MINICOMPUTER CONTROL OF A LATEX REACTOR

MINICOMPUTER CONTROL OF A LATEX REACTOR

By

STUART W. RAY, B.Eng.

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfilment of the Requirements

for the Degree

Master of Engineering

McMaster University

April 1976

MASTER OF ENGINEERING (1976) McMaster University
(Chemical Engineering) Hamilton, Ontario

TITLE: Minicomputer Control of a Latex Reactor

AUTHOR: Stuart W. Ray, B.Eng. (McMaster University)

SUPERVISORS: Professors A. E. Hamielec and J. D. Wright

NUMBER OF PAGES: viii, 106

ABSTRACT

A minicomputer software and hardware system was developed to monitor and control the particle size growth in the emulsion polymerization of vinyl acetate. The basis of the on line detector was a UV spectrophotometer which measured turbidity as a function of wavelength.

A preliminary investigation was made to develop a technique for converting turbidity measurements into particle size distributions.

TABLE OF CONTENTS

Table of Contents (continued)

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Emulsion polymerization is an important process by which several types of polymers are produced commercially as latices. An important property of the latex is the particle size distribution which influences the stability and film forming characteristics of the latex. A significant percentage of latices produced commercially do not meet particle size specifications and must therefore be recycled or discarded as waste. This provides motivation for the development of a control strategy to reduce the number of off grade batches, and thus the incentive for this study. The purpose of this work was to develop a computer monitoring and control system for particle growth. This should enable some future investigator to study emulsion polymerization on line and in great depth, and develop appropriate control strategy.

The computer system was developed specifically for polyvinyl acetate latices. However, with minor modifications the system should be applicable to any emulsion polymerization. Polyvinyl acetate was chosen here as a great deal of technical information was available through the studies by Keung (1) and Goosney (2). In particular, Keung(1) showed, using tedious off line measurements, that the particle size development could be followed through measurements of

turbidity with a UV spectrophotometer. He showed that the weight average and number average particle diameters could be calculated from a measurement of absorbance at two wavelengths. This measurement makes an estimation of the mean and variance of the distribution, important parameters. The detector response at different wavelengths must be sensitive to these parameters; although, for control purposes, it is not necessary to specify particle size distribution, but the turbidity profile at fixed wavelengths could also be used.

In this study the minicomputer was able to considerably reduce the amount of work and time involved in measuring absorbance as a function of wavelength. The system could prepare a sample automatically every two minutes. This frequency was much greater than the manual monitoring which had a frequency of one sample every twenty minutes. The absorbance (equivalent to light scattering at 180°) provided a convenient and cheap method of measurement, which could be related to the particle size distribution.

2. THEORY OF EMULSION POLYMERIZATION

The classical theory of emulsion polymerization has been detailed elsewhere (1,3). This discussion is limited to the emulsion polymerization of vinyl acetate.

The reaction is carried out as a batch system. At the start of the reaction, the mixture consists of approximately 60 wt % water, 40 wt % vinyl acetate monomer, 1 wt % sodium lauryl sulphate and ½ wt % potassium persulphate. The mixture is stirred vigorously to produce an emulsion which can be considered to consist of three phases. The first phase, water - the dispersion medium, contains potassium persulphate, the initiator, as a solute. A second phase is large monomer droplets, having a size of 1 to 10 microns, held stable by small amounts of emulsifier. The third phase consists of micelles. A micelle is a group of emulsifier molecules which have come together to form a small spherical aggregate of approximately 50 Å diameter. One end of the emulsifier molecule is hydrophilic and the other hydrophobic. The hydrophobic ends cluster together in the centre of the micelle. Monomer tends to diffuse from the water phase into the centre of the micelle, to form a monomer swollen micelle.

The number of micelles in the mixture is very large compared with the number of monomer droplets, providing an extremely large surface area for the capture of initiator

radicals. Thus most of the free radicals, formed in the aqueous phase via initiator decomposition, tend to enter micelles rather than monomer droplets. As soon as a radical enters a micelle the polymerization takes place rapidly within the micelle, because of the high monomer concentration. As the polymer particle grows, more monomer is absorbed from the monomer droplets which act as a reservoir. Also, as the particle grows, more emulsifier molecules adsorb causing unstung micelles to dissolve into the water phase. As the polymerization continues, we find at about 10 to 15% conversion that all the micelles have either become a polymer particle or dissolved into the water. This is considered as the end of the first stage of polymerization. At this point the total number of particles is fixed.

In the second stage polymer particles continue to grow and the monomer continues to diffuse out of the monomer droplets into the dispersion medium and then into the growing polymer particles. The end of the second stage is defined as when there is no monomer remaining as droplets. During this stage, monomer concentration in the polymer particles remains constant.

In the final stage of polymerization, there is no monomer left to diffuse into the particles, and the concentration of monomer drops in the polymer particle. In the final stages of polymerization it is possible for the polymer particles to coalesce. This will occur if the particles are small and there is insufficient emulsifier to stabilise them.

In summary, during stage one particles are nucleated and the total number of particles fixed. In stage two, particles grow but the total number is constant. Finally, in stage three there is a small shrinkage in particle size due to the density difference between monomer and polymer.

3. JUSTIFICATION FOR DIGITAL CONTROL

Digital control is essential due to the time limitation when control must be initiated and the complexity of calculations required. The first stage of polymerization occurs within the first 15 to 30 minutes of reaction. Measurements of particle size distribution and rate of growth must be made during this period and converted to the required control action, else the total number of particles is fixed. To control the variance of the distribution we must control in stage one.

Naturally, it is important that measurements be made as early as possible and the measurement is accurate. Most measurements of particle size distribution such as angular light scattering and electron microscope require extensive preparation of samples and the results may not be obtained until hours after the reaction is completed, a little late for process control. Measurement of the absorbance (light scattering at 180°) of the solution can be obtained very quickly using a UV spectrophotometer. However, in order to make such a measurement, a sample must be prepared quickly, accurately and consistently. This preparation involves diluting the latex solution with an accurate volume of diluent, and filling the light transmission cell with the prepared sample. This can not be done manually with

any degree of efficiency. Using the minicomputer, such a sample could be prepared every two minutes, or even more frequently.

4. DESCRIPTION OF THE HARDWARE

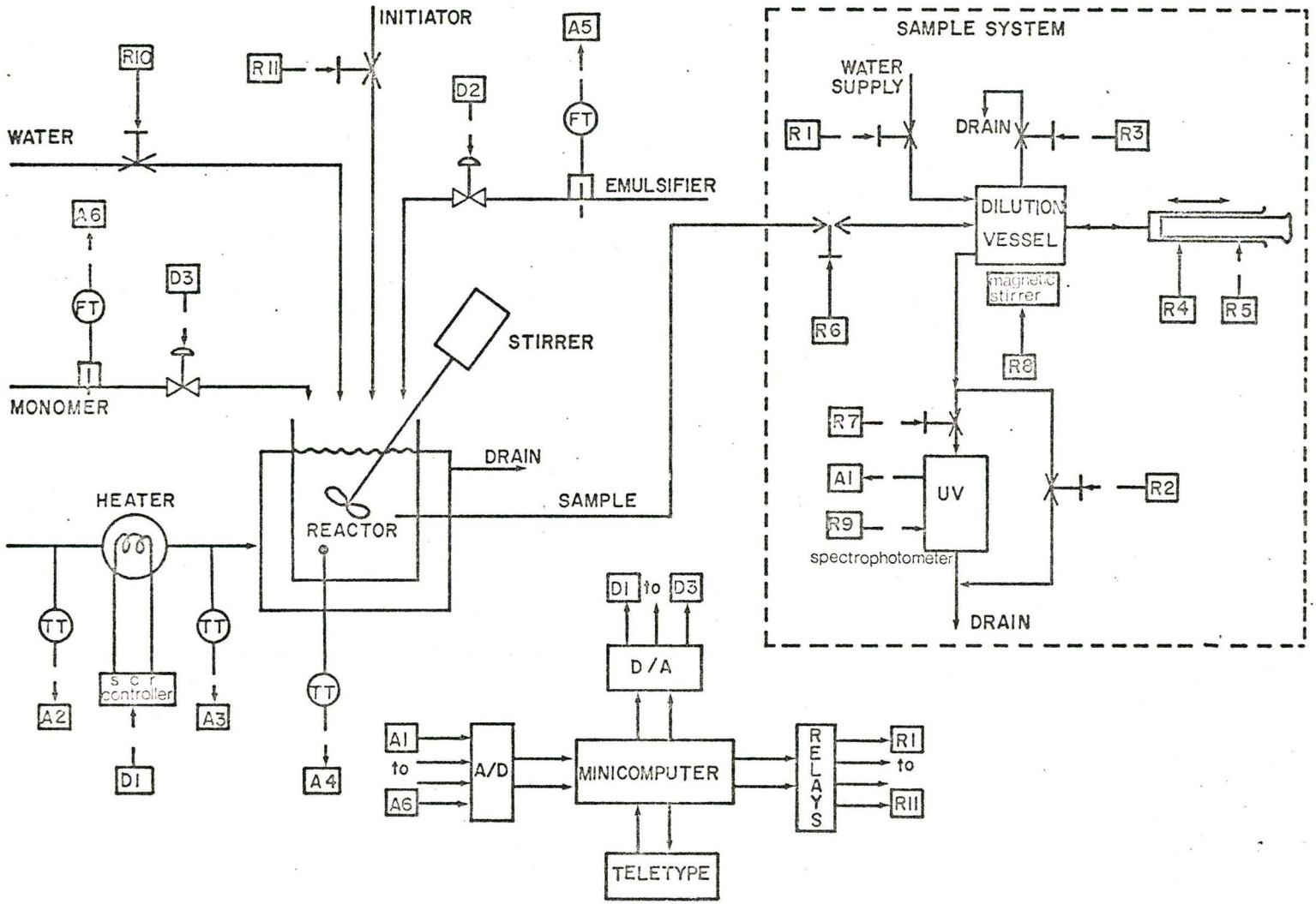
4.1 Process Equipment

Figure 1 is a schematic representation of the process equipment. The apparatus may be considered in four parts; the feed system, the temperature control system, the reaction system and the sampling system. They will be described in the above order.

The feed system consists of three vessels that hold the water-emulsifier mixture, the monomer, and the initiator solution. Each of these vessels is purged with nitrogen to eliminate oxygen. These mixtures are gravity fed to the reactor via three 1/8" stainless steel ASCO solenoid valves (120V, N.C.) which can be operated manually or by the computer. It is essential that the process streams do not contact brass or copper. The copper ions deactivate the catalyst.

Temperature is controlled in the reactor using a hot and cold water flow. The hot water is taken from the building supply and further heated using an electric heater to 70°C. The cold water, also from building supply runs at about 10°C. Water flow used in the experiments was about 2 litres/minute. The two water streams are controlled by means of 1/4" brass ASCO solenoid valves (120V, N.C.) which are interfaced to the computer (see section 4.2).

Fig.1 COMPUTER CONTROL OF A LATEX REACTOR



The reaction takes place in a 3 litre glass flask, surrounded by a water jacket through which the water heating/cooling passes. Inlets in the top of the vessel allow access by the feed streams, stirrer with speed control, thermometer, thermocouple, sample tube, nitrogen purge and a reflux condenser.

The sampling system is composed of a 250 ml earlenmeyer flask in which the concentrated latex from the reactor is diluted with distilled water. The vessel contents are mixed using a magnetic stirrer and then passed through a flow through light transmission cell where the absorbance is measured at different wavelengths using a UV spectrophotometer. All parts of the sampling system are interfaced to the mini-computer. Detailed operation of the sampling system can be found in section 5.4.

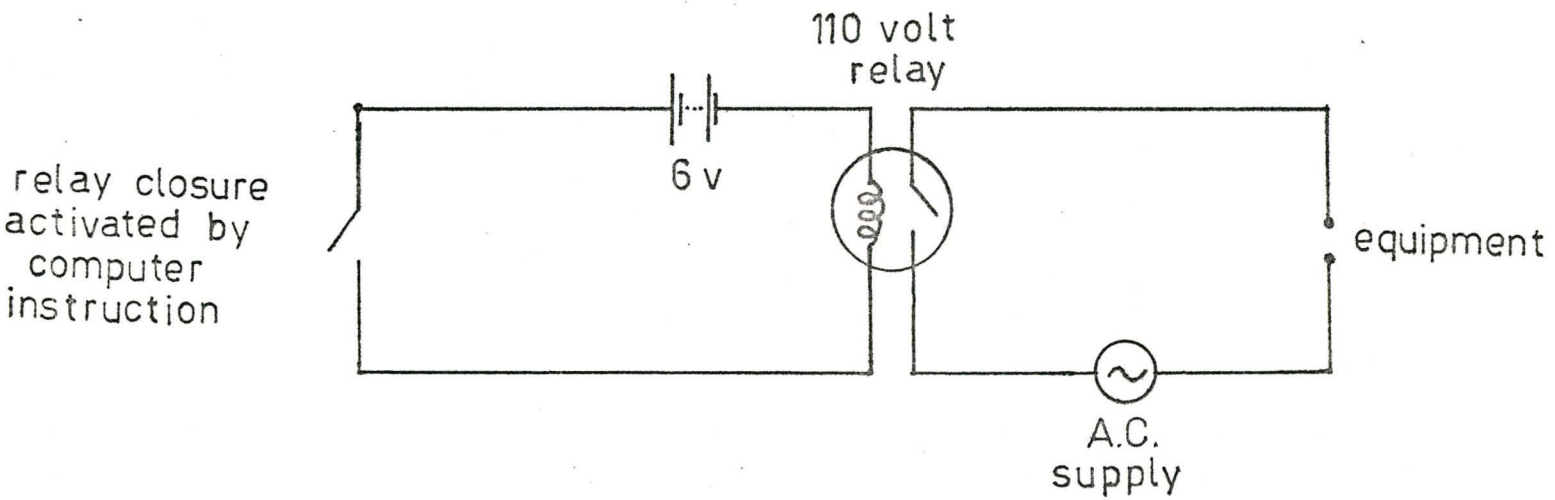
4.2 Control Computer and Process Interface

A dedicated minicomputer of the NOVA family (Data General manufacture) with 4K core was used to monitor and control the reaction. Each instruction and data storage location has a 16 bit word length. The machine has four accumulators, two of which can be used as index registers.

Peripherals included with this computer are a real time clock, power failure detector, teletypewriter, 16 A/D input channels and 16 relay outputs. The teletypewriter is used for loading the computer program via paper tape, operator communication with the system, and printing a log of the process conditions as the reaction progresses. Temperatures

within the reactor and jacket are measured using chromo-alumel thermocouples, then amplified with a temperature transmitter to provide a 0 to 10 volt input signal to the computer. The 16 relay outputs can only be used at low voltage (6V). When closed, they activate a second relay which closes the 110 volt supply circuit. When this secondary power supply is activated and the particular circuit is switched to computer control, the solenoid valve opens or the appropriate equipment is activated. Figure 2 describes this relay action. The real time clock allows the computer to keep track of physical time. A pulse is issued by the clock every one hundredth of a second, causing an interrupt to the CPU. The system then marks the passage of this time interval and can thus interface practically with the process.

