUSH    D
        XRA     A
        STA     16203           ;CLEAR THE GRAPHICS
        STA     16201           ;X
        MVI     A,5
        STA     16202           ;Y
        LXI     H,.100          ;Y AXIS TABLE
        CALL    .4
        LXI     H,.110
        CALL    .4
        POP     D
        POP     B
        POP     H
        POP     PSW
        RET
:100    121,3,21,6,121,3,41,31
:110    21,31,241,3,41,6,241,3
:4      MVI     B,12
:10     PUSH    H
        MVI     E,4
:12     MOV     A,M
        INX     H
        MOV     C,M
        INX     H
:14     STA     16200
        DCR     C
        JNZ     .14
        DCR     E
        JNZ     .12
        POP     H
        DCR     B
        JNZ     .10
        >>
<< PLT                          ;LOW LEVEL PLOTTING ROUTINE
        PUSH    PSW
        PUSH    B
        PUSH    D
        PUSH    H
        LHLD    10              ;LASTAD
        MOV     E,M
        INX     H
        MOV     D,M             ;XINIT
        INX     H
        PUSH    D               ;SAVE XINIT
        MOV     E,M             ;XINC
        INX     H
        MOV     D,M
        INX     H
        MOV     A,M             ;NEXT Y DATA
        CMA
        STA     16202           ;Y VALUE
        XTHL
        MOV     A,H
        STA     16201           ;X VALUE
        MVI     A,1
        STA     16200           ;TURN ON POINT
        DAD     D               ;INR X
        XTHL
        POP     D
        LHLD    10              ;LASTAD
        MOV     M,E             ;UPDATE XINIT
        INX     H
        MOV     M,D
        POP     H
        POP     D
        POP     B
        POP     PSW
        >>
<< DSP                          ;GRAPHICS ON UNTIL CTRL/C
        PUSH    PSW
        MVI     A,27
```

```
        STA     16500   ;TURN ON GRAPHICS
:10     READ
        CPI     3       ;TEST FOR CTRL/C
        JNZ     .10
        MVI     A,17
        STA     16500   ;TURN ON VRAM
        POP     PSW
        >>
<< LOCIT                        ;DRAWS LINE AT SPECIFIED X
        MVI     A,27
        STA     16500           ;TURN ON GRAPHICS
        LHLD    10              ;LASTAD
        MOV     C,M             ;LOAD X POINT IN B&C
        INX     H
        MOV     B,M
        INX     H
        MOV     E,M             ;XINC
        INX     H
        MOV     D,M
        LXI     H,0             ;XINIT
:10     DAD     D
        DCX     B
        MOV     A,B
        ORA     C
        JNZ     .10
        MOV     A,H
        STA     16201           ;X VALUE
        XRA     A
        STA     16202           ;Y VALUE
        LXI     B,5             ;C=5
        MVI     A,41            ;LINE DOWN
:20     STA     16200
        DCR     C
        JNZ     .20
        DSP                     ;VRAM ON AT CTRL/C
        >>
;
;
; ##############################
;
        MICRO   O
;
; ##############################
;
;
;
;
; (*)(*)(*)(*)(*)(*)(*)(*)
;
; GENERAL PURPOSE CONVERS-M
; ROUTINES
;
; (*)(*)(*)(*)(*)(*)(*)(*)
;
;
<< RUNSL
:4      BININ
        POP2
        LDA     52
        CMP     L
        MOV     B,L
        JNC     .10
:5      WHAT
        JMP     .4
:10     LDA     47
        PUSH    PSW
        MOV     A,L
        ON
        MCHG
        STRING
        SEARCH
```

```
        CLEAN
        JC          .20
        POP         PSW
        ON
        MCHG
        JMP         .5
:20     LHLD        36
        MOV         A,H
        ANI         37
        MOV         H,A
        POP         PSW
        ON
        MCHG
        PUSH2
        MOV         A,B
        ON
        RMICRO
        >>
<< SUB2                                   ;(D&E) = (D&E) - (H&L
        PUSH        PSW
        LDAX        D
        SUB         M
        STAX        D
        INX         H
        INX         D
        LDAX        D
        SBB         M
        STAX        D
        INX         H
        JNC         .10
        DCX         D
        XRA         A
        STAX        D
        INX         D
        STAX        D
:10     INX         D
        POP         PSW
        >>
<< DIV                                              ;16-BIT BY 8-BIT
        PUSH        PSW                   ;SAVE STATUS AND REGISTERS
        PUSH        H
        PUSH        D
        PUSH        B
        OFF                               ;GET DIVISOR
        MOV         B,A
        POP2                              ;GET DIVIDEND
        MVI         C,21                  ;BIT COUNTER
        XRA         A
:10     MOV         D,A
        MOV         A,L
        RAL                               ;ROTATE LOW BYTE
        MOV         L,A
        MOV         A,H
        RAL                               ;ROTATE HIGH BYTE
        MOV         H,A
        DCR         C                     ;DECREMENT COUNTER
        JZ          .30
        MOV         A,D
        RAL
        JNC         .20
        SUB         B
        JMP         .10
:20     SUB         B
        JNC         .10
        ADD         B
        JMP         .10
:30     MOV         A,H                   ;COMPLEMENT HIGH BYTE
        CMA
        MOV         H,A
        MOV         A,L
        CMA                               ;COMPLEMENT LOW BYTE
```

```
        MOV     L,A
        PUSH2                   ;QUOTIENT ON STACK
        MOV     A,D
        ON                      ;REMAINDER ON STACK
        POP     B               ;RESTORE STATUS AND REGISTERS
        POP     D
        POP     H
        POP     PSW
>>
<< DIV3                                 ;24-BIT X 16-BIT
        PUSH    PSW             ;SAVE STATUS AND REGISTERS
        PUSH    H
        PUSH    D
        PUSH    B
        POP2                    ;GET DIVISOR
        MOV     C,L             ;LOW BYTE TO REG C
        MOV     B,H             ;HIGH BYTE TO REG B
        POP2                    ;ADDRESS OF 3-BYTE DIVIDEND
        MVI     A,31            ;BIT COUNT
        STA     .25
        XRA     A
        MOV     E,A
:10     MOV     D,A
        MOV     A,M             ;ROTATE DIVIDEND LEFT ONE BIT
        RAL
        MOV     M,A
        INX     H
        MOV     A,M
        RAL
        MOV     M,A
        INX     H
        MOV     A,M
        RAL                     ;MSB OF DIVIDEND INTO CARRY
        MOV     M,A
        DCX     H
        DCX     H
        LDA     .25
        DCR     A               ;DECREMENT BIT COUNT
        JZ      .30             ;IF A=0, JUMP TO EXIT
        STA     .25             ;ELSE STORE COUNT AND CONT
        MOV     A,E
        RAL                     ;MSB OF DIVIDEND INTO REM
        MOV     E,A
        MOV     A,D
        RAL
        MOV     D,A
        JNC     .20             ;JUMP IF BORROW ON SUBTRACT
        MOV     A,E
        SUB     C               ;SUBTRACT LOWER BYTES
        MOV     E,A
        MOV     A,D
        SBB     B               ;SUBTRACT HIGH BYTES
        STC
        CMC
        JMP     .10             ;LOOP
:20     MOV     A,E
        SUB     C               ;SUBTRACT LOW BYTES
        MOV     E,A
        MOV     A,D
        SBB     B               ;SUBTRACT HIGH BYTES
        JNC     .10             ;JUMP IF NO BORROW
        MOV     D,A             ;RESTORE D
        MOV     A,E             ;ELSE ADD DIVISOR BACK
        ADD     C
        MOV     E,A
        MOV     A,D
        ADC     B
        JMP     .10             ;AND LOOP
:25     000                     ;STORAGE LOCATION FOR BIT
:30     MOV     A,M             ;GET QUOTIENT AND COMPLEMENT
        CMA
```

```
        MOV     M,A
        INX     H
        MOV     A,M
        CMA
        MOV     M,A
        INX     H
        MOV     A,M
        CMA
        MOV     M,A
        INX     H
        PUSH2                           ;ADDRESS OF NEXT DATA WORD
        POP     B                       ;RESTORE STATUS AND REGISTERS
        POP     D
        POP     H
        POP     PSW
>>
;**********************
;
;ROUTINE TO INPUT ASCII TO LASTAD
;FORM:
;       '' STRING
;
;THE ABOVE EXAMPLE WOULD CAUSE THE
;ASCII REPRESENTATION OF STRING TO
;BE STORED AT LASTAD PRECEEDED
;BY THE NUMBER OF CHARACTERS IN THE
;STRING.  THE STRING IS STORED FIRST
;CHARACTER FIRST.
;
;**********************
;
<< ''                                   ;INPUT ASCII TO LASTAD
        MODE    '',300                  ;CHANGE TO RUN ONLY MODE
        STRING                          ;GET THE CHARACTER STRING
        OFF
        MOV     C,A                     ;# CHAR.
        LHLD    32                      ;LASTAD
        MOV     M,A
        INX     H
:10     OFF
        MOV     M,A                     ;STORE NEXT CHAR.
        INX     H
        DCR     C
        JNZ     .10
        SHLD    32                      ;UPDATE LASTAD
        >>
<< BINDEC                               ;16-BIT BINARY TO 5 DIGIT BCD
        PUSH    H
        PUSH    B
        PUSH    PSW
        MVI     C,5                     ;INITIALIZE BCD DIGIT COUNTER
        MOV     B,C
:10     MVI     A,12                    ;DIVISOR FO 10
        ON
        DIV                             ;DIVIDE BINARY NUMBER BY 10
        OFF                             ;GET REMAINDER
        PUSH    PSW                     ;AND SAVE
        DCR     C                       ;DECREMENT BCD COUNT
        JNZ     .10                     ;JUMP IF MORE LEFT
        POP2                            ;GET QUOTIENT
        MOV     C,B                     ;REINITIALIZE BCD COUNTER
:15     POP     PSW                     ;RECOVER REMAINDER
        CPI     0                       ;IS IT A ZERO?
        JNZ     .25                     ;NO, REST OF DIGITS ARE SIGN
        DCR     B                       ;YES, DECREMENT COUNT
        DCR     C
        JNZ     .15                     ;JUMP IF MORE TO DO
        INR     C
        INR     B
        JMP     .25
:20     POP     PSW
```

```
:25     ON                          ;PUT REMAINDER ON STACK
        DCR     C
        JNZ     .20
        MOV     A,B                 ;GET NUMBER OF NON-ZERO DIGIT
        ON                          ;AND PUT ON STACK
        POP     PSW
        POP     B
        POP     H
>>

<< DECBIN                             ;DECIMAL TO BINARY CONVER
        PUSH    H
        PUSH    B
        PUSH    D
        PUSH    PSW
        OFF                         ;# CHARACTERS
        MOV     B,A
        LXI     H,0                 ;INITIALIZE THE BUFFER
:10     MVI     D,0
        OFF                         ;GET A DIGIT
        MOV     E,A
        CPI     12                  ;IS IT BCD?
        JNC     .30                 ;JUMP IF ERROR
        DAD     D                   ;ADD THE DIGIT TO THE BUFF
        JC      .30                 ;TOO LARGE IF CARRY
        DCR     B                   ;LAST DIGIT?
        JZ      .20                 ;JUMP IF DONE
        MVI     C,11                ;IF NOT THEN BUFFER X 10
        MOV     E,L
        MOV     D,H
:15     DAD     D                   ;ADD 10 TIMES FOR MULTIPLY
        JC      .40                 ;TOO LARGE IF CARRY
        DCR     C                   ;ADDED 10 TIMES?
        JNZ     .15
        JMP     .10                 ;BACK TO GET A DIGIT!!
:20     PUSH2                       ;PUT THE RESULT ON THE STAC
        POP     PSW
        POP     D
        POP     B
        POP     H
        STC
        CMC                         ;CLEAR THE CARRY (NO ERROR)
        RET
:30     DCR     B
:40     MOV     A,B
        ON                          ;# CHARACTERS LEFT
        POP     PSW
        POP     D
        POP     B
        POP     H
        STC                           ;SET THE CARRY (ERROR FLAG)
        >>                          << DECOUT

        PUSH    PSW                 ;CONVERT BINARY NUMBER TO DEC
        BINDEC
        MVI     A,1
        ON
        ASCII                       ;CONVERT TO ASCII CHARACTERS
        MESSAG                      ;OUTPUT
        POP     PSW
>>
<< DECIN                              ;INPUT A DECIMAL #
        PUSH    PSW
:10     STRING                      ;INPUT ASCII STRING
        XRA     A
        ON                          ;ASCII => OCTAL MODE
        ASCII                       ;CONVERT TO BCD
        DECBIN                      ;CONVERT TO BINARY
        JC      .20                 ;ERROR FLAG SET?
        POP     PSW
        RET
```

```
:20     CLEAN                           ;REMOVE GARBAGE
        WHAT                            ;SEND A '?' TO TTY
        JMP     .10                     ;TRY AGAIN
        >>
<< RUN2                                         ;RUN VIDICON MICRO
        PUSH    PSW
        RUN     2
:10     FLAG    2
        JZ      .10
        POP     PSW
        >>
;
;
;(*)(*)(*)(*)(*)(*)(*)(*)
;
;ROUTINES TO INTIIALIZE AND
;ACCESS DATA MEMORY
;
;(*)(*)(*)(*)(*)(*)(*)(*)
;
<< MINIT                                ;INITIALIZE DATA & CUT MEMORY
        LXI     H,0
        SHLD    40050                   ;#D
        SHLD    40052                   ;#C
        SHLD    40054                   ;GPTS
        SHLD    40032                   ;DPTS
        SHLD    40070                   ;CPTS
        LXI     H,51000                 ;TOP OF CUT MEMORY
        SHLD    40100                   ;FREE ADDRESS - CUTS
        LHLD    40010                   ;LASTAD
        MVI     M,0                     ;ZERO DATASET MAP
        >>
<< DMEM                                         ;DMEM  DPTS  OVLP
        DECIN                           ;DPTS
        DECIN                           ;OVLP
        POP2
        SHLD    40072                   ;OVLP
        POP2
        SHLD    40032                   ;DPTS
        MOV     E,L
        MOV     D,H
        DAD     D                       ;2 BYTES PER POINT
        SHLD    .30
        XCHG
        LHLD    40010                   ;LASTAD
        SHLD    .20
        LXI     H,77777                 ;END OF DATA MEMORY
:10     MOV     A,H
        CPI     140                     ;BOTTOM OF DATA MEMORY
        RC
        XRA     A
        MOV     M,A
        DCX     H
        DCX     D
        MOV     A,D
        ORA     E
        JNZ     .10                     ;LAST POINT?
        INX     H
        XCHG
        41                              ;LXI  H,
:20     0,0                             ;LASTAD TABLE
        MOV     A,D
        ANI     77                      ;REMOVE MICRO BIT
        MOV     M,A
        INX     H
        MOV     M,E                     ;STORE LOW BYTE
        INX     H
        MVI     M,0                     ;END OF TABLE MARK
        SHLD    .20
        LXI     H,40050                 ;#D (# OF DATASETS)
```

```
        INR     M
        LHLD    40072           ;OVLP   (OVERLAP OF DATASETS)
        XCHG
        DCX     H
        DAD     D
        DAD     D               ;2 BYTES PER POINT
        21                      ;LXI  D,
:30     0,0                     ;DPTS * 2
        JMP     .10
        >>
<< MCUT                                 ;MCUT  #
        DECIN                   ;#
        POP2
        MOV     C,L
        LHLD    40050           ;#D (# OF DATASETS)
        SHLD    40070           ;CPTS (# OF POINTS PER CUT)
        MOV     E,L
        MOV     D,H
        DAD     D               ;2 BYTES PER POINT
        SHLD    .50             ;TOTAL # OF BYTES
        XCHG
        LHLD    40010           ;LASTAD
        XRA     A
:10     CMP     M               ;SKIP OVER DATASET MAP
        JZ      .20
        INX     H
        INX     H
        JMP     .10
:20     INX     H
        SHLD    .40             ;ADDRESS OF CUT MAP
        LHLD    40100           ;FREE ADDRESS
:30     XRA     A
        MOV     M,A
        DCX     H
        DCX     D
        MOV     A,D
        ORA     E               ;TEST IF ZERO
        JNZ     .30
        INX     H
        XCHG
        41                      ;LXI  H,
:40     0,0                     ;ADDRESS OF CUT MAP
        MOV     M,D             ;HIGH BYTE FIRST
        INX     H
        MOV     M,E             ;LOW BYTE
        INX     H
        MVI     M,0             ;MOVE END OF DATA MARK
        SHLD    .40
        LXI     H,40052
        INR     M               ;INC.  # OF CUTS
        XCHG
        21                      ;LXI  D,
:50     0,0                     ;# OF BYTES
        DCX     H
        DCR     C               ;# OF CUTS TO CREATE
        JNZ     .30
        SHLD    40100           ;NEW FREE ADDRESS
        >>
<< TAB                                  ;PARAMETER TABLE
        '' AVGS         40034
        '' AVGA         40056
        '' INT          40040
        '' PTS          40036
        '' FREQ         40042
        '' BEAM         40044
        '' DELY         40046
        '' DPTS         40032
        '' CPTS         40070
        '' GPTS         40054
        '' SCNS         40062
        '' #D           40050
```

```
        '' #C                   40052
        '' OVLP                 40072
        '' WAV                  40074
        '' MIN                  40102
        '' MAX                  40104
        '' END                  0,0
        0
        >>
<< CHECK                                ;RETURNS WITH PARAMETER
        PUSH    B
        PUSH    D
        PUSH    PSW
        LHLD    50                      ;STACK
        XCHG
        MVI     A,41
        STA     .10                     ;ADDRESS OF PARAMETER TABLE
:10     TAB                             ;LXI  H,TAB
        PUSH    D                       ;SAVE STACK
:40     XRA     A
        CMP     M
        JZ      .60                     ;END OF TABLE
        MOV     C,M                     ;# CHAR.
        INR     C
        POP     D
        PUSH    D                       ;RESTORE STACK
        CALL    361                     ;LCMP
        JZ      .50                     ;MATCH
        INX     H                       ;JUMP OVER ADDRESS
        INX     H
        JMP     .40
:50     MOV     E,M                     ;GET THE ADDRESS
        INX     H
        MOV     D,M
        XCHG
        POP     D                       ;GET RID OF STACK
        POP     PSW
        POP     D
        POP     B
        STC                             ;CLEAR CARRY - FOUND
        CMC
        CLEAN
        RET
:60     POP     D
        POP     PSW
        POP     D
        POP     B
        STC                             ;SET CARRY - NOT FOUND
        CLEAN
        WHAT
        >>
<< POUT                                 ;PARAMETER OUTPUT ROUTINE
        PUSH    PSW
        PUSH    H
        PUSH    D
        PUSH    B
        MVI     A,41
        STA     .10
:10     TAB                     ;LXI  H,TAB  -  PARAMETER TABLE
        MVI     A,33            ;PAGE
        CALL    .50
:20     MOV     C,M
        XRA     A
        CMP     C
        JZ      .60
        MOV     B,C
:30     INX     H
        MOV     A,M
        ON                      ;PARAM.  NAME TO CRT
        DCR     B
        JNZ     .30
        MOV     A,C             ;# CHAR.
```