Fig 2 RELAY INTERFACE TO PROCESS



5. DESCRIPTION OF THE SOFTWARE

In this section the method by which a digital computer can operate in a real time environment is described. After a basic overview of the computer handling in section 5.1, the actual system package developed for the latex reactor is described in section 5.2. The details of each routine are given in sections 5.3, 5.4 and 5.5. In section 5.6 a procedure for startup of the system is given.

5.1 Real Time Data Processing

The following will describe how the processing takes place in the particular system used in this study. Further detailed discussion may be found elsewhere (4).

In real time data processing it is always necessary for the computer to do two or more actions simultaneously. For example, it may be necessary to supply the operator an important message at the same time that a measurement of the temperature in a reactor is required. However, in a computer there is just one place where logical action can occur. This is the CPU (central processing unit). The CPU always works in the same manner. An instruction consisting of 16 bits is picked out of memory at an address indicated in the CPU. Depending on the configuration of these bits, the CPU performs a certain action. This action might consist of

obtaining a word from core and storing it in one of its four working registers (called accumulators) or perhaps to instruct the CPU that its next instruction is to be obtained from a certain storage location. The instruction set used for this type of programming is called assembly, where the instruction implicitly tells the CPU what to do. This is apart from the higher level languages such as fortran. However, all computers operate in a similar manner to that described above (with slight differences such as length of instruction word and number of working registers). These high level language programs must be converted to machine language. A program that makes this translation is called a compiler. The advantage in using assembly language is that very efficient use of core space is possible whereas a compiler, which has to be general in nature, must sacrifice this efficiency. On the other hand, high level programs allow for ease of programming.

A typical instruction for the CPU takes less than one microsecond to execute. To print a character, a teletype takes approximately one tenth of a second or 100,000 microsecond. If the CPU could not do any further processing until the teletype had finished printing, the CPU would be idle for 99.999% of this time period. In order to circumvent this problem, interface devices such as A/D converter or teletype can cause an interrupt of the currently executing instructions of the CPU. Now the instruction to print a letter on the teletype can be issued by the CPU and can then

carry on with other tasks, returning to service the teletype 100,000 instructions later.

In the configuration used in this system there are 6 devices with which the CPU can communicate. These are the A/D converter, D/A converter, relays, real time clock and teletype in and out. The actual instructions which service these devices are contained in RTOS (real time operating system), a program package supplied by Data General (5). The user programs communicate with RTOS by a series of pseudo-instructions, described in table 1. Using RTOS, it is no longer necessary for the user to worry about handling device interrupts or scheduling the running of tasks. This is all handled by RTOS in response to these pseudo-instructions.

TABLE 1: Communication with RTOS

Pseudo instruction	Meaning
.IOX	Input/Output to requested device
.RCV	Suspend this task until .XMIT allows this task to proceed
.XMIT	Allow suspended task to proceed
.XMIT@	Allow suspended task to proceed but do proceed with this task until message received.
.WAIT	Suspend this task for a given time interval
.FORK	Create a new task
.QUIT	This task has been completed, drop from system
.PTY	Set this task to priority specified (higher priority tasks are given execution preference by RTOS)

5.2 Configuration of the System

This computer system was created in order to provide ease of communication between the operator and the computer. The system is set up to receive a series of keywords via the teletype so that communication is almost conversational.

To begin operator interaction with the computer system, a CTRL A character is issued at the keyboard. The system is always waiting for this instruction. Upon issuance, the system terminates all output to the teletype, generates a line feed and carriage return and then awaits further instructions. These instructions consist of a word followed by a space. At present the allowable words are LOG, CHANGE, FROM, TO, READ, STOP, SCHEDULE, READOCT and START. The computer then carries out the appropriate software code related to the specific instruction. The code is called a "communication" routine, the details of which can be found in section 5.3.

Normally, upon reading the first word above, a second word is expected indicating either a data location within the system or a task which is to be executed. Possible data locations that the operator needs access are described in the particular task routines described in section 5.4. The tasks currently in the system are SAMPLE, INITIATOR, MONOMER, EMULSIFIER and CONTROLT, each described in detail in section 5.4. CONTROLT bears the meaning control temperature, the other tasks are self explanatory.

The software is rigged to allow easy expansion of the system to accept new tasks, data locations, and "dynamic" or communication words. Only two additional storage locations are required (see service routine INPUT) for the system to recognize a new word. Also, it is possible for different words to represent the same task or data location. For example it may be desirable to have both SAMPLE and SAMPLING refer to the same task so that the operator can issue the command START SAMPLING.

Another type of routine within the system is called the service routine. These routines are not directly accessed by the operator. Each service routine is used by more than one task to perform a particular function. They are necessary to conserve computer memory.

In summary, there are three types of routines within the system. The first type is the communication routine which allows the operator to control system execution. The second performs the tasks related to the physical system. The third type, the service routines are used by the tasks to perform a given function and conserve core space.

5.3 Communication Routines:

All operator interactions with the system are controlled by the system executive, INTER. This program initialises the system, and then continually waits for an indication that the operator wishes to communicate. The operator accomplishes this by issuing a CTRL A character from the teletype. Upon receiving this character, a

carriage return and line feed is issued to the teletype and CPU control is transferred to program INPUT. INTER then remains inactive until a CTRL A is again used. Program INPUT reads a name from the teletype. This name must be one of the communication routines described below. Control is then transferred to the appropriate routine.

CHANGE: This word is required whenever the operator wishes to alter the contents of a data location. The routine requests INPUT to read the name of a data location. If the word is acceptable, control is then transferred to INPUT to obtain the next communications word, usually FROM or TO.

FROM: This word, usually used after a CHANGE command, reads the current value of the specified data location. RTBDØ (a service routine described in section 5.5) is then called to print the value on the teletype in base 10 notation.

TO: The final word required in altering a data location. RTDBØ is used to read the new value. The routine then stores that value in the data location specified after the CHANGE word. A carriage return and line feed is then issued to the teletype. The system is then ready to accept further input.

READ: This word is typed when the contents of a data location are desired. Upon reading this word, INPUT is called to receive the name of the data location. The current value is then printed on the teletype in base 10 using the service routine RTBDØ. After printing, control is not passed to INTER but remains with READ. Further data locations can be read by merely typing their name. To access other communi-

cation routines, a CTRL A must be issued.

READOCT: This routine is similar to READ routine with the exception that the contents of the data location are printed in octal (base 8).

STOP: If the operator wishes to terminate a currently executing task, he can type STOP and then the name of the task. There is no effect if the task named is already dormant.

START: Use this word to begin execution of one of the system tasks. INPUT is used to obtain the name of the task. The command has no effect on the system if an attempt to start a task that is already executing.

SCHEDULE: If a task is needed at some future time, but not immediately, then the SCHEDULE command is used. Following this word, the operator should enter the name of the task. The system then responds with "TO START AT", and the operator should then enter the time in minutes and seconds relative to the start of reaction. The system response is "TO STOP AT" and the operator enters the time when the task should be stopped. The system then converts the two input times to seconds and stores these values in PINIT (a service routine for program initiation).

5.4 Task Routines

Each of the task routines can be started or stopped by the operator either directly or through program SCHEDULE. There are presently six tasks in the system SAMPLE, EMULSIFIER, MONOMER, INITIATOR, CONTROLT and LOG. They are each described

in detail below.

SAMPLE:

The function of this routine is to take an accurate volume of sample from the reactor, dilute with distilled water, mix to a uniform solution of known dilution, transfer the solution to the UV analyser, and start the UV analyser routine.

It performs this task by setting ten computer relays to either the closed (bit set to 1) or the open (bit set to Ø) position. These low voltage relay circuits in turn activate 110 volt relay circuits which activate the solenoid valves and automatic syringe of the physical sampling system. The relationship between the computer relays and the physical sampling system is given in table 2.

Before taking the next sample, the task checks that the time necessary for taking a sample (120 seconds) does not exceed the allotted time interval set by the operator SMPLINT. Another check is made on the other operator input SMPLVOL, the volume of sample to be taken. The program ensures that the volume does not exceed the maximum volume of the syringe. If either of these values is out of range, the routine does not take a sample during this interval. No further checks are necessary at this point since altering the sample volume or interval does not affect the routine until the next sample is taken.

Upon completing the above checks the routine begins opening and closing relays in a set time sequence, at the end of which the routine is ready to take another sample. An

Table 2: Relays used in SAMPLE routine

Relay No.	Physical action when relay closed
1	Solenoid valve connecting dilution vessel to gravity feed tank is opened.
2	Valve connecting dilution vessel to drain is opened.
3	Air bleed valve atop the dilution vessel is opened.
4	Power connected to automatic syringe.
5	With power to syringe, the syringe infuses (when not set syringe withdraws).
6	Valve between reactor and dilution vessel opens.
7	Valve between dilution vessel and UV analyzer cell opens.
8	The stirrer is started.
9	Connects the reactor sample line to vacuum.
10	Valve to reactor is opened.

internal subroutine RESET is used to call the relay routine and wait the desired time interval. Below is a list of each of the relay configurations used by this routine in the sequence in which they are called.

Relay configuration 1: 0007 Time: PURGE: 3 secs

Action: The reaction mixture is opened to vacuum in order to purge the sample line with the current reacting mixture. In the present set up 3 seconds was found to be adequate. However, if the length of sample line or amount of vacuum is altered, this interval should be changed. If PURGE should be changed, MINTM (the minimum time to complete a sample) should have a corresponding adjustment.

Relay configuration 2: 0125 Time: DRAW: 20 secs.

Action: A known volume of sample is withdrawn from the reactor into the dilution vessel. The current reaction time is saved for later use by UVANS in SSEC.

Relay configuration 3: 0004 Time: STINT: 5 secs.

Action: The stirrer only is activated during this time interval. At the end of this interval, the sample is ready for measurement by UVANS. A message is sent to UVANS to indicate that a sample has been prepared and is about to be transferred.

Relay configuration 4: 0310 Time: UVINT: 25 secs.

Action: The air bleed valve and valve to UV analyzer cell are opened. The prepared solution flows by gravity through the UV cell to drain. An interval of 25 secs was found necessary to completely purge the line and cell of the previous sample.

Relay configuration 5: 0610 Time: UVINT: 25 secs

Action: The main drain valve of the dilution vessel is also opened to reduce time required to empty.

Relay configuration 6: 0600 Time: EMINT: 15 secs

Action: Valve to UV spectrophotometer is closed to ensure no air gets into the UV cell. These 15 secs are used to complete emptying of the dilution vessel of all polymer sample.

Relay configuration 7: 1400 Time: FSHTM: 10 secs

Action: The dilution vessel is flushed with distilled water from the gravity feed tank.

Relay configuration 8: 1340 Time: DRAW: 20 secs

Action: The dilution vessel is filled with water and the syringe emptied. The air bleed is open to allow filling. This interval is the same as the syringe withdrawal time so that the syringe is in the correct position to take the next sample.

Relay configuration 9: 1200 Time: FILLR-DRAW: 15 secs

Action: Until the dilution vessel is completely filled, water from the gravity feed tank is allowed into the vessel.

Relay configuration 10: 1022 Time: SLFSH: 10 secs

Action: In order to ensure consistent readings from sample to sample, the sample line is backflushed as far as the vacuum system.

Relay configuration 11: 0000 Time: SMPLINT-MINTM

Action: All valves are closed and program remains dormant until next sample.

EMULSIFIER, MONOMER, INITIATOR:

These three routines have been grouped together since their functions and coding are similar. They are routines for controlling the rate of addition of each to the reactor. A relay is set by each routine to open a solenoid valve and allow the respective feed to flow by gravity into the reactor. A controlled "rate" of addition is accomplished by allowing the particular solenoid valve to be open for only a portion of each second that passes. Thus an approximation of constant flowrate is obtained.

The rates of addition to the reactor can be adjusted by the CHANGE variable command. The names for the flowrates of emulsifier, monomer and initiator are respectively FLOWE, FLOWM and FLOWI. It should be noted that these are the flowrates that will occur when the respective tasks are started either by the START or SCHEDULE commands.

CONTROLT :

This routine controls the temperature of the reacting mixture. The task is started by typing START (or SCHEDULE) CONTROLT at the teletype. This starts the routine that records and stores the reactor and jacket temperatures. However no control actually occurs unless TYPECONTROL is set to 1.

With TYPECONTROL set to 1, on-off control is initiated. That is the two solenoid valves controlling the hot and cold water supplies to the reactor jacket are opened as necessary. At present the interval for control is set at 1 second, however this can be altered by the variable TEMPCINT (tempera-

ture control interval), the control interval in units of seconds.

LOG:

This is the logging routine which outputs on the teletype printer pertinent data as the reaction progresses. It is the one task which does not require the START command. When command input has been completed simply type LOG<space> and the routine starts printing the data at the current logging interval LOGINT (units are in seconds). The routine stops printing whenever a CTROL A is entered on the teletype.

5.5 SERVICE ROUTINES:

There are seven service routines presently included in the system. They are RTDBØ, RTBDØ, MATHS, INFCE, CLOCK, PINIT and INPUT. The function of these programs is to provide a service which can be accessed by communication routines, task routines or by other service routines. Each will be described in detail below.

RTDBØ: This routine is called when a number is to be input via the teletype by the operator. The routines name is derived from its function, real time decimal to binary conversion. The program reads a decimal number of up to 5 digits plus sign from the teletype, converts these digits to a single binary word and then returns to the calling program with the result in accumulator 1.

Upon entering this routine, the character "#" is printed on the teletype. This is to indicate to the operator

that this routine has been entered and a decimal number of up to 5 digits is expected. The routine will accept as input "+", "-", space, or a digit 0 through 9. Any other character or symbol (other than CTROL A) causes the routine to terminate input. Transfer is made to the beginning of the routine, clearing all registers and printing "#" once more. This allows the operator to reenter a number whenever a typing error occurs. Typing CTRL A causes all current execution of I/O to be terminated and control transferred to INTER. A space indicates the end of the number and is the normal termination of this routine. The calculation of the binary number is completed and return is made to the calling program with the result in AC1.

RTBDØ: This is the reverse of the previous program. The routine's service is to take a binary number transmitted via ACØ, convert the number to base 10 and print the result on the teletype. AC1 should contain the necessary printing information. For the purpose of this program, AC1 is considered to consist of two 8 bit integers. The first 8 bits contain the position of the decimal point in the number to be printed. The value is equal to the number of digits after the decimal point. No decimal point is printed if the first 8 bits are all zero. The remaining 8 bits indicate the number of characters to be printed. This number is exclusive of the decimal point or the space which is always printed at the end of a number. If the value is negative, then the word in ACØ is printed in octal, no decimal point is used, and 6 digits are always printed. If the value is zero then the word in ACØ is printed in decimal

but only the minimum number of characters necessary to complete the number are used. This option is especially useful when the number of digits is not known and a special format is not desired such as in the READ and CHANGE commands.

MATHS: This service routine is actually a collection of all the math routines required by the system. The routines are all re-entrant which means that more than one task can access the routines at the same time without affecting the result. A re-entrant task stores all working numbers either in the calling program or accumulators. Each routine is described separately below.