```
ON
MESSAG
MVI     A,40    ;SPACE
CALL    .50
INX     H
MOV     E,M     ;ADDRESS OF DATA
INX     H
MOV     D,M
INX     H
PUSH    H       ;SAVE POINTER
XCHG
MOV     E,M     ;GET THE DATA
INX     H
MOV     D,M
XCHG
PUSH2
DECOUT          ;OUTPUT THE PARAMETER
POP     H
MVI     A,11    ;TAB
CALL    .50
JMP     .20
:50     ON
        MVI     A,1
        ON
        MESSAG
        RET
:60     POP     B
        POP     D
        POP     H
        POP     PSW
        >>
<< SET                          ;ROUTINE TO SETUP VIDICON
        PUSH    H
        PUSH    PSW
:10     41,1,52         ;1,*
        PUSH2
        MESSAG
        STRING
        CHECK
        JC      .10     ;ERROR
        MOV     A,H     ;ALL DONE
        ORA     L
        JZ      .20     ;ALL DONE IF ZERO
        DECIN           ;GET THE VALUE
        XCHG
        POP2
        XCHG
        MOV     M,E
        INX     H
        MOV     M,D     ;STORE IT
        JMP     .10     ;NEXT PARAMETER
:20     POP     PSW
        POP     H
        >>
;
;
;(*)(*)(*)(*)(*)(*)(*)(*)
;
;DATA FILE MANIPULATION
;AND STORAGE ROUTINES
;
;(*)(*)(*)(*)(*)(*)(*)(*)
;
;
<< CFIND                        ;CFIND  NUMBER
        PUSH    PSW
        PUSH    H
        PUSH    D
:10     DECIN
        LDA     40052           ;#C (NUMBER OF CUTS)
        POP2
```

```
        CMP     L
        JNC     .20
        WHAT
        JMP     .10
:20     XCHG
        LHLD    40010           ;LASTAD
        XRA     A
:30     CMP     M               ;SKIP OVER DATASET MAP
        JZ      .40
        INX     H
        INX     H
        JMP     .30
:40     INX     H
        MOV     A,E
        DCR     A
        RAL
        MOV     E,A
        DAD     D               ;POINT TO CUT ADDRESS
        MOV     D,M
        INX     H
        MOV     E,M
        XCHG
        PUSH2                   ;CUT ADDRESS TO STACK
        LHLD    40070
        SHLD    40054           ;GPTS = CPTS
        POP     D
        POP     H
        POP     PSW
        >>
<< DFIND                        ;DFIND  NUMBER
        PUSH    PSW
        PUSH    H
        PUSH    D
:10     DECIN                   ;NUMBER OF THE DATASET
        LDA     40050           ;CURRENT NUMBER OF DATASETS
        POP2
        CMP     L
        JNC     .20
        WHAT
        JMP     .10
:20     MOV     A,L
        DCR     A
        RAL
        MOV     E,A
        MVI     D,0
        LHLD    40010           ;LASTAD
        DAD     D               ;POINT TO DATASET ADDRESS
        INX     H
        MOV     E,M             ;GET LOW BYTE OF ADDRESS
        DCX     H
        MOV     A,M             ;HIGH BYTE OF ADDRESS
        ORI     140             ;MICRO ADDRESS 3
        MOV     D,A
        LHLD    40072           ;OVLP (DATASET OVERLAP)
        XCHG
        DAD     D
        DAD     D               ;2 BYTES PER POINT
        PUSH2                   ;DATASET ADDRESS TO STACK
        LHLD    40032           ;DPTS (POINTS PER DATASET)
        MOV     A,L
        SUB     E
        MOV     L,A
        MOV     A,H
        SBB     D
        MOV     H,A
        SHLD    40054           ;GPTS = DPTS - OVLP
        POP     D
        POP     H
        POP     PSW
        >>
<< TFIND                        ;TFIND  TYPE,NUMBER
```

```
        PUSH    PSW
        PUSH    H
:10     STRING                  ;GET THE TYPE
        POP2                    ;C = CUT (CFIND) ;  D = DATA
        PUSH2
        CLEAN
        DCR     L               ;SHOULD BE 1
        JZ      .30
:20     WHAT
        JMP     .10
:30     MOV     A,H
        CPI     104             ;D
        JNZ     .40
        DFIND
        JMP     .50
:40     CPI     103             ;C
        JNZ     .20
        CFIND
:50     POP     H
        POP     PSW
        >>
<< CUT                                  ;CUT  DST  POINT
        LHLD    40010           ;LASTAD
        MOV     A,M
        ORI     140             ;REPLACE MICRO ADDRESS BITS
        MOV     D,A
        INX     H
        MOV     E,M             ;D&E = FIRST DATASET
        LHLD    40032           ;DPTS (POINTS PER DATASET)
        MOV     A,L
        CMA
        MOV     C,A
        MOV     A,H
        CMA
        MOV     B,A
        INX     B               ;B&C = -DPTS
        CFIND                   ;INPUT DESTINATION
        DECIN                   ;POINT
        POP2
        DCX     H
        XCHG
        DAD     D               ;H&L POINT TO SOURCE
        DAD     D               ;2 BYTES PER POINT
        XCHG
        POP2
        XCHG                    ;D&E POINT TO DESTINATION
        LDA     40050           ;#D (# OF DATASETS)
:10     PUSH    PSW
        MOV     A,M
        STAX    D
        INX     H
        INX     D
        MOV     A,M
        STAX    D
        INX     D
        DCX     H
        DAD     B               ;POINT TO NEXT DATASET
        DAD     B               ;2 BYTES PER POINT
        POP     PSW
        DCR     A               ;CHECK IF LAST DATASET
        JNZ     .10
        LHLD    40050
        SHLD    40070           ;CPTS = #D
        >>
<< STORE                        ;STORE DATASET ON FLOPY
        TFIND                   ;DATASET ADDRESS
        SETUP   1,DFILE         ;OPEN THE FILE
        CALL    .60             ;RUN IT
        SETUP   1,DATAF         ;DATA BYTE TO FLOPY
        LXI     H,40032         ;PARAMETER STORAGE
        MVI     B,44            ;# BYTES
```

```
:20     MOV     A,M             ;GET A BYTE
        INX     H
        CALL    .50             ;STORE IT
        DCR     B
        JNZ     .20
        LHLD    40054           ;GPTS
        XCHG
        POP2                    ;DATASET ADDRESS => H&L
:30     MOV     A,M             ;LOW BYTE OF DATA
        INX     H
        CALL    .50             ;SEND LOW BYTE TO THE FLOPPY
        MOV     A,M             ;HIGH BYTE OF THE DATA
        INX     H
        CALL    .50             ;SEND HIGH BYTE TO THE FLOPPY
        DCX     D               ;TEST IF END OF DATA
        MOV     A,D
        ORA     E
        JNZ     .30
        LXI     H,1171          ;SETUP  1,EOTS
        SHLD    20004
        CALL    .60
        MVI     A,166           ;REPLACE THE HLT
        STA     20007
        RET
:50     STA     20007           ;DATA BUFFER
:60     RUN     1
:64     FLAG    1               ;MICRO DONE YET
        JZ      .64
        >>
<< TSTORE                       ;STORE ALL DATASETS
        SETUP   1,DFILE         ;OPEN THE FILE
        CALL    .60
        SETUP   1,DATAF
        LXI     H,40032         ;START OF PARAMETERS
        MVI     C,44
:10     MOV     A,M             ;GET A BYTE
        INX     H
        CALL    .50             ;SEND IT TO THE FLOPPY
        DCR     C
        JNZ     .10
        LHLD    40072           ;OVLP
        MOV     C,L
        MOV     B,H             ;B&C = OVERLAP
        LHLD    40032           ;DPTS
        MOV     A,L
        SUB     C
        MOV     L,A
        MOV     A,H
        SBB     B
        MOV     H,A
        SHLD    .30             ;#POINTS = DPTS - OVLP
        LHLD    40010           ;LASTAD
        SHLD    .24             ;ADDRESS POINTER
:20     41                      ;LXI    H,ADDRESS POHINTER
:24     0,0                     ;ADDRESS POINTER
        MOV     A,M             ;HIGH BYTE OF DATASET ADDRESS
        CPI     0               ;0 IF END OF DATASETS
        JZ      .40             ;END OF DATASETS
        ORI     140             ;REPLACE MICRO ADDRESS BITS
        MOV     D,A             ;HIGH BYTE OF DATASET ADDRESS
        INX     H
        MOV     E,M             ;LOW BYTE OF DATASET ADDRESS
        INX     H
        SHLD    .24             ;REPLACE ADDRESS POINTER
        XCHG
        DAD     B
        DAD     B               ;H&L = DATASET ADDRESS + 2
        21                      ;LXI    D,#POINTS
:30     0,0                     ;#POINTS
:34     MOV     A,M             ;LOW BYTE OF DATA
        CALL    .50             ;SEND LOW BYTE TO FLOPPY
```

```
        INX     H
        MOV     A,M             ;HIGH BYTE OF DATA
        CALL    .50             ;SEND HIGH BYTE TO FLOPPY
        INX     H
        DCX     D               ;TEST IF LAST POINT
        MOV     A,D
        ORA     E
        JNZ     .34
        JMP     .20             ;NEXT DATASET
:40     LXI     H,1171          ;SETUP  1,EOTS
        SHLD    20004
        CALL    .60
        MVI     A,166           ;REPLACE HLT ON MICRO 1
        STA     20007
        RET
:50     STA     20007           ;DATA BUFFER
:60     RUN     1
:70     FLAG    1
        JZ      .70
        >>
<< MAX                          ;MAX  TYPE,NUMBER
        TFIND
        LHLD    40054           ;GPTS
        MOV     C,L
        MOV     B,H             ;B&C = GPTS
        POP2                    ;H&L = ADDRESS
        MOV     E,M
        INX     H
        MOV     D,M
        XCHG
        SHLD    .110            ;MIN
        SHLD    .120            ;MAX
        XCHG
:10     LXI     D,.124          ;HIGH BYTE OF MAX
        INX     H
        INX     H               ;POINT TO NEXT DATA POINT
        DCX     B               ;TEST IF LAST POINT
        MOV     A,B
        ORA     C
        JZ      .30
        CALL    .100            ;CHECK FOR NEW MAX
        JC      .20             ;JUMP IF NEW MAX
        DCX     D               ;POINT TO MIN
        DCX     D
        CALL    .100
        JC      .10             ;JUMP IF NOT NEW MIN
:20     DCX     H
        DCX     D
        MOV     A,M             ;GET LOW BYTE
        STAX    D
        INX     H
        INX     D
        MOV     A,M             ;GET THE HIGH BYTE
        STAX    D
        JMP     .10
:30     LHLD    .110
        SHLD    40102           ;MIN
        LHLD    .120
        SHLD    40104           ;MAX
        RET
:100    LDAX    D
        CMP     M
        RNZ
        DCX     H
        DCX     D
        LDAX    D
        CMP     M
        INX     H
        INX     D
        RET
:110    0,0                     ;MIN
```

```
:120    0                                   ;MAX
:124    0
        >>
<< RBACK                        ;ROUTINE TO SUBTRACT TWO DATA SETS
        PUSH    PSW
        PUSH    H
        PUSH    D
        POP2                    ;DST
        XCHG
        LHLD    40054           ;GPTS
        PUSH    H
        POP2                    ;SRC
:10     SUB2
        XTHL                    ;# PTS.
        DCX     H               ;DONE
        MOV     A,H
        ORA     L
        XTHL
        JNZ     .10
        POP     H
        POP     D
        POP     H
        POP     PSW
        >>
<< BACK                                         ;BACK  DST,SRC
        TFIND                           ;(DST) = (DST) - (SRC)
        POP2
        TFIND
        PUSH2
        RBACK
        >>
<< RCLEAR                       ;CLEARS SCAN MEMORY
        PUSH    PSW
        PUSH    H
        PUSH    D
        PUSH    B
        LHLD    40054           ;GPTS
        XCHG
        POP2                    ;DATASET ADDRESS
        MVI     B,2             ;2 BYTES
:10     MOV     C,B
        XRA     A
:20     MOV     M,A             ;CLEAR IT
        INX     H               ;NEXT LOCATION
        DCR     C
        JNZ     .20
        DCX     D
        MOV     A,D
        ORA     E
        JNZ     .10
        POP     B
        POP     D
        POP     H
        POP     PSW
        >>
<< CLEAR                                        ;CLEAR  TYPE,NUMBER
        TFIND                                   ;GET DATASET ADDRESS
        RCLEAR
        >>
<< CLR8K                                        ;CLEARS ALL 8K OF DATA MEMORY
        PUSH    PSW
        PUSH    H
        LXI     H,60000                 ;START OF DATA MEMORY
:10     XRA     A
        MOV     M,A
        INX     H
        MOV     A,H
        CPI     200                     ;END OF DATA MEMORY
        JC      .10
        POP     H
        POP     PSW
```

```
        >>
<< MPLOT                                    ;PLOTTING ROUTINE
        TFIND
        POP2
        SHLD    .24                 ;SRC
        SETUP   1,PLT               ;PLOT HANDLER
        LXI     H,.100              ;BUFFER
        PUSH2
        XRA     A
        MOV     M,A                 ;X SCALING
        INX     H
        MOV     M,A
        INX     H
        MVI     M,372
        LHLD    40054               ;GPTS
        PUSH2
        SHLD    .50
        DIV3                        ;250/GPTS
        POP2                        ;REMOVE ADDRESS
        LHLD    20010               ;LASTAD
        MOV     M,A                 ;XINIT = 0
        INX     H
        MOV     M,A
        INX     H
        LXI     D,.100
        INX     D                   ;X SCALE HIGH BYTES
        LDAX    D
        MOV     M,A
        INX     D
        INX     H
        LDAX    D
        MOV     M,A
        INX     H
        SHLD    .44                 ;DESTINATION
        LHLD    40104               ;MAX
        SHLD    .100
        LXI     D,.100
        LXI     H,40102
        SUB2                        ;SCALE = MAX - MIN
        LHLD    .100
        MVI     E,0
:10     INR     E
        MOV     A,L                 ;ROTATE SCALE UNTIL CARRY
        RAL
        MOV     L,A
        MOV     A,H
        RAL
        MOV     H,A
        JNC     .10
        MOV     A,E
        INR     A
        STA     .30                 ;SHIFT
        LHLD    40102               ;MIN
        MOV     A,L
        CMA
        MOV     C,A
        MOV     A,H
        CMA
        MOV     B,A
        INX     B                   ;B&C = -MIN
:20     41                          ;LXI    H,SRS
:24     0,0                         ;SRC
        MOV     E,M
        INX     H
        MOV     D,M
        INX     H
        SHLD    .24                 ;SRC
        XCHG
        DAD     B                   ;DATA = DATA - MIN
        76                          ;MVI    A,SHIFT
:30     0                           ;SHIFT
```

```
:34     DCR     A
        JZ      .40
        MOV     E,L
        MOV     D,H
        DAD     D
        JMP     .34
:40     MOV     A,D
        62                              ;STA     DST
:44     0,0                             ;DST
        CALL    .60                     ;PLOT IT
        41                              ;LXI     H,POINTS
:50     0,0                             ;POINTS
        DCX     H
        SHLD    .50
        MOV     A,H
        ORA     L
        JNZ     .20
        SETUP   1,DSP                   ;TURN ON GRAPH,  CTRL/C ENDS
:60     RUN     1
:64     FLAG    1
        JZ      .64
        RET
:100    0,0
        >>
<< PLOT                                         ;ROUTINE TO DRAW AXIS
        SETUP   1,AXIS                  ;DRAW THE AXIS
        RUN     1
:10     FLAG    1
        JZ      .10
        MPLOT                           ;PLOT THE DATA
        >>
<< LOC                                          ;LOCATES SPECIFIED
        DECIN                           ;RELATIVE LOCATION (1-NPTS)
        POP2
        XCHG
        DCX     D
        LHLD    20010                   ;MICRO 1 LASTAD
        MOV     M,E
        INX     H
        MOV     M,D
        SETUP   1,LOCIT
        RUN     1
:10     FLAG    1
        JZ      .10
        >>
;
;
; (*)(*)(*)(*)(*)(*)(*)(*)
;
;HIGH LEVEL VIDICON
;SPECTROPHOTOMETER ROUTINES
;
; (*)(*)(*)(*)(*)(*)(*)(*)
;
;
<< OPEN                                 ;OPEN THE SHUTTER
        PUSH    PSW
        SETUP   2,OPEN2
        RUN     2
:10     FLAG    2
        JZ      .10
        POP     PSW
        >>
<< SHUT                                 ;CLOSE THE SHUTTER
        PUSH    PSW
        SETUP   2,SHUT2
        RUN     2
:10     FLAG    2
        JZ      .10
        POP     PSW
        >>
```

```
<< VINIT                                    ;INITIALIZE VIDICON
        SETUP   2,VSET
        RUN2
        >>
<< VERR                                     ;OUTPUTS VIDICON ERR
        PUSH    PSW
        PUSH    H
        LHLD    40060                   ;VIDICON ERROR STATUS
        XRA     A
        CMP     L
        JZ      .10
        PUSH2
        BINOUT
:10     POP     H
        POP     PSW
        >>
<< SCAN                                     ;TAKE A SET OF SCANS
        SETUP   2,SCAN2
        RUN2
        VERR
        >>
<< LAG                                      ;LAG PRE-INT,POST-INT,
        DECIN                           ;PRE-INTEGRATIONS
        POP2
        MOV     A,L
        STA     40040                   ;INT
        LXI     H,40064                 ;DATA STORAGE
        MVI     C,3
:10     DECIN
        XCHG
        POP2
        XCHG
        MOV     M,E
        INX     H
        DCR     C
        JNZ     .10
        SETUP   2,LAG2
        RUN2
        VERR
        >>
;
;
;#############################
;
        MICRO   0
;
;#############################
;
;
<< BOOT4                                    ;INITIALIZE MICRO #4
        MVI     A,4                     ;4 MICROS
        STA     52
        MVI     B,17                    ;SEND 17 BYTES
        LXI     H,100000                ;START OF MICRO 4
        LXI     D,.30
:10     LDAX    D
        MOV     M,A
        INX     H
        INX     D
        DCR     B
        JNZ     .10
        RET
:30     LXI     SP,3777                 ;MICRO #4
        CALL    0
        EI
        HLT
        100050                  ;10 - LASTAD
        100016                  ;12 - LINKAD
        105300                  ;14 - SOFTWARE STACK
        0                       ;16 - END OF LIBRARY
        0                       ;17 - # TIMES TO FLUSH
```

```
        0                       ;20 - # VEL. PTS
        >>
BOOT4
/KI
;
;
;##############################
;
        MICRO   4
;
;##############################
;
;
<< DRIVE4                               ;RETURN STOPPED FLOW
        PUSH    PSW
        XRA     A
        STA     16000
        POP     PSW
        >>
<< FILL4                                ;FILL S.F. SYRINGES
        PUSH    PSW
        MVI     A,2
        STA     16000
        POP     PSW
        >>
<< FLOW4                                ;RUN A PUSH ON S.F.
        PUSH    PSW
        MVI     A,4
        STA     16000
        POP     PSW
        >>
<< WAIT4
        PUSH    PSW                                 ;LONG WAIT (APPROX. 3
        PUSH    B
        PUSH    D
        MVI     D,1
:10     MVI     B,377
:20     MVI     C,377
:30     DCR     C
        JNZ     .30
        DCR     B
        JNZ     .20
        DCR     D
        JNZ     .10
        POP     D
        POP     B
        POP     PSW
        >>
<< PUSHIT                       ;FILL THE SYRINGES AND DO A S.F. PUSH
        FILL4                   ;FILL THE SYRINGES
        WAIT4
        DRIVE4                  ;PRESSUREIZE THE SYRINGES
        WAIT4
        FLOW4                   ;DO THE PUSH
        >>
<< EMPTY                        ;EMPTY THE DRIVE SYRINGES
        MVI     A,4
:10     WAIT4
        FLOW4
        WAIT4
        DRIVE4
        DCR     A
        JNZ     .10
        >>
<< FLUSH4                               ;FLUSH OUT THE S.F.
        PUSH    PSW
        PUSH    B
        LDA     17                      ;TIMES TO FLUSH
        MOV     B,A
:10     WAIT4
        PUSHIT                          ;TAKE A PUSH
```