BIDEC: This routine converts the binary number supplied through AC0 to ascii code. The ascii code is stored in the first 6 locations following the call statement. The seventh location following the call is used by BIDEC as a temporary storage register. The first location contains the sign of the converted number; ascii "-" for negative numbers and ascii "0" for positive numbers. The call sequence for BIDEC would then appear in the calling program as:

```
JSR @ <location of address of BIDEC>
.BLK 7
<Return>
```

MULTY: This is a simple single precision multiply routine. AC0 is multiplied by AC1 and the result returned in AC0 and AC1. The high order part of the result is in AC0 and the low order in AC1.

DIVID: This is the single precision divide routine. Upon calling this routine AC0 and AC1 should contain the numerator

and AC2 the denominator. The result of the division is returned in AC1.

INFCE: This program is a collection of routines which handle input/output to the various devices. Each of these routines acts independently of the others.

RELYS: Output to the relays is controlled by this routine. The routine maintains a record of the current bit pattern of the relays and outputs the new pattern whenever called from the other tasks. The call sequence is:

```
.FORK
1
RELYS
```

ACØ should contain the new relay configuration. Each bit represents a relay channel. The respective bit is set to 1 to close the relay and Ø to open. AC1 should contain the relays that should not be changed at this time. For example, if the contents of ACØ are Ø51ØØØ and AC1 are 1ØØ777, then relays 2, 4 and 7 will be closed and 3,5 and 6 will be opened. The contact positions of relays 1 and 8 through 16 will remain as they were.

TTØO: This routine outputs a string of characters in byte format (two ascii characters per word) to the teletype. The last byte of the string is a null character (ØØØ). The call sequence is:

```
JSR   TTØO
<address of byte pointer>
<Return>
```

The byte pointer contains a numerical value equal to twice the address location of the first byte in the character string.

TRANS: At the present TRANS reads all 16 of the A/D input

channels continuously whenever the CPU is not busy with other parts of the system. It runs at the lowest priority. Note, that when the system is expanded to include long and time consuming calculations, it will be necessary to split CPU time between these and TRANS. It will be necessary to increase the priority of TRANS and create CPU time for the long calculations by use of the .WAIT command.

TRANS contains a coarse type of digital filter used on A/D channels Ø and 1. A continuous average of the last 37 readings from these two channels is maintained. In the present system this was found to be an acceptable method of measuring temperature. However, as the CPU becomes overloaded, a more sophisticated filter can be written. There is no evidence of time delay using this method of measurement. CONTROLT uses the measurements once every second in controlling the reactor temperatures and, during this time period, many averages are computed.

CLOCK: This routine keeps track of the real time. The time of day can be accessed by the operator using the CHANGE and READ commands and the words HOUR, MINUTE and SECOND. CLOCK also maintains a record of the time relative to the start of reaction. This can be accessed by REACTIME from the teletype. From within the system, routines can access these values by reading the locations of HOUR, MINIT, SECND and RTIME. CLOCK uses the .WAIT command to determine the passage of one second, then increments SECND and RTIME. When SECND and MINIT reach a value of 60, MINIT and HOUR respectively are incremented. Time is based on a 12 hour clock, thus HOUR is set to 1 when-

ever 13 is reached.

PINIT: This section coordinates the execution of tasks as set by the SCHEDULE command. With the passage of each second, a "pulse" is received from the CLOCK routine. PINIT then checks its stack of programs to determine if any should be started or stopped at this time. The stack of programs, ITEMS, is a list of starting locations of tasks with the times when they should be started and stopped. Three locations are used for each SCHEDULE command. At present there is room for 7 programs. This can be increased by merely lengthening the ITEM table.

INPUT: Routine INPUT reads a word from the teletype. The word is terminated by a space. Each character, as entered, is coded by taking the 7 bit ascii configuration and adding to the previously calculated code and shifting the result one bit to the left. The code formed in this manner, upon being terminated with a space, is compared with a table of codes. If the code is found in the table, INPUT returns to the calling program with the core address, associated with the code, in accumulater 1. If the code of the input word is not found in the table, then INPUT causes the error message NO to be printed and control is passed to the program executive, INTER.

INPUT can be entered at three separate locations, INPTT, INPTP and INPTD. Entry to the program at these addresses causes a different table of acceptable codes to be used. INPTT is called when the address of a task routine is required; only task names will be found in the table searched

by INPUT. INPTP is used when one of the system programs or communications routines are desired, and INPTD for the address of a data location. As an example of how INPUT is used, consider the command line

READ TEMPSP

After issuing CTROL A, the executive is called into action, terminating the log routine and entering INPUT at INPTP. INPUT then reads the character string READ<space>. Upon encountering the space character, INPUT completes the code which for this word is 2152. The code is compared with the items in the code table for communication routines. In this case the code is found, so INPUT loads the address corresponding to the READ communication routine into accumulator 1. Control of the CPU is transferred back to INTER which then calls the READ routine at the address supplied by INPUT. READ routine now needs the address of a data location, so INPUT is again called but this time at INPTD. TEMPSP<space> is then read from the teletype. The resultant code, 11556, is compared with the table of data locations. The code is found, so INPUT loads the address corresponding to the input word and returns it via AC1 to the READ routine.

The routine was written in the above manner for two reasons. Firstly separate tables were used so that when a data location is required, the address of a routine cannot be obtained. Similarly the system will not start executing a string of data. Secondly input words are not limited by length so that names can closely approximate real words. Thus operator communication with the system is easy to follow.

5.6 STARTUP AND RUN PROCEDURE

This procedure is limited to the startup and running of the computer system. Procedure relating to the reaction itself may be found elsewhere (Keung(1)).

(i) Charge the monomer, emulsifier and initiator to the respective feed tanks.

(ii) Load the computer program into core.

(iii) Set the current time and time until the reaction is to start using the CHANGE command.

(iv) Set the time when the computer should feed the reactor with monomer, emulsifier and initiator using the SCHEDULE command.

(v) Set the reaction temperature using CHANGE TEMPSP and start the temperature control routine with START CONTROLT

(vi) Schedule the sample routine and set desired sample volume (SAMPLE VOL) and sample interval (SAMPLINT).

(vii) Start the logging routine.

6. EXPERIMENTAL RESULTS AND DISCUSSION

6.1 TEMPERATURE CONTROL

A simple on-off control scheme on hot and cold water supply was used to control the reactor temperature with the computer. This type of control was sufficiently accurate for these experiments. The reactor temperature was held within ¼°C of the desired set point at all times.

6.2 INDUCTION PERIOD

An important problem encountered in industrial reactors for vinyl acetate polymerization is determining the start of reaction. Depending on the amount of oxygen and other impurities present in the feed materials, there is a certain length of time, called the induction period, in which no reaction takes place. This induction time can be anywhere from 0 to 60 minutes.

In this study the induction period was determined accurately by monitoring the absorption near the start of the reaction. This is demonstrated in table 3. Three readings prior and two readings following the start of reaction are given. The time interval between measurement is 5 minutes. The absorbance values are those obtained from the UV analyser, corrected for baseline and multiplied by 1000. A significant increase in absorbance can be seen at the start of the reaction.

Experiments 6 through 8 were not included in table 3. In experiment 6 no reaction occurred after 2 hours; at which

time the experiment was aborted. The absorbance remained constant throughout the experiment. This lack of reaction occurred when the reactants were fed via the feed tanks rather than being added manually. Brass solenoid valves were used. The copper in the brass probably interfered with the initiator making it inactive. This has been reported by other investigators (6). After this experiment, stainless steel solenoid valves were ordered but did not arrive before the conclusion of this study. Experiments 7 and 8 were not included in the table since there was no detectable induction period. This was attributed to the improved procedure of removing trace amounts of oxygen from the supply nitrogen by bubbling the gas through a 5% pyrogallol solution in 2N sodium hydroxide.

TABLE 3: Corrected Absorbance Readings at the Start of Reaction (Wavelength = 3000 Å)

Experiment No.	Ax1000 Sample 1	Ax1000 Sample 2	Ax1000 Sample 3	Start of Reaction	Ax1000 Sample 4	Ax1000 Sample 5
1	0	0	0		8	44
2	0	-1	0		4	13
3	0	0	2		13	28
4	0	2	3		--	291
5	0	0	0		11	70

6.3 PARTICLE SIZE DISTRIBUTION

The Mie theory presents a method by which the number of polymer particles and the particle size distribution can be determined from measurements of absorbance. The Mie theory predicts that for a monodispersed spherical suspension

$$\tau^* = K^* \pi r^2 N \tag{1}$$

where τ^* is the turbidity of a dispersion defined by

$$\tau^* = 1/\ell \ \ell n(Io/I) \tag{2}$$

r is the ratius of the particle (cm)

K* is the light scattering coefficient

N is the number of particles (#/cc)

ℓ is the path length of transmission cell (cm)

Io,I are the incident and transmitted intensities.

Since the absorbance, A, is defined as

$$A = \log_{10}(Io/I) \tag{3}$$

and the length of transmission cell in this study was 1.000 cm

$$\tau = 2.303 \ A. \tag{4}$$

For a polydispersed spherical suspension the turbidity, τ can be determined by the summation of the individual effects of each particle, that is

$$\tau = N \int_0^\infty K^* \pi r^2 \ f(r)dr \tag{5}$$

where f(r)dr is the number frequency of particles in the size range r to r + dr.

If an analytical form for the particles size distribution is now assumed, then by measurement of turbidities at several wavelengths, the particle size distribution, the

number of particles and thus the concentration of particles can be calculated. This is explained in Appendix A.

The β and generalised exponential distributions were compared using a minimum sum of squares deviation as a criteria of fit. The particle size distributions measured by Keung(1) were used as the basis for comparison. The β distribution was found to give a closer approximation to the electron microscope data. Table 4 shows the calculated values of $\bar{d}n$ and $\bar{d}w$ compared with the electron microscope values. Note that the weight avarage particle diameter compares favourably with the electron microscope data in most cases and has the same order of error as the light scattering data, also from Keung(1).

TABLE 4: Comparison of Number and Weight Average Particle Diameters using the β-Distribution

Expt #	$\bar{d}n$(E.M.) (Å)	$\bar{d}n(\tau)$ (Å)	$\bar{d}w$(E.M.) (Å)	$\bar{d}w(\tau)$ (Å)	% Dev. $\bar{d}n$	% Dev. $\bar{d}w$	% Dev. (E.M./L.S.)
A1	1090	1280	1650	1604	+17	-3	-5
A2	685	1050	1230	1380	+53	+12	-9
A3	420	770	770				-9
A4	520	874	910	1320	+68	+35	+21
A5	840	1056	1360	1390	+25	+2	+5
A6	490	722	850	1084	+47	+27	+10
A7	650	920	1160	1262	+41	+9	+7

However, the number average particle diameters are higher than electron microscope data by approximately 50%. This greater error in the number average particle diameter is to be expected. When the particle size is very small compared with the wavelength of light, only the weight average particle diameter can be measured as explained below.

A problem arises in the ability to calculate the particle size distribution when the particles are very small, or when a large portion of the distribution are small particles. This can be demonstrated by considering the scattering coefficient, K*. For very small particles this coefficient approaches that for Rayleigh scattering

$$K^* = k\frac{r^4}{\lambda^4} \tag{6}$$

where k is a constant dependent only on the dispersion media and independent of the wavelength or particle size. Since

$$\tau = N \int_0^\infty K^* \pi r^2 f(r)dr \tag{5}$$

Then for Rayleigh particles

$$\frac{\tau \lambda^4}{N} = k \int_0^\infty r^6 f(r)dr \tag{7}$$

or in terms of concentration

$$\frac{\tau \lambda^4}{C} = \frac{k \int_0^\infty r^6 f(r)dr}{\rho \frac{4}{3} \pi \int_0^\infty r^3 f(r)dr} = \frac{3k}{4\pi\rho} \overline{rw} \tag{8}$$

The right hand sides of equations (7) and (8) are

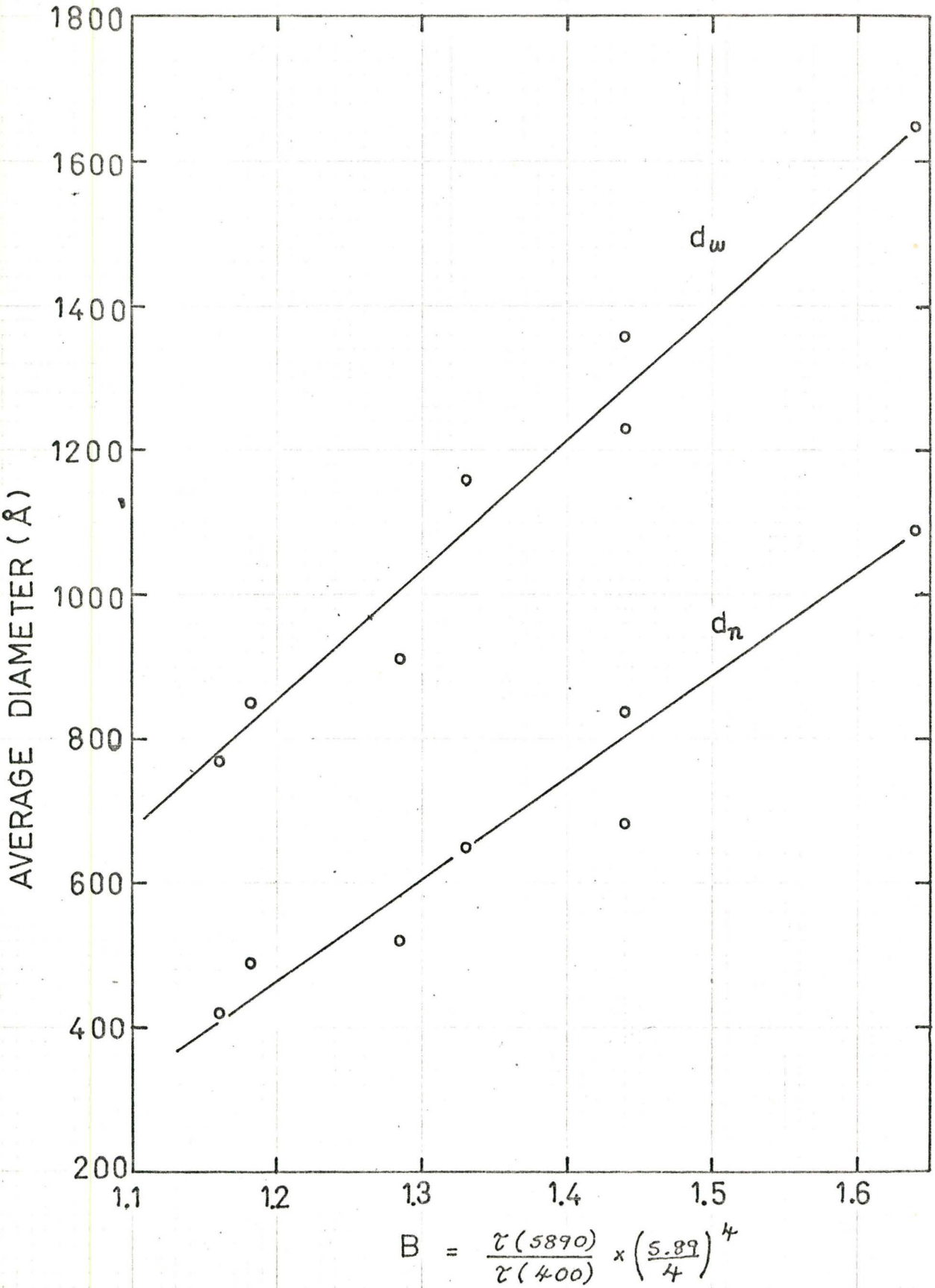
constant, independent of the wavelength and distribution function (f(r). Only one moment of the distribution can be calculated, commonly termed the turbidity average particle diameter. This value is the same as the weight average particle diameter used in this study.