```
        WAIT4
        DRIVE4                          ;EXHAUST TO WASTE
        EMPTY                           ;EMPTY THE SYRINGES
        DCR     B
        JNZ     .10
        POP     B
        POP     PSW
        >>
<< CLEAR1                       ;CLEAR CLOCK # 1
        PUSH    PSW
        MVI     A,377           ;DOWN COUNTERS
        STA     12000
        STA     12000
        POP     PSW
        >>
<< CLEAR2                       ;CLEAR CLOCK # 2
        PUSH    PSW
        MVI     A,377           ;DOWN COUNTERS
        STA     12001
        STA     12001
        POP     PSW
        >>
<< VEL4                                 ;RECORD A TIME PROFILE
        XRA     A
        STA     20              ;# POINTS = 0
        MVI     A,60            ;INIT COUNTER # 1
        STA     12003
        MVI     A,160           ;INIT COUNTER # 2
        STA     12003
        CLEAR1                  ;CLEAR COUNTER # 1
        CLEAR2                  ;CLEAR COUNTER # 2
        LHLD    10              ;LASTAD
        MOV     A,H
        ANI     37              ;REMOVE MICRO ADDRESS
        MOV     H,A
        LXI     D,20            ;ADDRESS OF THE # OF COUNTS
        LDA     16000           ;WATCH POSITION FLAG
        RAR                     ;BIT 0 IS THE FLAG
        JNC     .20             ;WATCH COUNTER # 2
:10     LDA     16000           ;POSITION FLAG
        RAR
        JC      .10
        LDA     12000           ;GET THE LOW BYTE
        MOV     M,A             ;STORE IT
        INX     H
        LDA     12000           ;GET THE HIGH BYTE
        MOV     M,A             ;STORE IT
        INX     H
        XCHG
        INR     M               ;INC.  THE # OF POINTS
        XCHG
        CLEAR1                  ;CLEAR COUNTER # 1
:20     LDA     16000           ;POSITION FLAG
        RAR
        JNC     .20
        LDA     12001           ;GET THE LOW BYTE
        MOV     M,A             ;STORE IT
        INX     H
        LDA     12001           ;GET THE HIGH BYTE
        MOV     M,A             ;STORE IT
        INX     H
        XCHG
        INR     M               ;INC.  THE # OF POINTS
        XCHG
        CLEAR2                  ;CLEAR COUNTER # 2
        JMP     .10             ;WATCH COUNTER # 1
        >>
;
;
; ##############################
;
```

```
        MICRO   0
;
;##########################
;
;
<< STOP                                 ;STOP ALL SLAVE MICROS
        PUSH    PSW
        XRA     A
        STA     16000   ;OUTPUT
        STA     46      ;OUTPUT SAVE
        POP     PSW
        >>
<< RUN4                                 ;RUN MICRO #4
        RUN     4
:10     FLAG    4
        JZ      .10
        LDA     46
        ANI     77                      ;CLEAR RUN BIT
        STA     16000
        STA     46
        >>
<< DRIVE                                ;RETURN STOPPED-FLOW
        PUSH    PSW
        SETUP   4,DRIVE4
        RUN4
        POP     PSW
        >>
<< WASTE
        DRIVE                           ;RETURN TO DEFAULT
        >>
<< FILL                                 ;FILL S.F.  SYRINGES
        PUSH    PSW
        SETUP   4,FILL4
        RUN4
        POP     PSW
        >>
<< FLOW                                 ;RUN A PUSH ON S.F.
        PUSH    PSW
        SETUP   4,FLOW4
        RUN4
        POP     PSW
        >>
<< FLUSH                                ;FLUSH THE S.F.
        BININ                           ;# OF TIMES TO FLUSH
        POP2
        MOV     A,L
        STA     100017
        SETUP   4,FLUSH4
        RUN4
        >>
<< WAIT                                 ;A CALL HERE CAUSES 99.999
        PUSH    H
        PUSH    PSW
        41,367,50           ;LXI H,367,50
        SHLD    .100
:10     LXI     H,.100
        MVI     A,367
        STA     .100
:20     DCR     M
        JNZ     .20
        INX     H
        DCR     M
        JNZ     .10
        NOP
        NOP
        POP     PSW
        POP     H
        RET
:100    0
        >>
<< VEL                                  ;RECORD A TIME PROFILE
```

```
        CFIND                   ;INPUT DATASET TO STORE IT IN
        SETUP   4,PUSHIT
        RUN4                    ;START THE PUSH
        SETUP   4,VEL4
        RUN     4               ;START TAKING DATA
        MVI     A,12            ;WAIT 1 SEC TO COLLECT DATA
:10     WAIT
        DCR     A
        JNZ     .10
        STOP
        SETUP   4,EMPTY
        RUN4                    ;EMPTY THE DRIVE SYRINGES
        LDA     100020          ;# POINTS
        STA     40070           ;CPTS
        STA     40054           ;GPTS
        MOV     C,A
        MVI     B,0             ;DON'T CAUSE TROUBLE
        POP2                    ;GET DATASET ADDRESS (FROM CFIND)
        XCHG
        LHLD    100010          ;START OF DATA
:20     MOV     A,M             ;GET A BYTE
        CMA                     ;THEY'RE DOWN COUNTERS
        STAX    D               ;STORE THE DATA
        INX     H
        INX     D
        MOV     A,M             ;HIGH BYTE
        CMA
        STAX    D
        INX     H
        INX     D
        DCR     C               ;DONE?
        JNZ     .20
        MVI     A,1
        STA     40056           ;AVGA = 1
        STA     40034           ;ACGS = 1
        >>
;
;
;##############################
;
        MICRO   4
;
;##############################
;
;
<< VCNT
        LDA     20
        MOV     C,A
        LDA     16000
        RAR
        JNC     .20
:10     LDA     16000
        RAR
        JC      .10
        DCR     C
        RZ
:20     LDA     16000
        RAR
        JNC     .20
        DCR     C
        RZ
        JMP     .10
        >>
<< VSF4
        PUSHIT                          ;TAKE A PUSH
        VCNT                            ;WAIT FOR THE TRIGGER POINT
        >>
;
;
;##############################
;
```

```
        MICRO   2
;
;############################
;
;
<< VSF2
        LDA     50              ;# SCANS TO TAKE
        MOV     D,A
        LDA     64              ;# SCANS TO SKIP
        MOV     E,A
        LDA     34              ;# AVERAGES
        MOV     B,A
        MVI     C,0             ;# INT. = 0
        LHLD    10              ;DATASET POINTER
        MOV     A,H
        ANI     77              ;REMOVE MICRO BITS
        MOV     H,A
        STA     140010          ;CLEAR SCAN FLAG
:10     LDA     140011          ;READ SCAN FLAG
        RAL
        JNC     .10
        STA     140010          ;CLEAR SCAN FLAG
:20     MVI     A,21            ;SET F.A. SCAN BIT
        STA     120013
:30     MVI     A,1             ;TAKE 1 SCAN
        MSCAN
        DCR     D               ;LAST SCAN?
        JZ      .60
        CMP     E               ;SKIP = 0?
        JZ      .30
        PUSH    D
        MVI     A,1
        STA     120013          ;STOP F.A.
:40     LDA     140011          ;READ THE SCAN FLAG
        RAL
        JNC     .40
        STA     140010          ;CLEAR SCAN FLAG
        DCR     E               ;LAST SCAN TO SKIP?
        JNZ     .40
        POP     D
        JMP     .20
:60     ENDS
        >>
<< SFMAX2
        LHLD    76              ;DATASET POINTER
        LDA     34              ;NUMBER OF AVERAGES
        MOV     B,A
        MVI     C,0             ;# INT. = 0
        LDA     64
        MOV     E,A             ;# SCANS TO SKIP
        STA     140010          ;CLEAR SCAN FLAG
:10     LDA     140011          ;READ THE SCAN FLAG
        RAL
        JNC     .10
        STA     140010          ;CLEAR THE SCAN FLAG
        DCR     E               ;SKIP?
        JNZ     .10
        MVI     A,21            ;ENABLE F.A.
        STA     120013
        MVI     A,1             ;1 SCAN
        MSCAN
        ENDS
        >>
;
;
;############################
;
        MICRO   0
;
;############################
;
```

```
;
<< SFSCAN                          ;SFSCAN  # SKIP, TRIGGER, #
     STOP
     DECIN                         ;# TO SKIP
     POP2
     MOV       A,L
     STA       40064
     DECIN                         ;TRIGGER POINT
     POP2
     MOV       A,L
     STA       100020
     FLUSH                         ;FLUSH IT
     SETUP     2,VSF2
     SETUP     4,VSF4
     RUN4                          ;RUN THE STOPPED-FLOW
     RUN2                          ;TAKE THE SPECTRA
     STOP
     SETUP     4,EMPTY             ;EMPTY THE SYRINGES
     RUN4
     >>

<< SFMAX                           ;SFMAX  # SKIP, DATASET, TRIG
     STOP
     DECIN                         ;# SKIP
     POP2
     MOV       A,L
     STA       40064
:10  DECIN                         ;DATASET #
     LDA       40050               ;CURRENT # OF DATASETS
     POP2
     CMP       L
     JNC       .20
     WHAT
     JMP       .10
:20  MOV       A,L
     DCR       A
     RAL
     MOV       E,A
     MVI       D,0
     LHLD      40010               ;DATASET TABLE
     DAD       D
     MOV       A,H
     ANI       77                  ;REMOVE MICRO ADDRESS BITS
     MOV       H,A
     SHLD      40076
     DECIN                         ;TRIGGER POINT
     POP2
     MOV       A,L
     STA       100020
     FLUSH                         ;FLUSH IT
     SETUP     2,SFMAX2
     SETUP     4,VSF4
     RUN4                          ;RUN THE STOPPED-FLOW
     RUN2                          ;TAKE THE SCAN
     STOP
     SETUP     4,EMPTY             ;EMPTY THE SYRINGES
     RUN4
     >>
```

B. Minicomputer Programs

```
C*****************************
C
C   CONV2.FTN
C
C     THIS ROUTINE IS USED
C  TO CONVERT STANDARD
C  BINARY FILES FROM THE
C  VIDICON SYSTEM INTO ASCII
C  FILES.
C
C*****************************
      BYTE IFILE(24),OFILE(24)
      INTEGER PARAM(20),COUNT,DATA(1024)
10    WRITE(6,900)
900   FORMAT(' INPUT FILE:'/)
      READ(6,905)IFILE
905   FORMAT(24A1)
      IERR=0
      CALL FORPIP(IFILE,IERR)
      IF(IERR)20,30,40
20    WRITE(6,904)
904   FORMAT(' FILE ERROR')
      GOTO 10
30    WRITE(6,906)
906   FORMAT(' FILE NOT FOUND')
      GOTO 10
40    CALL UNFMO(IERR,IFILE,0,1)
      IF(IERR.EQ.0)GOTO 44
      WRITE(6,907)
907   FORMAT(' UNFMO ERROR')
      GOTO 10
44    WRITE(6,910)
910   FORMAT(' OUTPUT FILE:'/)
      READ(6,905)OFILE
      IERR=0
      CALL FORPIP(OFILE,IERR)
      IF(IERR)50,70,60
50    WRITE(6,904)
      GOTO 44
60    WRITE(6,915)
915   FORMAT(' FILE ALREADY EXISTS')
      GOTO 44
70    CALL ASSIGN(4,OFILE,24,IERR)
      IF(IERR.EQ.0)GOTO 80
      WRITE(6,920)
920   FORMAT(' ASSIGN ERROR')
      GOTO 44
80    COUNT=36
      CALL UNFMRW(IERR,PARAM,COUNT)
      AVG=(FLOAT(PARAM(2)))*(FLOAT(PARAM(11)))
      WRITE(6,925)
925   FORMAT(' XINIT,XINC:'/)
      READ(6,927)X,XINC
927   FORMAT(2F15.0)
      COUNT=2048
      CALL UNFMRW(IERR,DATA,COUNT)
      COUNT=COUNT/2
      DO 100 I=1,COUNT
      Y=FLOAT(DATA(I))
      IF(Y.GE.0)GOTO 90
      Y=65536.0 + Y
90    Y=Y/AVG
      WRITE(4,930)X,Y
930   FORMAT('RD',2G15.5)
      X=X+XINC
100   CONTINUE
```

```
      CALL UNFMC(IERR)
      ENDFILE 4
      WRITE(6,940)
940   FORMAT(' CONVERSION COMMPLETE')
      GOTO 10
      END
C*********************************
C
C   ERIK CARLSON    19-JULY-1978
C
C   THIS ROUTINE ALLOWS EVALUATION
C   OF DATA FROM THE STOPPED-FLOW
C   VIDICON RAPID SCANNING SYSTEM.
C   THE OUTPUT OF THIS ROUTINE
C   CAN BE FILES CONTAINING EITHER
C   ABSORBANCE VS TIME OR ABSORBANCE
C   VS WAVELENGTH.
C
C*********************************
      BYTE IFILE(24),OFILE(24)
      INTEGER RPTS,DPTS,NPTS,TSCNS
      INTEGER PARAM(20),COUNT,DATA(1024)
      DIMENSION REF(512),DARK(512),WORK(512)
C##########
C
C FILE INPUT SECTION FOR REFERENCE AND DARK SIGNALS
C
C##########
10    WRITE(6,900)
900   FORMAT(' REFERENCE')
      CALL INPUT(REF,RPTS)
      WRITE(6,902)
902   FORMAT(' DARK SIGNAL')
      CALL INPUT(DARK,DPTS)
      IF(RPTS.EQ.DPTS)GOTO 20
      WRITE(6,904)
904   FORMAT(' NUMBER OF POINTS NOT EQUAL!!')
      GOTO 10
20    DO 25 I=1,RPTS
25    REF(I) = REF(I) - DARK(I)
22    WRITE(6,906)
906   FORMAT(' INPUT FILE:'/)
      READ(6,908)IFILE
908   FORMAT(24A1)
      IERR=0
      CALL FORPIP(IFILE,IERR)
      IF(IERR)30,40,50
30    WRITE(6,910)
910   FORMAT(' FILE ERROR')
      GOTO 22
40    WRITE(6,912)
912   FORMAT(' FILE NOT FOUND')
      GOTO 22
50    CALL UNFMO(IERR,IFILE,0,1)
      IF(IERR.EQ.0)GOTO 60
      WRITE(6,914)
914   FORMAT(' UNFMO ERROR')
      GOTO 22
60    COUNT=36
      CALL UNFMRW(IERR,PARAM,COUNT)
      AVG=(FLOAT(PARAM(2)))*(FLOAT(PARAM(11)))
      NPTS=PARAM(1)-PARAM(17)
      TSCNS=PARAM(8)
      IF(NPTS.EQ.RPTS)GOTO 70
      WRITE(6,904)
      GOTO 80
C##########
C
C SELECTION OF TYPE OF DATA ANALYSIS
C
C##########
```

```
70      WRITE(6,916)
916     FORMAT(' 1=ABS VS WAV, 2=ABS VS TIME, 3=END'/)
        READ(6,918)IERR
918     FORMAT(I5)
        GOTO(100,200)IERR
80      CALL UNFMC(IERR)
        GOTO 10
C###########
C
C ABSORBANCE VS WAVELENGTH
C
C###########
100     WRITE(6,920)
920     FORMAT(' SCAN#, SKIP#, THRESHOLD: '/)
        READ(6,922)ISCAN,ISKIP,THR
922     FORMAT(2I6,F10.0)
        WRITE(6,924)
924     FORMAT(' INIT. WAV., WAV. INC.: '/)
        READ(6,926)XINIT,XINC
926     FORMAT(2F10.0)
        IF(TSCNS.GE.ISCAN)GOTO 110
        WRITE(6,928)TSCNS
928     FORMAT(' SCAN# GREATER THAN TOTAL OF: ',I7)
        GOTO 100
110     DO 120 I=1,ISCAN
        COUNT=NPTS*2
        CALL UNFMRW(IERR,DATA,COUNT)
120     CONTINUE
        DO 130 I=ISKIP,NPTS
        XY=FLOAT(DATA(I))
        IF(XY.GE.0.0)GOTO 130
        XY=65536.0 + XY
130     WORK(I)=(XY/AVG)-DARK(I)
C###########
C
C CALCULATE THE ABSORBANCES
C
C###########
500     DO 560 I=ISKIP,NPTS
        R=REF(I)
        S=WORK(I)
        IF((R-S)-THR)550,540,540
540     T=S/R
        IF(1.0-T)550,550,545
545     IF(T)550,550,547
547     WORK(I)=-1.0*ALOG10(T)
        GOTO 560
550     WORK(I)=0.0
560     CONTINUE
        GOTO 600
C###########
C
C ABSORBANCE VS TIME
C
C###########
200     WRITE(6,930)
930     FORMAT(' POINT#, THRESHOLD: '/)
        READ(6,932)IPT,THR
932     FORMAT(I6,F10.0)
        WRITE(6,934)
934     FORMAT(' INIT. TIME, TIME INC.: '/)
        READ(6,926)XINIT,XINC
        DO 220 I=1,TSCNS
        COUNT=NPTS*2
        CALL UNFMRW(IERR,DATA,COUNT)
        XY=FLOAT(DATA(IPT))
        IF(XY.GE.0.0)GOTO 220
        XY=65536.0 + XY
220     WORK(I)=(XY/AVG)-DARK(IPT)
C###########
C
```

```
C CALCULATE THE ABSORBANCES
C
C###########
      R=REF(IPT)
      DO 460 I=1,TSCNS
      S=WORK(I)
      IF((R-S)-THR)450,440,440
440   T=S/R
      IF(1.0-T)450,450,445
445   IF(T)450,450,447
447   WORK(I)=-1.0*ALOG10(T)
      GOTO 460
450   WORK(I)=0.0
460   CONTINUE
      ISKIP=1
      NPTS=TSCNS
C###########
C
C FILE OUTPUT SECTION
C
C###########
600   WRITE(6,936)
936   FORMAT(' OUTPUT FILE: '/)
      READ(6,908)OFILE
      IERR=0
      CALL FORPIP(OFILE,IERR)
      IF(IERR)650,670,660
650   WRITE(6,910)
      GOTO 600
660   WRITE(6,938)
938   FORMAT(' FILE ALREADY EXISTS')
      GOTO 600
670   CALL ASSIGN(4,OFILE,24,IERR)
      IF(IERR.EQ.0)GOTO 700
      WRITE(6,940)
940   FORMAT(' ASSIGN ERROR')
      GOTO 600
700   DO 720 I=ISKIP,NPTS
      WRITE(4,942)XINIT,WORK(I)
942   FORMAT('RD',2G15.5)
720   XINIT=XINIT+XINC
      ENDFILE 4
      CALL UNFMC(IERR)
      GOTO 50
      END
C*********************************
C
C ERIK CARLSON   27-JULY-1978
C
C LSLOPE.FTN
C
C THIS ROUTINE CALCULATES THE
C SLOPE OF A SET OF DATA IN
C A STANDARD DATA FILE AND
C OUTPUTS A NEW DATA FILE
C OF SLOPE VS X-DATA.
C THE SLOPE IS CALCULATED BY
C A LEAST-SQUARES FIT TO A
C PORTION OF THE DATA.
C    ROUTINES NEEDED:
C
C      FORPIP
C      LEASTS
C
C*********************************
      BYTE IFILE(24)
      INTEGER COUNT
      DIMENSION X(200),Y(200),Z(200)
10    WRITE(6,900)
900   FORMAT(' INPUT FILE: '/)
      READ(6,901)IFILE
```