In order to further examine the results of the experiment, the B function was defined. B is the turbidity ratio

$$B = \frac{\tau_2 \lambda_2^4}{\tau_1 \lambda_1^4} \tag{9}$$

The turbidity ratio has several rather important qualities. Firstly it is independent of both the concentration of sample and the number of particles within the sample. It is a function only of the particle size distribution. Secondly, for a similar distribution, average particle diameters can be correlated with the turbidity ratio. This is demonstrated with the data from Keung(1) where the turbidity ratio with λ_1 = 4000 Å and λ_2 = 5890 Å is correlated with the average particle diameters measured by electron microscopy. Figure 3 defines this relationship for the various particle size distributions used in this study and by Keung. The third important quality of the B function is that several qualitative conclusions can be made about the numerical value of the turbidity ratio. The lower limit of B is 1.0, this is the value of B when there are only very small particles in the distribution. That is, when the particles are in the Rayleigh scattering region and the term $\tau\lambda^4$ does not vary with wavelength.

Fig. 3 AVERAGE PARTICLE SIZE versus B



Narrow particle size distributions cause lower values for B. It can be shown that for the β distribution, developed in Appendix A, as the distribution becomes wider (p,q parameters are smaller) the function B becomes larger.

Figure 4 shows the B values obtained in this study for λ_2 = 6000 Å and λ_1 = 4000 Å. Note that near the start of the reaction the turbidity ratio is high. The probable reason is that early in the reaction a wide distribution of particles is obtained. As soon as a polymer particle is formed, it grows very rapidly but new particles continue to form resulting in the wide distribution. At the end of stage 1, however, new particles are no longer formed. The distribution becomes narrower as the smaller particles grow more rapidly because of their large surface area. The B value decreases until the particle size distribution becomes established, near the end of stage 2 polymerization. The particles then continue to grow and B increases correspondingly. At this point particle diameters can be correlated with the turbidity ratio as indicated in figure 3.

Figures 5 and 6 show how the turbidity ratio using wavelengths of 3000 Å and 4000 Å varies throughout the reaction. The increase of B with the growth of the particles is clearly demonstrated. Note that using these higher frequency light waves, the turbidity ratio appears more dependent on the average particle diameter than the shape of the particle size distribution.

Fig. 4 TURBIDITY RATIOS

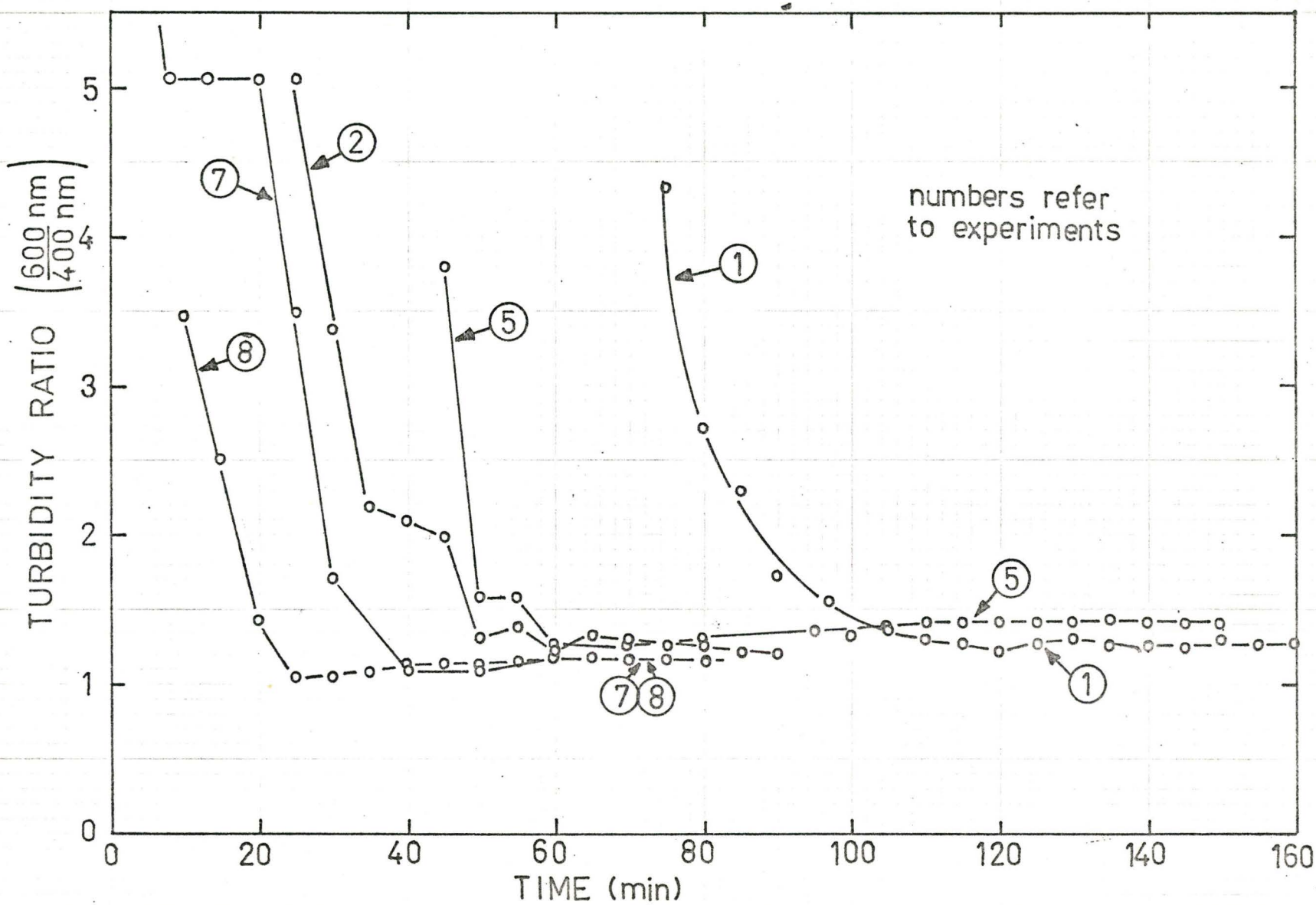


Fig. 5 TURBIDITY RATIOS, EXPTS 1-5

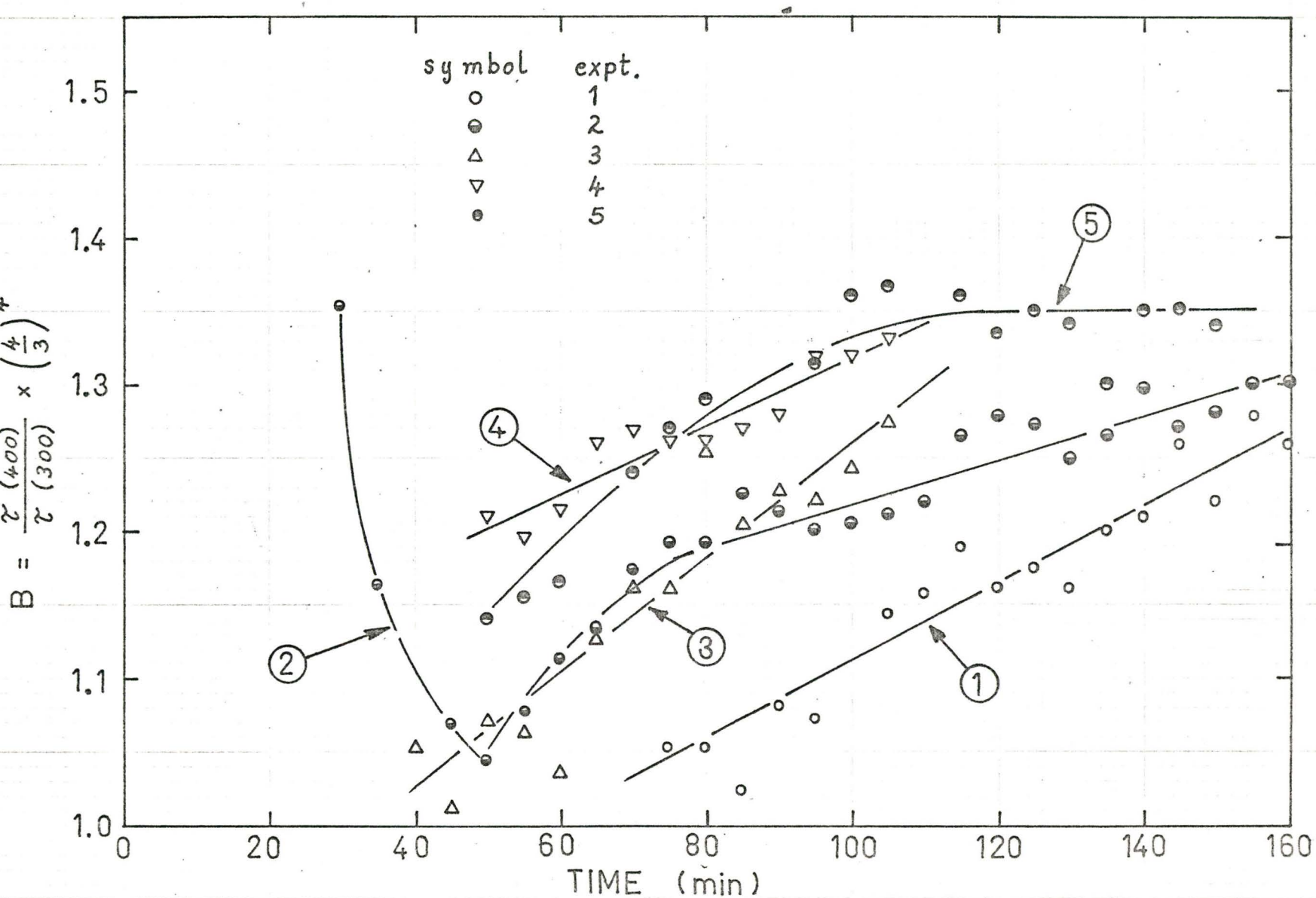
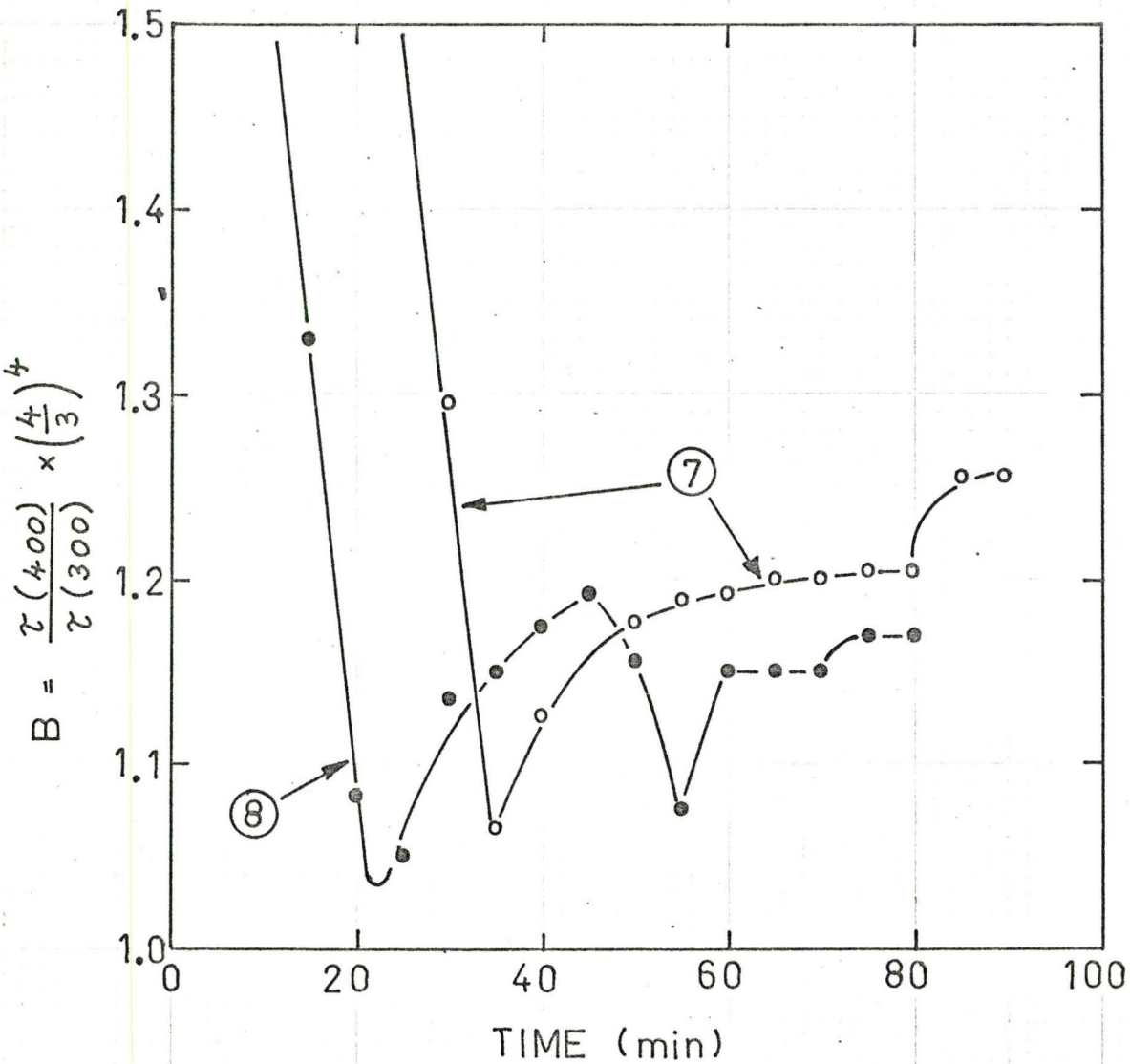


Fig. 6 TURBIDITY RATIOS, EXPTS 7, 8



7. CONCLUSIONS AND RECOMMENDATIONS

A hardware and software system was developed for on-line measurement of the turbidity of a latex in growth. The interaction between the user and the software was simplified so that an engineer with minimal experience can use the system. The software system is modular in that additional routines can be built into the system with a minimum of effort. The computer controlled sampling system allows fast and accurate preparation of samples for measurement by the U.V. spectrophotometer. It is recommended that the U.V. spectrophotometer now be interfaced with the minicomputer so that full use of this sampling system can be made. The computer can prepare a sample every two minutes but manual measurement of the sample takes over four minutes so that, for this study, samples were taken at five minute intervals. A computer controlled spectrophotometer would reduce overall sample analysis to less than two minutes and would eliminate lost samples due to operator error, which sometimes occurred during a run. This computer system can now be used to assist in the development of a control strategy for the latex reactor.