```
901     FORMAT(24A1)
        IERR=0
        CALL FORPIP(IFILE,IERR)
        IF(IERR)20,30,40
20      WRITE(6,902)
902     FORMAT(' FILE ERROR')
        GOTO 10
30      WRITE(6,903)
903     FORMAT(' FILE NOT FOUND')
        GOTO 10
40      CALL ASSIGN(4,IFILE,24,IERR)
        COUNT=1
50      READ(4,904,END=60)X(COUNT),Y(COUNT)
904     FORMAT(2X,2G15.5)
        COUNT=COUNT+1
        GOTO 50
60      ENDFILE 4
        COUNT=COUNT-1
        WRITE(6,920)COUNT
920     FORMAT(' TOTAL PTS:',I5,/' START OF FIT, END OF FIT
        1, FIRST POINT, WIDTH OF FIT:'/)
        READ(6,922)IFS,IFE,IPT,IWID
922     FORMAT(4I5)
        JPT=IPT
        COUNT=0
200     IE=IFS+IWID
        CALL LEASTS(X(I),Y(I),5,SLOPE,YINIT,CORR)
        DO 220 I=IFS,IE
        CALL LEASTS(X(I),Y(I),1,SLOPE,YINIT,CORR)
220     CONTINUE
        CALL LEASTS(X(I),Y(I),4,Z(IPT),YINIT,CORR)
        COUNT=COUNT+1
        IPT=IPT+1
        IFS=IFS+1
        IF(IPT.LE.IFE.AND.IE.LT.IFE)GOTO 200
75      WRITE(6,907)
907     FORMAT(' OUTPUT FILE:'/)
        READ(6,901)IFILE
        IERR=0
        CALL FORPIP(IFILE,IERR)
        IF(IERR)80,100,90
80      WRITE(6,902)
        GOTO 75
90      WRITE(6,908)
908     FORMAT(' FILE ALREADY EXISTS')
        GOTO 75
100     CALL ASSIGN(4,IFILE,24,IERR)
        IE=JPT+COUNT-1
        DO 120 I=JPT,IE
        WRITE(4,909)X(I),Z(I)
909     FORMAT('RD',2G15.5)
120     CONTINUE
        ENDFILE 4
        GOTO 10
        END
C*****************************
C
C   TFREQ.FTN
C
C     THIS ROUTINE CONVERTS FILES
C  OF RELATIVE INTENSITY VS.
C  FREQUENCY TO FILES WHICH
C  HAVE BEEN CORRECTED FOR THE
C  VARYING EXPOSURE TIMES.
C  THIS IS DONE BY SIMPLY
C  MULTIPLYING THE ORIGINAL
C  INTENSITY BY THE FREQUENCY.
C
C*****************************
        BYTE IFILE(24)
        INTEGER COUNT
```

```
      DIMENSION X(200),Y(200)
10    WRITE(6,900)
900   FORMAT(' INPUT FILE: '/)
      READ(6,901)IFILE
901   FORMAT(24A1)
      IERR=0
      CALL FORPIP(IFILE,IERR)
      IF(IERR)20,30,40
20    WRITE(6,902)
902   FORMAT(' FILE ERROR')
      GOTO 10
30    WRITE(6,903)
903   FORMAT(' FILE NOT FOUND')
      GOTO 10
40    CALL ASSIGN(4,IFILE,24,IERR)
      COUNT=1
50    READ(4,904,END=60)X(COUNT),Y(COUNT)
904   FORMAT(2X,2G15.5)
      COUNT=COUNT+1
      GOTO 50
60    ENDFILE 4
      COUNT=COUNT-1
75    WRITE(6,907)
907   FORMAT(' OUTPUT FILE: '/)
      READ(6,901)IFILE
      IERR=0
      CALL FORPIP(IFILE,IERR)
      IF(IERR)80,100,90
80    WRITE(6,902)
      GOTO 75
90    WRITE(6,908)
908   FORMAT(' FILE ALREADY EXISTS')
      GOTO 75
100   CALL ASSIGN(4,IFILE,24,IERR)
      DO 120 I=1,COUNT
      Y(I)=Y(I)*X(I)
      WRITE(4,909)X(I),Y(I)
909   FORMAT('RD',2G15.5)
120   CONTINUE
      ENDFILE 4
      GOTO 10
      END
C*****************************
C
C    NOISE2.FTN
C
C    THIS ROUTINE IS USED TO
C STUDY NOISE IN THE VIDICON
C SYSTEM.   IT TAKES A STANDARD
C VIDICON BINARY DATA ARRAY
C FILE AND CALCULATES THE
C RELATIVE STANDARD DEVIATION
C OF THE DATA AT EACH CHANNEL.
C
C*****************************
      BYTE IFILE(24)
      INTEGER PARAM(20),COUNT,DATA(1024)
      DIMENSION XDATA(3,1024)
10    DO 5 I=1,3
      DO 5 J=1,1024
5     XDATA(I,J)=0.0
      WRITE(6,900)
900   FORMAT(' INPUT FILE: '/)
      READ(6,905)IFILE,IERR
905   FORMAT(24A1,Q)
      IERR=0
      CALL FORPIP(IFILE,IERR)
      IF(IERR)10,10,20
20    CALL UNFMO(IERR,IFILE,0,1)
      IF(IERR.NE.0)GOTO 10
      COUNT=36
```

```
      CALL UNFMRW(IERR,PARAM,COUNT)
      AVG=(FLOAT(PARAM(2)))*(FLOAT(PARAM(11)))
      NPTS=PARAM(1)-PARAM(17)
      NSCNS=PARAM(8)
      YAVG=FLOAT(NSCNS)
      DO 30 J=1,NSCNS
      COUNT=NPTS*2
      CALL UNFMRW(IERR,DATA,COUNT)
      DO 30 I=1,NPTS
      X=FLOAT(DATA(I))
30    XDATA(1,I)=XDATA(1,I)+(X/AVG)
      DO 50 I=1,NPTS
50    XDATA(1,I)=(XDATA(1,I))/YAVG
      CALL UNFMC(IERR)
      CALL UNFMO(IERR,IFILE,0,1)
      COUNT=36
      CALL UNFMRW(IERR,PARAM,COUNT)
      DO 60 J=1,NSCNS
      COUNT=NPTS*2
      CALL UNFMRW(IERR,DATA,COUNT)
      DO 60 I=1,NPTS
      X=(FLOAT(DATA(I)))/AVG
60    XDATA(2,I)=XDATA(2,I)+(X-XDATA(1,I))*(X-XDATA(1,I))
      DO 80 I=1,NPTS
      XDATA(2,I)=SQRT((XDATA(2,I))/(YAVG-1.0))
80    XDATA(3,I)=XDATA(2,I)/XDATA(1,I)
      DO 90 I=1,3
85    WRITE(6,910)I
910   FORMAT(' OUTPUT FILE',I3,': ',/)
      READ(6,905)IFILE,IERR
      IF(IERR.EQ.0)GOTO 90
      IERR=0
      CALL FORPIP(IFILE,IERR)
      IF(IERR)85,88,85
88    CALL ASSIGN(4,IFILE,24,IERR)
      IF(IERR.NE.0)GOTO 85
      DO 89 J=1,NPTS
      X=FLOAT(J)
      Y=XDATA(I,J)
      WRITE(4,920)X,Y
920   FORMAT('RD',2G15.5)
89    CONTINUE
      ENDFILE 4
90    CONTINUE
      GOTO 10
      END
C*****************************
C
C    VELC.FTN
C
C     THIS ROUTINE IS USED
C  TO PERFORM VARIOUS TYPE OF
C  DATA MANIPULATIONS ON THE
C  DATA FROM THE VIDICON STOPPED-
C  FLOW POSITION/VELOCITY TRANSDUCER.
C  THE DATA CAN BE OUTPUT AS
C  VELOCITY VS DISTANCE,VELOCITY VS.
C  TIME OF DISTANCE VS TIME.
C
C*****************************
      BYTE IFILE(24)
      INTEGER COUNT
      DIMENSION X(200),Y(200),XO(200),YO(200)
      WRITE(6,906)
906   FORMAT(' LINES PER INCH:'/)
      READ(6,905)XLN
905   FORMAT(F10.0)
      DX=2.54/XLN
10    WRITE(6,900)
900   FORMAT(' INPUT FILE:'/)
      READ(6,901)IFILE
```