A preliminary study was made to develop techniques to convert turbidity spectra into particle size distributions. The results show that there is a definite correlation between the turbidity and the size distribution. It was found that

the β distribution gave a reasonably precise description of the particle size distribution. The parameters of the β distribution show sensitive variation with conversion. A B function was defined which is sensitive during the early stages of the polymerization and is probably related to the nucleation phenomena.

To control the latex reactor it is not necessary to follow the exact particle size distribution. A possible approach to the problem of control is to correlate the turbidity spectrum with the desired end product properties. In other words, the turbidity spectrum can give a measurement of the particle size distribution and the particle size distribution determines the properties of the latex. However, the calculation of the particle size distribution can be bypassed. If the effects of the various variables of the reaction on the turbidity spectrum are determined, then to control the reaction simply follow a predetermined turbidity curve, or properties of the turbidity curve such as the turbidity ratio at certain wavelengths.

8. NOMENCLATURE

A	Absorbance
A*	Equivalent absorbance
A/D	Analog to digital
B	urbidity ratio
CPU	Central processing unit
D/A	Digital to Analog
K*	Light scattering coefficient
N	Number of particles per unit volume
RTOS	Real time operating system
Å	Angstrom
dn	Number average particle diameter
dw	weight average particle diameter
ℓ	path length transmission cell
p	parameter of β distribution
q	parameter of β distribution
r	particle radius
r_n	number average particle radius
r_w	weight average particle radius
λ	wavelength of light
ρ	density of polymer
τ*	turbidity

9. REFERENCES

(1) Keung, J., M.Eng. Thesis, McMaster University, Hamilton, Ontario, Canada (1974).

(2) Goosney, D., M. Eng. Thesis, McMaster University, Hamilton, Ontario, Canada (1973).

(3) Smith, W.V., Ewart, R.H., J. Chem. Phys. 16, 592 (1948).

(4) Smith, C.L., Digital Computer Process Control, Intext Educational Publishers, Toronto (1972).

(5) RTOS operating manual, Data General Inc.

APPENDIX A PARTICLE SIZE DISTRIBUTION FROM LIGHT TRANSMISSION MEASUREMENTS

As discussed in section 6.3 the turbidity can be calculated from the expression

$$\tau = N \int_0^\infty K^* \pi r^2 f(f) \, dr \qquad (5)$$

K^* is a complicated function of the ratio r/λ but can be approximated with a high degree of accuracy (when $r/\lambda < 1.4$) by the expression

$$\pi r^2 K^* = 55.818 \frac{r^4}{\lambda^4} - 650.72 \frac{r^8}{\lambda^8} + 1860.8 \frac{r^{10}}{\lambda^{10}} \qquad (10)$$

If a function is now chosen to represent the particle size distribution, then the distribution and number of particles can be determined by solving n simultaneous equations; where n is one greater than the number of parameters in the chosen distribution. This is demonstrated in the remainder of this section with the β distribution.

For the β distribution, the particle size distribution is represented by

$$f(r) = \frac{1}{\beta(p,q)} r^{p-1} (1+r)^{-(p+q)} \qquad (11)$$

where p and q are the parameters of the distribution and r is the particle radius. Upon substitution of this distribution

into equation (5) the following expression is obtained

$$\tau = \frac{N}{\beta(p,q)} \int_0^\infty K^* \pi r^2 \, r^{p-1}(1+r)^{-(p+q)} dr \qquad (12)$$

Substitution of $K^* \pi r^2$ with equation 10 and integrating the result is the following expression.

$$\tau = \frac{N}{\beta(p,q)} \left[\frac{55.818}{\lambda^4} \beta(p+6,q-6) - \frac{650.72}{\lambda^6} \beta(p+8,q-8) \right.$$

$$\left. + \frac{1860.8}{\lambda^8} \beta(p+10,q-10) \right] \qquad (13)$$

Using the relationship between the β and Γ functions

$$\beta(p,q) = \frac{\Gamma(p)\Gamma(q)}{\Gamma(p+q)} \qquad (14)$$

and rearranging 13, the following is obtained.

$$\frac{\tau\lambda^4}{N} = 55.818 \frac{\Gamma(p+6)\Gamma(q-6)}{\Gamma(p)\Gamma(q)} - \frac{650.72}{\lambda^2} \frac{\Gamma(p+8)\Gamma(q-8)}{\Gamma(p)\Gamma(q)}$$

$$+ \frac{1860.8}{\lambda^4} \frac{\Gamma(p+10)\Gamma(q-10)}{\Gamma(p)\Gamma(q)} \qquad (15)$$

This equation can be further simplified by making use of the relationship

$$\Gamma(n) = (n-1)\Gamma(n-1) \qquad (16)$$

equation 15 can be rewritten as

$$\frac{\tau\lambda^4}{N} = 55.818 \, \gamma \left[1. - \frac{11.658}{\lambda^2} \alpha \left(1. - \frac{2.8596}{\lambda^2} \delta\right)\right] \qquad (17)$$

where $$\gamma = \frac{(p+5)(p+4)(p+3)(p+2)(p+1)\,p}{(q-1)(q-2)(q-3)(q-4)(q-5)(q-6)} \qquad (18)$$

$$\delta = \frac{(p+8)(p+9)}{(q-9)(q-10)} \tag{19}$$

$$\alpha = \frac{(p+6)(p+7)}{(q-7)(q-8)} \tag{20}$$

If turbidities are now measured at three different wavelengths, then the above becomes a system of three equations and three unknowns. The parameters, p, q and N, can be calculated as explained below.

$$\text{Let} \quad E = \frac{\tau_2\lambda_2^4 - \tau_1\lambda_1^4}{\tau_3\lambda_3^4 \quad \tau_1\lambda_1^4} \tag{21}$$

$$\text{Then } E = \frac{\lambda_1^2\left(1-\left(\frac{\lambda_1}{\lambda_2}\right)^2\right) + 2.8596\ \delta\left(\left(\frac{\lambda_1}{\lambda_2}\right)^4 - 1\right)}{\lambda_1^2\left(1-\left(\frac{\lambda_1}{\lambda_3}\right)^2\right) + 2.8596\ \delta\left(\left(\frac{\lambda_1}{\lambda_3}\right)^4 - 1\right)} \tag{22}$$

Rearranging

$$\delta = \frac{\lambda_1^2\left[E\left(\left(\frac{\lambda_1}{\lambda_3}\right)^2 - 1\right) - \left(\left(\frac{\lambda_1}{\lambda_2}\right)^2 - 1\right)\right]}{2.8596\left[E\left(\left(\frac{\lambda_1}{\lambda_3}\right)^4 - 1\right) - \left(\left(\frac{\lambda_1}{\lambda_2}\right)^4 - 1\right)\right]} \tag{23}$$

$$\text{Let} \quad \beta = \frac{\tau_3\lambda_3^4}{\tau_1\lambda_1^4} \tag{9}$$

$$\text{then from 17 } \alpha = \frac{(1-B)\ \lambda_1^2}{11.658\left[\left(\frac{\lambda_1}{\lambda_3}\right)^2\left(1-\frac{2.8596\ \delta}{\lambda_3^2}\right) - B\left(1-\frac{2.8596\ \delta}{\lambda_1^2}\right)\right]} \tag{24}$$

p and q can now be calculated from α and β using equations 19 and 20. Upon determination of these parameters, the number and weight average particle diameters can be calculated as follows

$$\bar{r}_n^3 = \frac{\int_0^\infty r^3 f(r)\,dr}{\int_0^\infty f(r)\,dr} \tag{25}$$

Upon substitution of the β distribution (11) for $f(r)$, noting that $\int_0^\infty f(r)\,dr = 1$, and integrating

$$\bar{r}_n = \left[\frac{p(p+1)(p+2)}{(q-1)(q-2)(q-3)}\right]^{1/3} \tag{26}$$

Similarly

$$\bar{r}_w^3 = \frac{\int_0^\infty r^6 f(r)\,dr}{\int_0^\infty r^3 f(r)\,dr} \tag{27}$$

$$\bar{r}_w = \left[\frac{(p+3)(p+4)(p+5)}{(q-4)(q-5)(q-6)}\right]^{1/3} \tag{28}$$

The number of particles per unit volume of solution, N, can be calculated from equation 17. The concentration can then be determined from

$$c = N\rho \int_0^\infty 4/3\ \pi r^3\ f(r)\,dr \tag{29}$$

That is

$$c = N\rho\, 4/3\pi\, \frac{p(p+1)(p+2)}{(q-1)(q-2)(q-3)} \tag{30}$$

APPENDIX B. EXPERIMENTAL RESULTS

In this section, the results obtained from the 8 experiments carried out during the summer of 1974 are given. The measurement from the UV spectrophotometer is reported as equivalent absorbance rather than the actual reading. Equivalent absorbance is the absorbance that would be obtained from an undiluted sample of latex, if no secondary scattering occurred, that is

$$A^* = A \times \frac{Vd}{Vs}$$

where A^* equivalent absorbance

A absorbance measured on UV spectrophotometer

Vd total diluted volume

Vs volume of latex from reactor

A^* is given since the value is independent of the dilution system.

The experimental reaction conditions are given below:

	Expts. 1-5	Expts. 6-8
Temperature	50°C	50°C
Volume monomer	800 ml	400 ml
Volume water	1750 ml	1750 ml
Initiator	1.2 g	1.2 g
Emulsifier	15 g	12.5 g
Stirrer speed	150 rpm	150 rpm

Table B1: EQUIVALENT ABSORBANCE, REACTION 1

Sample No.	Time (Min)	Wavelength 600 nm	Wavelength 500 nm	Wavelength 400 nm	Wavelength 350 nm	Wavelength 300 nm
7	55	0	0	0	0	0
8	60	.4	.3	.2	.2	.4
9	75	.6	.6	.7	1.1	2.1
10	80	.7	.8	1.3	2.1	3.9
11	85	1.0	1.3	2.2	3.7	6.8
12	90	1.3	2.0	3.8	6.1	11.1
13	95	1.8	3.0	5.8	9.5	17.1
14	105	4.2	7.4	15.7	25.6	43.3
15	110	4.7	8.4	18.3	29.3	50.0
16	115	6.3	11.7	25.2	40.8	67.0
17	120	5.2	9.2	21.4	34.0	58.2
18	125	7.7	14.3	30.9	49.1	83.1
19	130	9.2	16.9	36.2	57.4	98.9
20	135	10.7	20.0	43.2	68.4	112.8
21	140	12.5	23.8	50.2	79.1	130.0
22	145	14.2	26.9	57.5	90.5	143.8

Table B1: (continued)

Sample No.	Time (Min)	Wavelength 600 nm	Wavelength 500 nm	Wavelength 400 nm	Wavelength 350 nm	Wavelength 300 nm
23	150	10.5	19.3	40.9	63.5	107.1
24	155	14.3	27.4	58.1	91.3	143.8
25	160	16.2	31.4	64.0	100.5	160.1
26	165	18.0	33.8	70.3	110.4	179.3
27	170	19.6	36.7	76.6	118.8	---*

*Sample is above the measurement limit of spectrophotometer

Fig. B1 EQUIVALENT ABSORBANCE REACTION 1

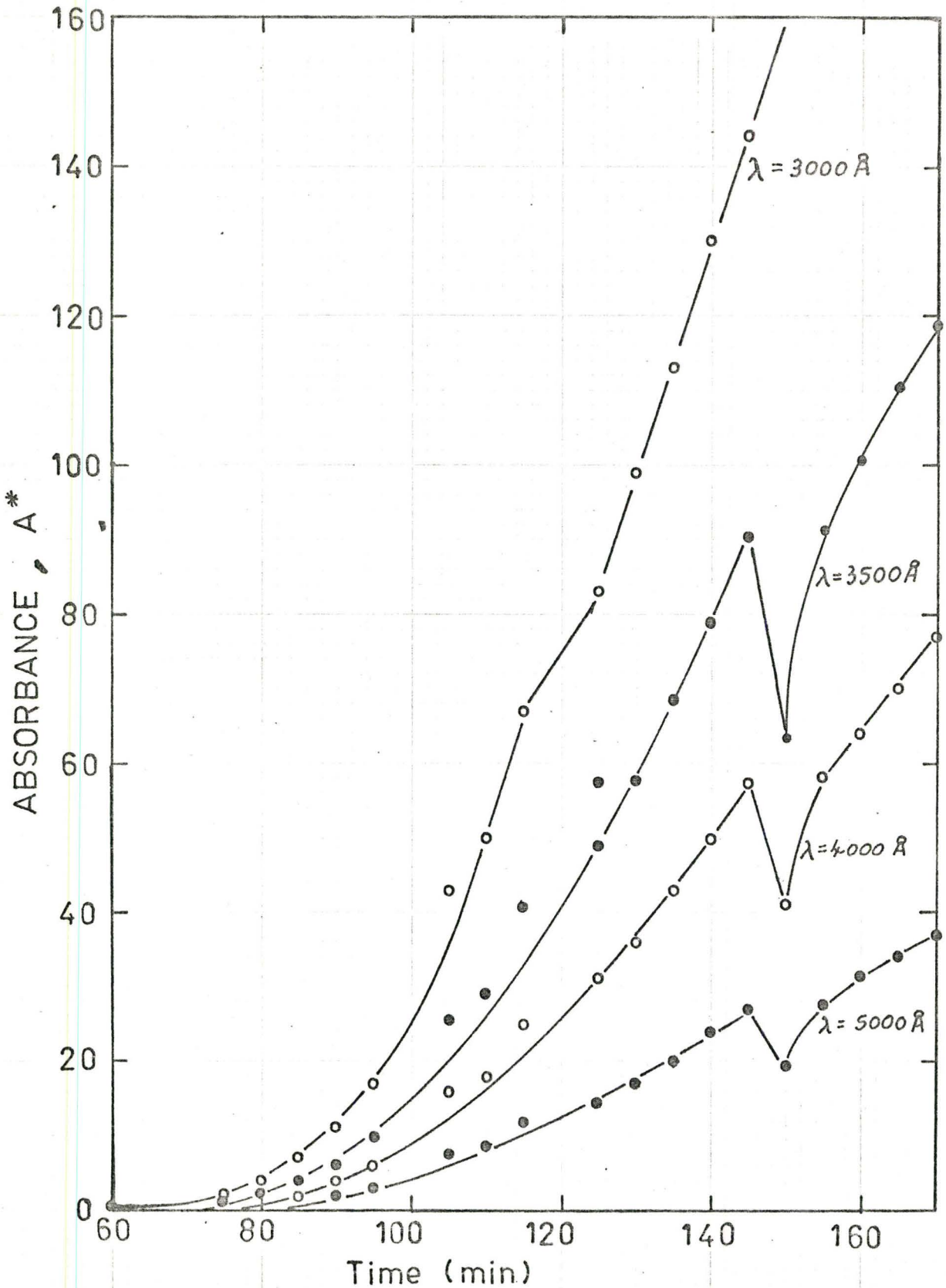


TABLE: B2 EQUIVALENT ABSORBANCE, REACTION 2

Sample No.	Time (Min)	Wavelength 600 nm	Wavelength 500 nm	Wavelength 400 nm	Wavelength 350 nm	Wavelength 300 nm
4	20	0	.1	.1	0	0
5	25	.2	.2	.2	.1	.2
6	30	.2	.3	.3	.4	.7
7	35	.3	.5	.7	1.0	1.9
8	40	.5	.7	1.2	1.9	4.0
9	45	.8	1.3	2.4	3.9	7.1
10	50	1.0	1.8	3.8	6.3	11.5
11	55	1.6	2.7	5.8	9.5	17.0
12	60	2.1	3.9	8.6	14.2	24.4
13	65	3.1	5.3	11.7	19.2	32.6
14	70	4.1	7.1	15.5	24.8	41.7
15	75	4.8	8.7	19.4	30.5	51.4
16	80	5.8	10.6	23.3	37.0	61.8
17	85	6.8	13.3	28.7	45.2	74.0
18	90	7.9	15.4	33.0	51.9	86.0
19	95	9.5	17.2	36.8	57.7	97.1

continued

TABLE: B2 (continued)

Sample No.	Time (Min)	Wavelength 600 nm	Wavelength 500 nm	Wavelength 400 nm	Wavelength 350 nm	Wavelength 300 nm
20	100	10.6	19.5	41.7	66.0	109.4
21	105	12.0	22.2	47.1	73.7	122.8
22	110	13.3	24.8	51.6	81.8	133.7
23	115	15.1	27.8	58.5	91.5	146.2
24	120	16.4	30.4	64.0	99.5	158.4
25	125	--	33.2	70.0	107.8	173.8
26	130	20.2	37.2	77.6	119.4	197.2
27	135	21.8	40.2	84.0	129.4	209.8
28	140	21.2	39.6	83.0	127.9	202.2
29	145	24.8	45.0	93.0	142.0	232.0
30	150	25.2	46.3	95.9	149.2	237.4
31	155	27.0	51.0	104.2	159.8	254.2
32	160	29.2	53.0	109.2	169.0	266.0
33	165	30.2	56.0	114.2	174.0	278.0
34	170	32.8	59.6	121.1	186.9	296.0

Fig. B2 EQUIVALENT ABSORBANCE REACTION 2

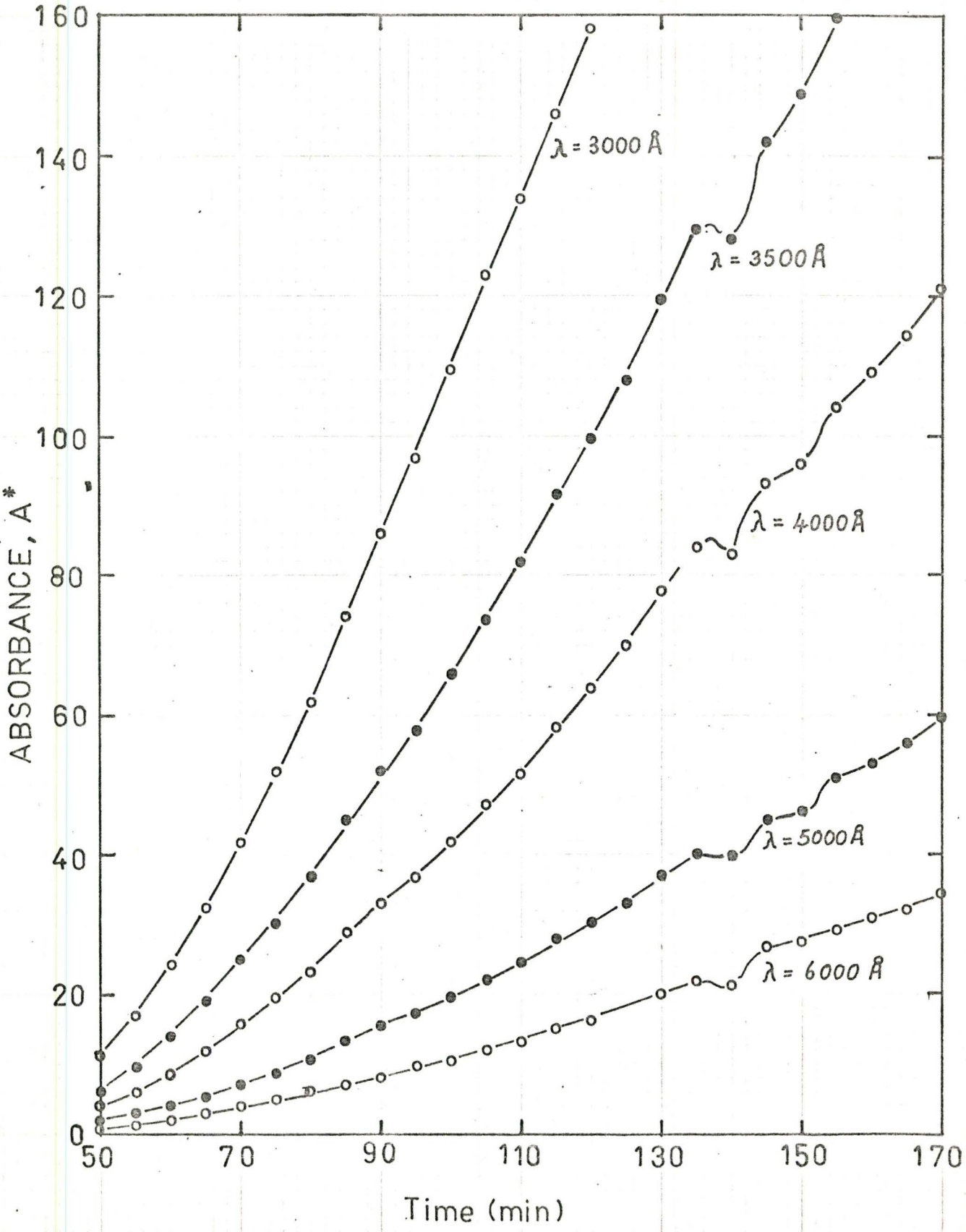


TABLE: B3 EQUIVALENT ABSORBANCE, REACTION 3

Sample No.	Time (Min)	Wavelength 600 nm	Wavelength 500 nm	Wavelength 400 nm	Wavelength 350 nm	Wavelength 300 nm
7	30	0	0	0	0	0
8	35	.1	.1	.1	.1	.2
9	40	.1	.2	.3	.5	.9
10	45	.2	.3	.8	1.4	2.5
11	50	.5	.9	2.0	3.1	5.9
12	55	1.0	1.6	3.6	6.0	10.7
13	60	1.5	2.6	5.5	9.2	16.8
14	65	2.4	4.2	9.1	14.5	25.3
15	70	3.0	5.5	12.3	19.9	33.4
16	75	4.3	7.7	16.8	26.9	45.8
17	80	5.3	9.8	21.9	34.7	55.2
18	85	6.6	12.3	25.7	41.5	67.4
19	90	8.0	14.9	31.8	49.6	81.9
20	95	9.5	17.5	36.9	57.4	95.5
21	100	12.0	21.4	45.3	70.0	115.1
22	105	12.9	24.2	50.8	78.3	125.8

Fig.B3 EQUIVALENT ABSORBANCE REACTION 3

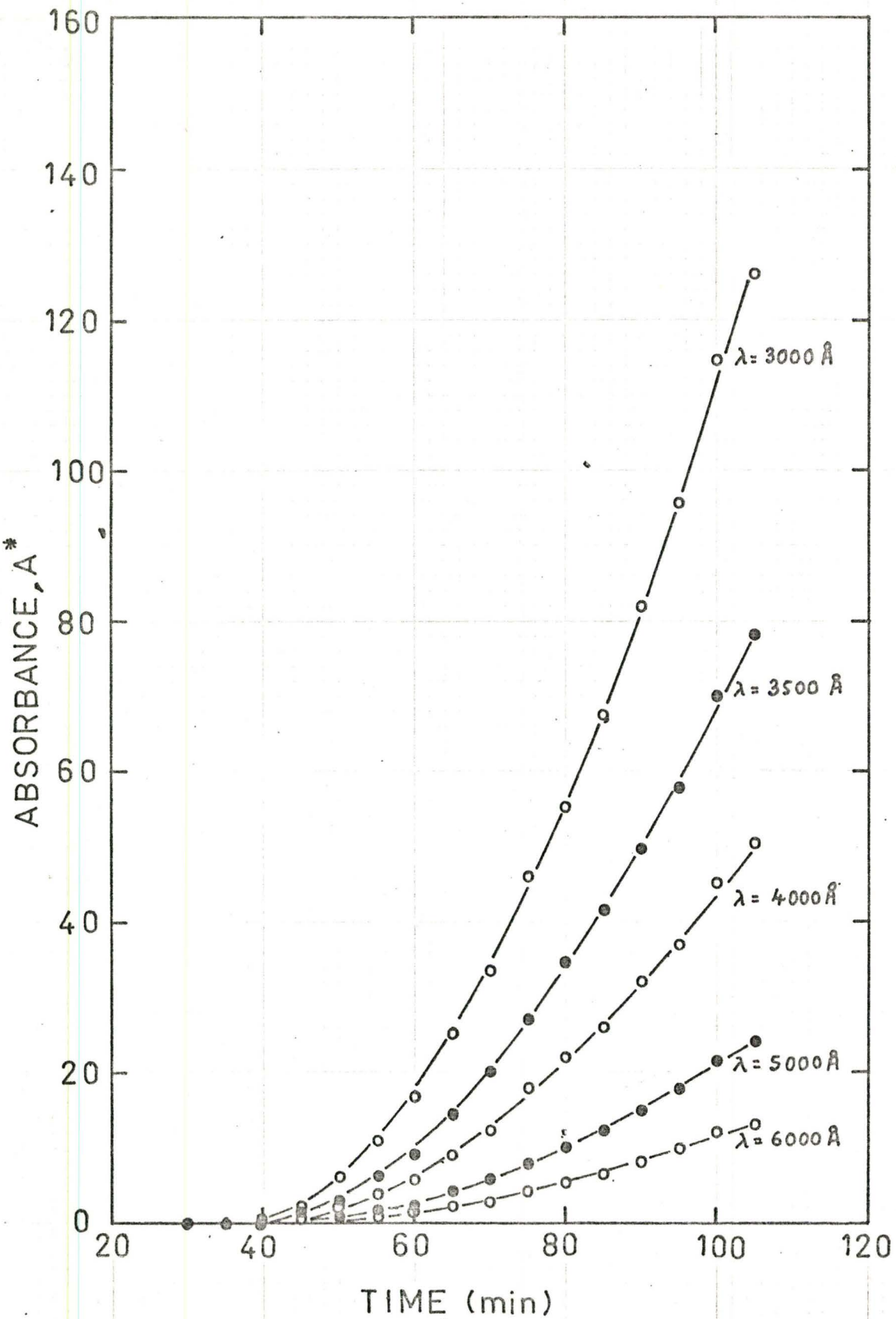


TABLE: B4 EQUIVALENT ABSORBANCE, REACTION 4

Sample No.	Time (Min)	Wavelength 600 nm	Wavelength 500 nm	Wavelength 400 nm	Wavelength 350 nm	Wavelength 300 nm
10	50	2.6	4.6	9.8	15.7	25.6
11	55	3.7	6.5	14.4	23.0	38.1
13	65	8.1	15.0	31.3	48.4	81.5
14	70	11.1	19.9	42.3	66.2	106.0
15	75	14.3	25.8	53.7	83.0	134.0
16	80	15.6	29.0	59.5	90.6	149.2
17	85	18.5	33.4	68.9	105.7	173.0
18	90	17.6	32.3	67.1	103.4	167.3
19	95	20.6	37.4	75.6	114.2	187.0
20	100	28.8	52.4	105.1	158.4	252.2
21	105	28.8	52.4	105.1	158.4	252.2
22	110	31.2	55.6	110.1	166.3	261.6
23	115	39.4	69.5	140.4	211.8	---*
24	120	36.6	64.9	131.2	197.5	---*
25	125	42.0	73.0	143.2	214.0	---*
26	130	44.8	78.7	153.9	229.0	---*
27	135	48.0	85.5	166.2	248.8	---*
28	140	60.6	105.0	208.0	---*	---*
29	145	55.6	96.6	191.0	---*	---*

TABLE: B4 (continued)

Sample No.	Time (Min)	Wavelength 600 nm	Wavelength 500 nm	Wavelength 400 nm	Wavelength 350 nm	Wavelength 300 nm
30	150	57.0	99.0	191.0	---*	---*
31	155	58.4	102.0	198.7	---*	---*
32	160	58.4	101.2	194.9	---*	---*

*Sample is above the measurement limit of spectrophotometer (a more diluted sample is required to measure absorbance)