```
901     FORMAT(24A1)
        IERR=0
        CALL FORPIP(IFILE, IERR)
        IF(IERR)20, 30, 40
20      WRITE(6, 902)
902     FORMAT(' FILE ERROR')
        GOTO 10
30      WRITE(6, 903)
903     FORMAT(' FILE NOT FOUND')
        GOTO 10
40      CALL ASSIGN(4, IFILE, 24, IERR)
        COUNT=1
50      READ(4, 904, END=60)X(COUNT), Y(COUNT)
904     FORMAT(2X, 2G15.5)
        COUNT=COUNT+1
        GOTO 50
60      ENDFILE 4
        COUNT=COUNT-1
65      WRITE(6, 910)
910     FORMAT(' 1=V/D, 2=V/T, 3=D/T, 4=END'/)
        READ(6, 911)ITYPE
911     FORMAT(I4)
        GOTO(200, 300, 400)ITYPE
        GOTO 10
200     DO 210 I=1, COUNT
        XO(I)=X(I)*DX
210     YO(I)=(DX*1000000.0)/Y(I)
        GOTO 75
300     TSEC=0.0
        DO 310 I=1, COUNT
        TSEC=(Y(I)/1000.0) + TSEC
        XO(I)=TSEC
310     YO(I)=(DX*1000000.0)/Y(I)
        GOTO 75
400     TSEC=0.0
        DO 410 I=1, COUNT
        TSEC=(Y(I)/1000.0) + TSEC
        XO(I)=TSEC
410     YO(I)=X(I)*DX
75      WRITE(6, 907)
907     FORMAT(' OUTPUT FILE: '/)
        READ(6, 901)IFILE
        IERR=0
        CALL FORPIP(IFILE, IERR)
        IF(IERR)80, 100, 90
80      WRITE(6, 902)
        GOTO 75
90      WRITE(6, 908)
908     FORMAT(' FILE ALREADY EXISTS')
        GOTO 75
100     CALL ASSIGN(4, IFILE, 24, IERR)
        DO 120 I=1, COUNT
        WRITE(4, 909)XO(I), YO(I)
909     FORMAT('RD', 2G15.5)
120     CONTINUE
        ENDFILE 4
        GOTO 65
        END
```

REFERENCES

REFERENCES

1. R.E. Dessy, Anal. Chem., 49, 1100A (1977).

2. G.A. Korn, Computer Design, 177 (June, 1977).

3. W.J. Lennon, in Computer Networking and Chemistry, P. Lykos, Ed., Amer. Chem. Soc., Washington, D.C., 1975, pp. 67-84.

4. P. Day, in Computer Networking and Chemistry, P. Lykos, Ed., Amer. Chem. Soc., Washington, D.C., 1975, pp. 85-107.

5. R.L. Ashenhurst, in Computer Networking and Chemistry, P. Lykos, Ed., Amer. Chem. Soc., Washington, D.C., 1975, pp. 108-117.

6. J.C. Emery, in Computer Networking and Chemistry, P. Lykos, Ed., Amer. Chem. Soc., Washington, D.C., 1975, pp. 219-230.

7. B.C. Searls and D.E. Freberg, Computer, 8, 22 (1975).

8. D.H. Chung, Electronic Design, 12, 132 (1976).

9. D.R. McGlynn, Microprocessors: Technology, Architecture, and Applications, John Wiley and Sons, Inc., 1976, pp. 182-187.

10. D.H. Chung, Electronics, 102 (January, 1977).

11. P.M. Russo, Computer, 10, 67 (1977).

12. A.J. Weissberger, Computer Design, 151 (June, 1977).

13. D. Phillips and A. Goodman, Electronics, 109 (July, 1977).

14. C.J. Sipple, Digital Design, 68 (August, 1977).

15. D.J. Mueller, Computer Design, 81 (October, 1977).

16. G. Adams and T. Rolander, Computer Design, 81 (March, 1978).

17. S. Hourdakis, Ph.D. Thesis, Michigan State University, 1978.

18. Intel Corporation, Intel 8080 Microcomputer Systems User's Manual, Intel Corporation, Santa Clara, CA., 1976.

19. M.J. Milano and H.L. Pardue, Anal. Chem., 47, 25 (1975).

20. R.B. Coolen, Ph.D. Thesis, Michigan State University, 1974.

21. M.J. Milano and H.L. Pardue, Clin. Chem., 21, 211 (1975).

22. G.M. Rider and D.W. Margerum, Anal. Chem., 49, 2098 (1977).

23. T.A. Nieman, F.J. Holler, and C.G. Enke, Anal. Chem., 48, 899 (1976).

24. G.C. Pimentel, Appl. Opt., 7, 2155 (1968).

25. R.E. Santini, M.J. Milano, and J.L. Pardue, Anal. Chem., 45, 915A (1973).

26. Y. Talmi, Anal. Chem., 47, 658A (1975).

27. Y. Talmi, Anal. Chem., 47, 697 (1975).

28. J.L. Dye and L.H. Feldman, Rev. Sci. Instrum., 37, 154 (1966).

29. N. Papadakis, R.B. Coolen, and J.L. Dye, Anal. Chem., 47, 1644 (1975).

30. R.B. Coolen, N. Papadakis, J. Avery, C.G. Enke, and J.L. Dye, Anal. Chem., 47, 1649 (1975).

31. N. Papadakis, Ph.D. Thesis, Michigan State University, 1974.

32. R.M. Wightman, R.L. Scott, C.N. Reilley, and R.W. Murray, Anal. Chem., 46, 1492 (1974).

33. J.W. Strojek, G.A. Gruver, and T. Kuwana, Anal. Chem., 41, 481 (1969).

34. M.R. Hollaway and H.A. White, Biochem. J., 149, 221 (1975).

35. G.J. Faini, R.J. Desa, and J. Lee, in Flavins and Flavoproteins, T.P. Singer, Ed., Elsevier Scien. Pub. Co., Amsterday, Neth., 1976, pp. 82-86.

36. M.J. Milano, H.L. Pardue, T.E. Cook, R.E. Santini, D.W. Margerum, and J.M.T. Raycheba, Anal. Chem., 46, 374 (1974).

37. T.A. Nieman and C.G. Enke, Anal. Chem., 48, 619 (1976).

38. T.A. Nieman, Ph.D. Thesis, Michigan State University, 1975.

39. R.E. Dessy, in Computers in Chemical Research and Education, D. Hadi, Ed., Elsevier Scien. Pub. Co., Amsterdam, Neth., 1973, pp. 121-134.

40. D.A. Yates and T. Kuwana, Anal. Chem., 48, 510 (1976).

41. G. Horlick and E.G. Codding, Anal. Chem., 45, 1490 (1973).

42. G. Horlick, Appl. Spect., 30, 113 (1976).

43. J.E. Hornshuh, Ph.D. Thesis, Michigan State University, 1978.

44. D.G. Mitchell, K.W. Jackson, and K.M. Aldous, Anal. Chem., 45, 1215A (1973).

45. R.E. Johnson, Applications of RCA Silicon Diode Array Target Vidicons, Camera Tube Application Note AN-4623, RCA Electronics Components, Harrison, N.J., 1971.

46. B. Chance, J. Franklin Inst., 229, 455 (1940).

47. B. Chance, J. Franklin Inst., 229, 613 (1940).

48. B. Chance, J. Franklin Inst., 229, 737 (1940).

49. S.R. Crouch, F.J. Holler, P.K. Notz, and P.M. Beckwith, Appl. Spect. Rev., 13, 165 (1977).

50. D. Sanderson, J.C. Bittikofer, and H.L. Pardue, Anal. Chem., 42, 1340 (1970).

51. G.E. Mieling, R.W. Taylor, L.G. Hargis, J. English, and H.L. Pardue, Anal. Chem., 48, 1686 (1976).

52. K.R. O'Keefe and H.V. Malmstadt, Anal. Chem., 707 (1975).

53. B.W. Renoe, K.R. O'Keefe, and H.V. Malmstadt, Anal. Chem., 48, 661 (1976).

54. P.K. Notz, Ph.D. Thesis, Michigan State University, 1977.

55. F.J. Holler, S.R. Crouch, and C.G. Enke, Anal. Chem., 48, 1429 (1976).

56. E.C. Douglas, IEEE Trans. Electron. Devices, ED-22, 224 (1975).

57. Data Sheet on Delfection Assembly Type CY100, Penn-Tran Corp., Bellefonte, Penn.

58. L.A. Selke, IEEE Trans. Electron Devices, ED-16, 625 (1969).

59. H.L. Felkel, Jr., and H.L. Pardue, Anal. Chem., 49, 1112 (1977).

60. T.B. McCord and M.J. Frankston, Appl. Optics, 14, 1437 (1975).

61. J.E. Stewart, Durrum Application Notes No. 4, Durrum Instr. Corp., Palo Alto, CA.

62. J.F. Below, Jr., R.E. Connick, and C.P. Coppel, J. Amer. Chem. Soc., 80, 2961 (1958).

63. J.L. Dye and V.A. Nicely, J. Chem. Ed., 48, 443 (1971).

64. N.W. Tietz, Fundamentals of Clinical Chemistry, W.B. Saunders Co., Philadelphia, PA., 1976, pp. 298-304.

65. G.R. Kingsley, J. Biol. Chem., 131, 197 (1939).

66. A.G. Gornall, C.J. Bardawill, and M.M. David, J Biol. Chem., 177, 751 (1949).

67. J.W. Mehl, E. Pacovska, and R.J. Winzler, J Biol. Chem., 177, 13 (1949).

68. T.R. Koch, G.F. Johnson, and M.E. Chilcote, Clin. Chem., 20, 392 (1974).

69. J. Savory, M.G. Heintges, M. Sonowane, and R.E. Cross, Clin. Chem., 22, 1102 (1976).

70. Wai-Tak Law, Ph.D. Thesis, Michigan State University, 1978.

71. M.B. Denton and S.B. Tilden, paper no. 225 presented at the Pittsburgh Conference on Analytical Chemistry and Applied Spectros Copy, 1978.

72. C.H. Moore, Astron. Astrophys. Suppl., 15, 497 (1974).

73. Product Specifications on Model 1070 Intelligent Diskette Controller, Persci, Marina Del Ray, CA.

MICHIGAN STATE UNIVERSITY LIBRARIES
3 1293 03082 6436