Fig B4 EQUIVALENT ABSORBANCE REACTION 4

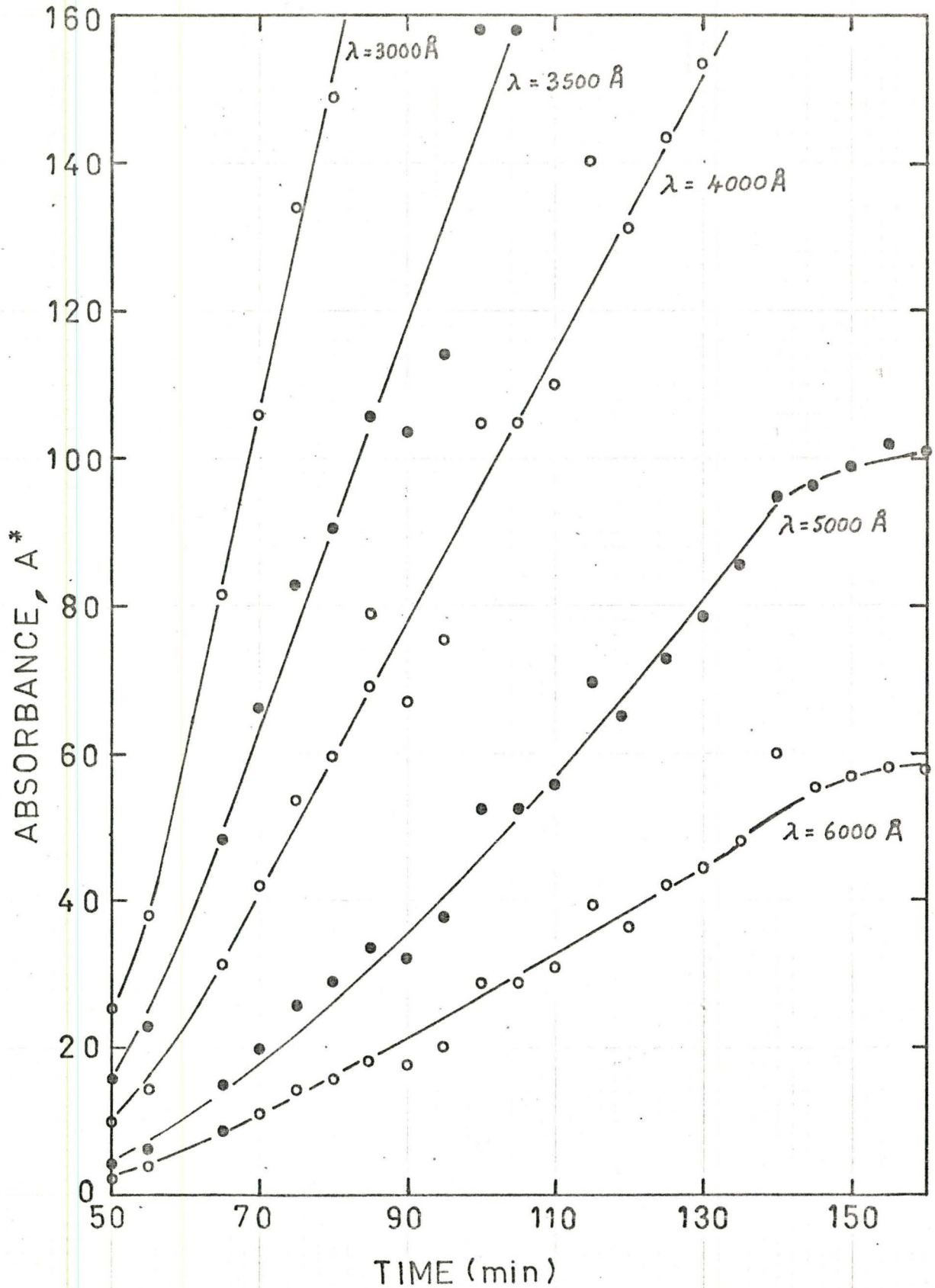


TABLE: B5 EQUIVALENT ABSORBANCE, REACTION 5

Sample No.	Time (Min)	Wavelength 600 nm	Wavelength 500 nm	Wavelength 400 nm	Wavelength 350 nm	Wavelength 300 nm
8	40	0	0	0	0	0
9	45	.6	.6	.8	1.2	1.7
10	50	1.2	2.0	3.9	6.4	10.8
11	55	3.0	4.6	9.6	15.8	26.3
12	60	3.9	7.3	15.6	25.6	42.3
14	70	8.5	15.6	34.2	53.8	87.0
15	75	12.4	24.9	49.6	77.1	123.4
16	80	15.6	29.0	59.9	91.9	146.9
19	95	25.2	46.8	93.5	142.8	224.2
20	100	28.9	56.2	111.7	168.1	259.6
21	105	35.2	64.0	127.1	191.8	294.0
22	110	41.0	73.4	145.2	217.6	---
23	115	38.2	68.9	136.1	205.0	316.0
24	120	39.0	70.1	140.2	212.0	332.0
25	125	44.7	82.0	162.4	246.0	380.0
26	130	41.9	74.6	149.0	225.0	351.8
27	135	44.7	80.4	156.2	233.8	380.0
28	140	46.7	86.6	170.9	257.0	401.0
29	145	45.9	85.8	169.0	253.0	395.5
30	150	44.7	81.6	158.4	238.4	374.4

Fig. B5 EQUIVALENT ABSORBANCE REACTION 5

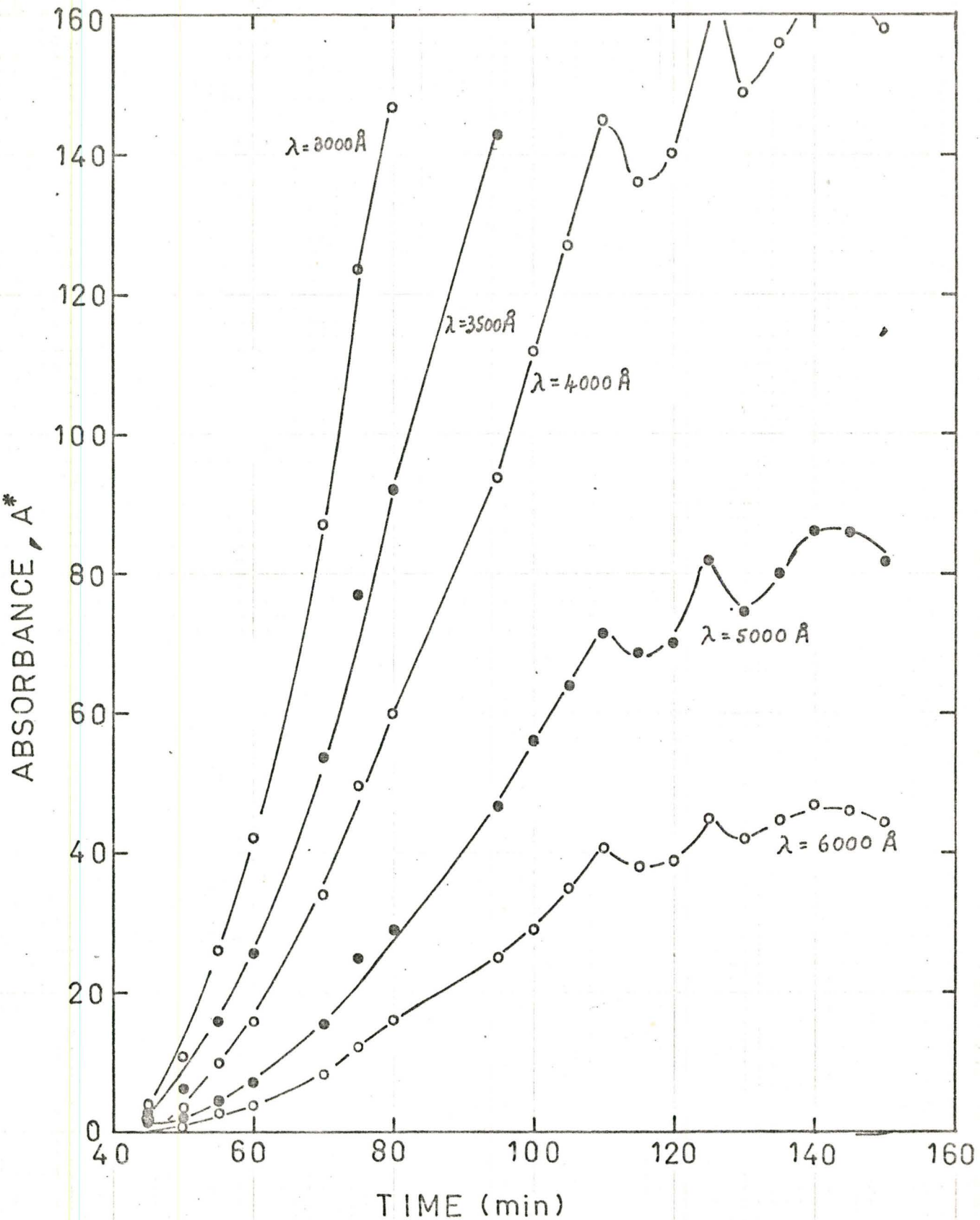


TABLE: B6 EQUIVALENT ABSORBANCE, REACTION 7

Sample No.	Time (Min)	Wavelength 600 nm	Wavelength 500 nm	Wavelength 400 nm	Wavelength 350 nm	Wavelength 300 nm
1	5	0	0	0	0	0
2	10	.9	.6	.9	.9	1.4
3	15	.9	.6	.9	.9	1.4
4	20	.6	.3	.6	.6	.9
5	25	2.2	2.5	3.2	4.1	6.0
6	30	3.5	5.7	10.4	15.8	25.4
7	35	4.1	8.9	20.6	34.2	61.1
8	40	8.9	18.0	41.2	66.5	115.7
10	50	14.2	29.2	65.2	103.0	175.1
11	55	15.8	31.0	68.1	107.2	181.3
12	60	16.8	32.6	71.5	113.0	189.5
13	65	16.5	32.0	70.0	110.2	184.5
14	70	16.5	32.3	70.9	111.0	186.9
15	75	16.5	32.3	70.9	110.8	186.0
16	80	16.5	32.3	70.9	111.0	186.0
17	85	17.4	33.5	72.4	111.8	182.2
18	90	17.5	33.6	72.4	112.0	182.2

Fig.B6

EQUIVALENT ABSORBANCE REACTION 7

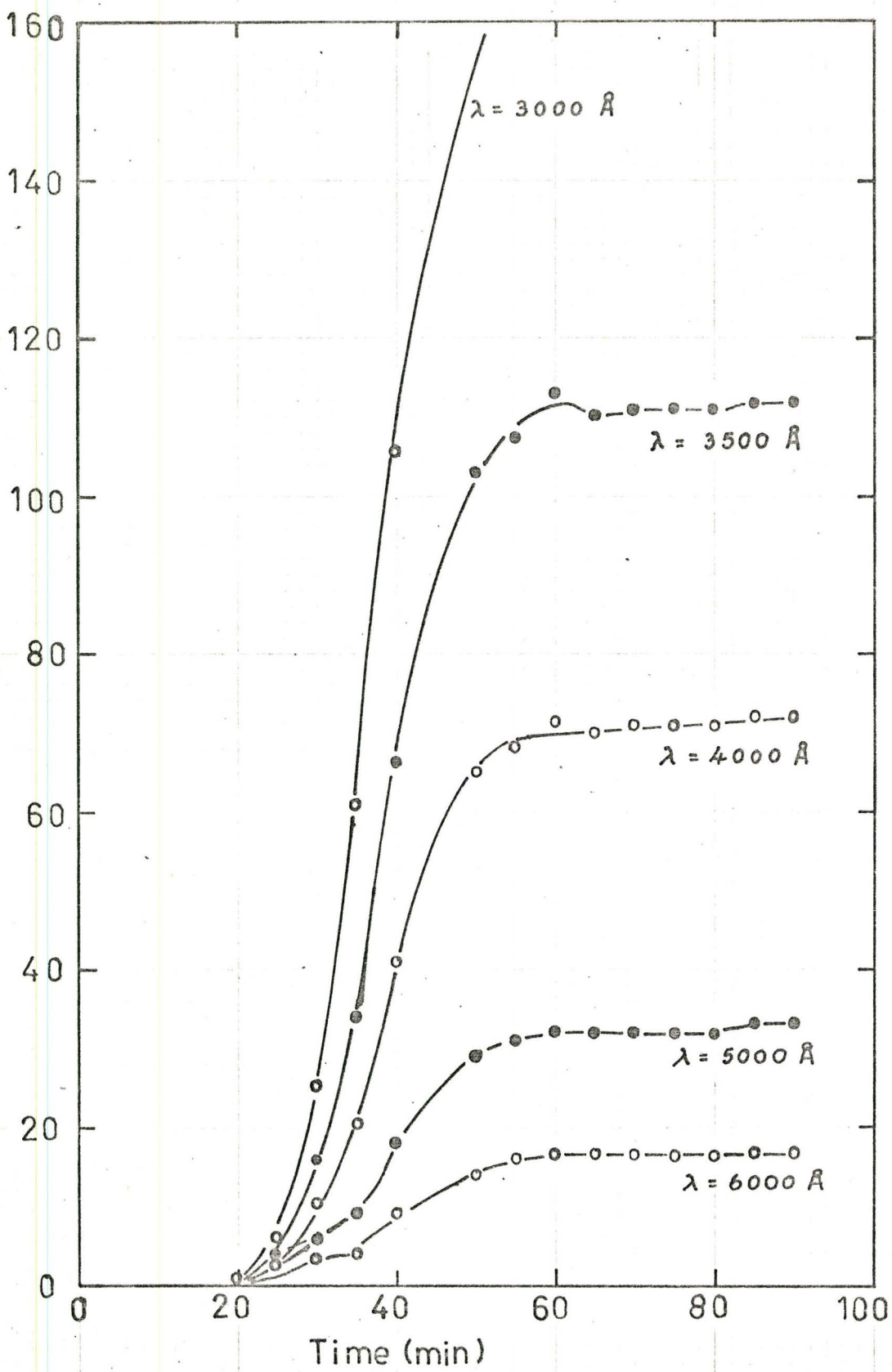
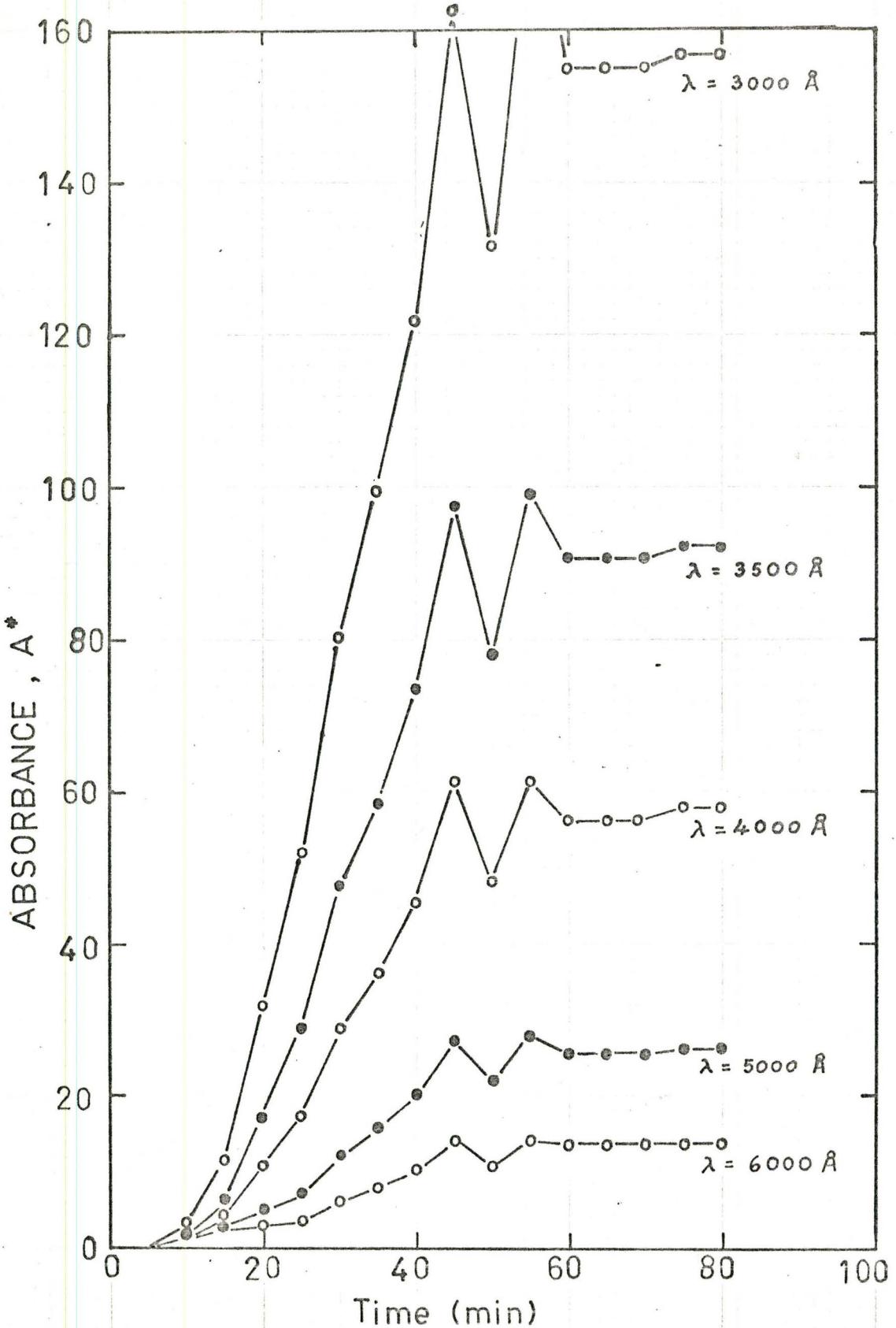


TABLE: B7 EQUIVALENT ABSORBANCE, REACTION 8

Sample No.	Time (Min)	Wavelength 600 nm	Wavelength 500 nm	Wavelength 400 nm	Wavelength 350 nm	Wavelength 300 nm
1	5	0	0	0	0	0
2	10	1.5	1.7	2.2	2.4	3.4
3	15	2.4	3.1	4.8	6.6	11.4
4	20	3.1	4.9	10.9	17.3	31.8
5	25	3.6	7.1	17.3	29.0	52.0
6	30	6.1	12.4	28.9	47.5	80.5
7	35	7.8	15.5	36.0	58.3	99.1
8	40	10.4	20.2	45.4	73.4	122.2
9	45	13.8	27.0	61.5	97.5	162.9
10	50	10.7	21.9	48.0	78.0	131.5
11	55	13.8	27.7	61.5	99.1	181.0
12	60	13.3	25.3	56.5	90.7	155.2
13	65	13.3	25.3	56.5	90.7	155.2
14	70	13.3	25.3	56.5	90.7	155.2
15	75	13.4	26.2	58.1	92.4	157.0
16	80	13.4	26.2	58.1	92.4	157.0

Fig. B7 EQUIVALENT ABSORBANCE REACTION 8



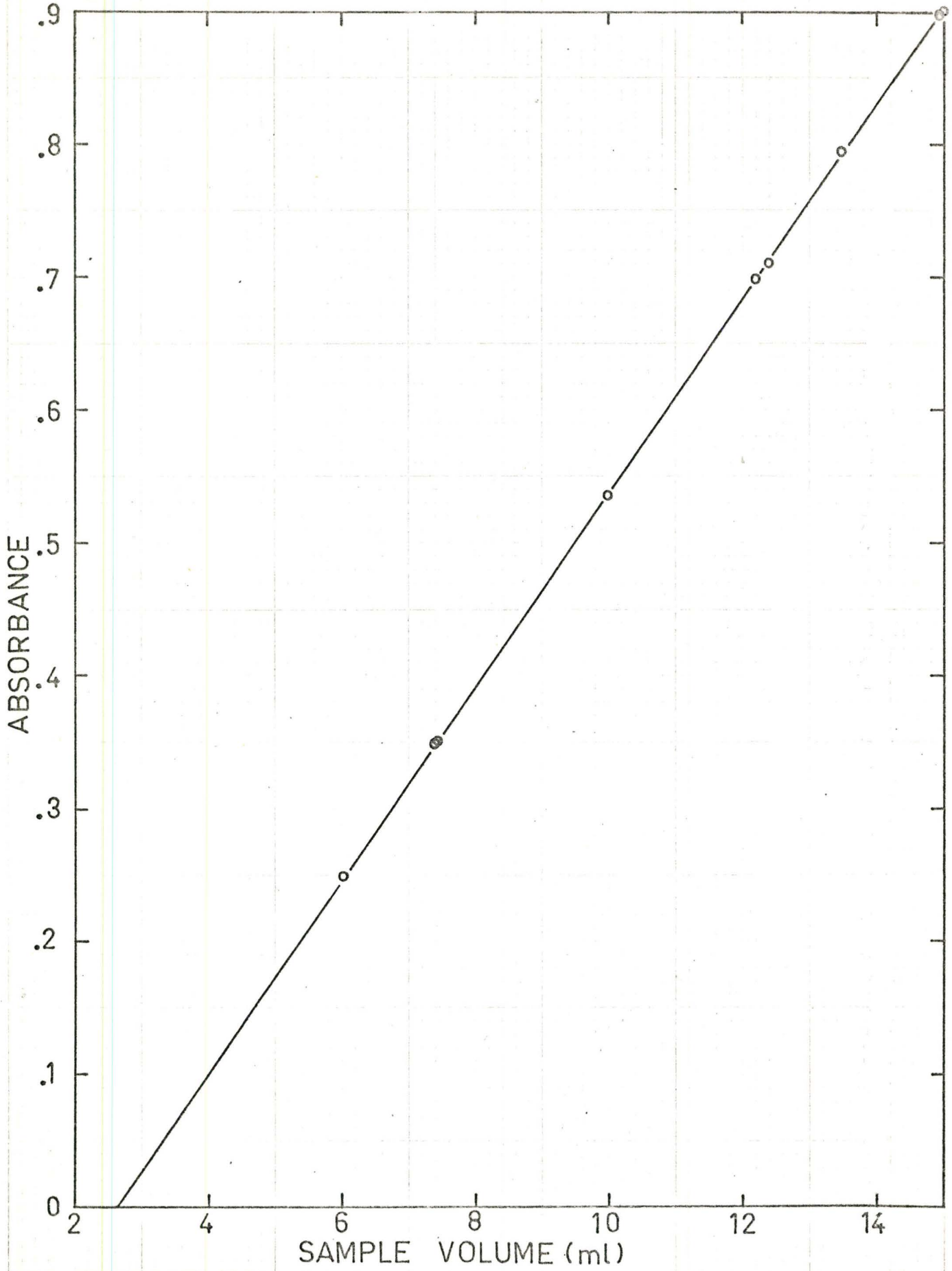
APPENDIX C SAMPLE PREPARATION

Upon final development of the sample system, an accurate controlled volume of latex solution could be diluted for measurement of absorbance using the computer. However, it was necessary to calibrate the system to obtain the correct volumes of latex and dilution vessel. To this end, samples were prepared using different withdrawal times of the syringe, using the same latex solution and measuring the absorbance of the diluted solution at a single wavelength (400 nm). The results of this experiment are tabulated in table C1 and displayed graphically in figure C1.

Table C1: ABSORBANCE vs LATEX CONCENTRATION

Syringe withdrawal time (secs)	volume withdrawn (ml)	measured absorbance
24.8	12.4	.710
24.4	12.2	.698
30.0	15.0	.900
14.9	7.45	.350
14.8	7.4	.348
29.85	14.92	.897
19.95	9.98	.534
27.0	13.5	.795
12.1	6.05	.248
20.0	10.0	.535

Fig C1 ABSORBANCE v LATEX CONC.



The important result from analysis of the data is that the absorbance has a linear relationship with latex concentration. This demonstrates that secondary scattering has no significant effect on the absorbance for the latex solutions used in this study. This is a requirement for determining particle size distributions by the Mie theory.

The linear relationship between absorbance and concentration also allows the use of the following equation to determine the effective volume of the dilution vessel and sample volume.

$$A = \frac{k}{VT} (V - Vd)$$

where A is the measured absorbance

k is a constant depending on the particle size distribution and number of particles in undiluted latex sample.

VT is the effective volume of the dilution vessel (total diluted volume)

V is volume withdrawn by syringe

Vd is volume of latex sample that remains in sample line and is not diluted (dead volume)

Using least squares linear regression, the dead volume was calculated to be 2.65 ml and the volume of the dilution vessel was 570 ml.

APPENDIX D COMPUTER PROGRAMS

In this section, the entire user part of the computer software system is presented. The programs are given in the following order,

1. INTER ... Intercommunication with operator (program executive)
2. INPUT ... Interpretation of names input by operator
3. SAMPL ... Operation of sampling system
4. RTDBØ ... Conversion of decimal input on teletype to binary
5. RTBDØ ... Conversion of binary to decimal output on teletype
6. MATHS ... Math routines
7. INFCE ... Routines to handle interface equipment
8. CLOCK ... System clock
9. PINIT ... Initializer of scheduled programs
10. UVANS ... Operation of UV analyzer
11. LOG ... Standard logging routine
12. FEEDS ... Controls feedrates to reactor
13. TCTRL ... Temperature control of reactor

```
01                      .TITL INTER
02                      .ENT START ENTRY JOBS ERROR TMIN BEGIN
03                      .ENT CHNGE FROM TO READ RDOCT STOP SCHED
04                      .EXTN FLAGL SAMPL UVANS LOGST TTOO PINIT TRANS ITEM1
05                      .EXTN CLOCK INPTT INPTP INPTD RTBDO RTDBO
06                      .EXTN .BRK .PTY .FORK .QUIT
07                      .EXTN INIT
08                      .EXTN DEBUG
09                      .NREL
10                ; THIS IS THE EXECUTIVE ROUTINE FOR RDOS
11                ; ALL OPERATOR COMMUNICATION ENTERS THROUGH THIS PROGRAM
12                ; THIS FIRST SECTION DEALS WITH START UP OF SYSTEM
13
14 00000'002401 TMIN:   JMP @.+1
15 00001'177777          INIT
16
17 00002'177777 START:   .PTY              ;SET HIGH PRIORITY FOR THIS PROGRAM
18 00003'000007          7
19 00004'177777          .FORK             ;START UP SYSTEM CLOCK
20 00005'000015          15
21 00006'177777          CLOCK
22 00007'000004'         .FORK             ;INITIALIZE SCHEDULER
23 00010'000040          40
24 00011'177777          PINIT
25 00012'000007'         .FORK             ;START READING A/D CHANNELS
26 00013'000377          377
27 00014'177777          TRANS
28 00015'000012'         .FORK             ;INITIALIZE LOGGING ROUTINE
29 00016'000077          77
30 00017'177777          LOGST
31 00020'000015'         .FORK             ;PREPARE UV SPECTROPHOTOMETER
32 00021'000050          50
33 00022'177777          UVANS
34
35 00023'177777 PAUSE:   .BRK              ;OPERATOR INTERRUPT - THIS SECTION
36 00024'000201          201               ;PROCEEDS UPON ISSUE OF CTRL A
37 00025'102400          SUB 0 0           ;STOP LOGGING ROUTINE
38 00026'042430          STA 0 @POINT+5    ;FROM PRINTING
39 00027'000020'         .FORK
40 00030'000010          10                ;FIND OUT WHAT OPERATOR WANTS
41 00031'000033'         JOBS
42 00032'000771          JMP PAUSE
43
44 00033'006422 JOBS:    JSR @POINT+4      ;PRINT CR AND LF
45 00034'000043'         MESS
46 00035'006415 ENTRY:   JSR @POINT+1      ;GET OPERATOR INSTRUCTION
47 00036'115000          MOV 0 3
48 00037'001400          JMP 0 3           ;GO TO REQUESTED JOB
49
50 00040'006415 ERROR:   JSR @POINT+4      ;PRINT NO IF INPUT NOT ACCEPTABLE
51 00041'000044'         MESS+1
52 00042'000771          JMP JOBS          ;GO BACK FOR MORE INPUT
53
54 00043'000430"MESS:    AA*2
55 00044'000450"         CC*2
56 00045'000434"         BB*2
57 00046'000454"         DD*2
58 00047'000460"         EE*2
59 00050'000464"         FF*2
```

```
01
02
03                      ;       THE FOLLOWING SET OF ROUTINES CARRY OUT THE
04                      ;       OPERATOR INSTRUCTIONS
05
06 00051'177777 POINT:  INPTD
07 00052'177777         INPTP
08 00053'177777         RTBDO
09 00054'177777         RTDBO
10 00055'177777         TTOO
11 00056'177777         FLAGL
12 00057'177777         INPTT
13
14 00060'000000 ADRES:  0
15 00061'000006 C6:     6
16 00062'177772 N6:     -6
17
18                      ;THIS SECTION STARTS CHANGE OF DATA
19
20
21 00063'006766 CHNGE:  JSR @POINT      ;GET DATA LOCATION
22 00064'040774         STA 0 ADRES     ;THIS ADDRESS - STORE IT
23 00065'000750         JMP ENTRY       ;GO FOR NEXT INSTRUCTION
24
25                      ;THE FOLLOWING CAUSES CURRENT DATA REGISTER TO BE PRINTED
26
27 00066'022772 FROM:   LDA 0 @ADRES    ;LOAD VALUE IN ADDRESS
28 00067'024772         LDA 1 C6        ;#SPACES TO PRINT
29 00070'006763         JSR @POINT+2    ;PRINT VALUE IN DECIMAL
30 00071'000744         JMP ENTRY       ;GO FOR NEXT INSTRUCTION
31
32                      ; THIS SECTION COMPLETE CHANGE OF DATA
33
34 00072'006762 TO:     JSR @POINT+3    ;READ VALUE
35 00073'046765         STA 1 @ADRES    ;STORE IT
36 00074'000737         JMP JOBS
37
38                      ; THIS SECTION READS A DATA LOCATION AND PRINTS IT
39
40 00075'006754 READ:   JSR @POINT      ;GET LOCATION
41 00076'115000         MOV 0 3
42 00077'021400         LDA 0 0 3       ;LOAD VALUE
43 00100'024761         LDA 1 C6        ;#SPACES TO PRINT
44 00101'006752         JSR @POINT+2    ;PRINT VALUE IN DECIMAL
45 00102'000773         JMP READ        ;REPEAT ON SAME LINE
46
47                      ; THIS SECTION READS A LOCATION IN OCTAL
48
49 00103'006746 RDOCT:  JSR @POINT      ;GET DATA LOCATION
50 00104'115000         MOV 0 3
51 00105'021400         LDA 0 0 3       ;LOAD VALUE
52 00106'024754         LDA 1 N6        ;INDICATE OCTAL NUMBER
53 00107'006744         JSR @POINT+2    ;PRINT 6 SPACES OCTAL
54 00110'000773         JMP RDOCT       ;REPEAT
55
56                      ;THIS SECTION STOPS A PROGRAM RUNNING
57
58 00111'006746 STOP:   JSR @POINT+6    ;GET PROGRAM
59 00112'040404         STA 0 .+4
```

```
  0003  INTER
01 00113'014403          DSZ .+3
02 00114'000027'         .FORK              ;INITIATE PROGRAM STOP
03 00115'000050          50
04 00116'177777          SAMPL
05 00117'000714          JMP JOBS
06
07               ; THIS SECTION STARTS A TASK PROGRAM
08
09 00120'006737 BEGIN:   JSR @POINT+6       ;GET TASK TO BE STARTED
10 00121'040403          STA 0 .+3
11 00122'000114'         .FORK              ;CREATE JOB
12 00123'000050          50
13 00124'000116'         SAMPL
14 00125'000706          JMP JOBS           ;GO BACK FOR NEXT INSTRUCTION
15
16               ;THIS SECTION SCHEDULES A TASK TO BE RUN AT SOME FUTURE TIME
17
18 00126'006731 SCHED:   JSR @POINT+6       ;INPUT PROGRAM TO BE SCHEDULED
19 00127'040425          STA 0 PROG         ;SAVE PROGRAM ADDRESS
20 00130'006725          JSR @POINT+4       ;OUTPUT
21 00131'000045'         MESS+2             ;"TO START AT"
22 00132'004424          JSR GTTME          ;GET STARTING TIME
23 00133'000000 STRT:    0
24 00134'006721          JSR @POINT+4       ;OUTPUT
25 00135'000050'         MESS+5             ;"AND STOP AT"
26 00136'004420          JSR GTTME          ;GET STOPPING TIME
27 00137'000000 STP:     0
28 00140'034415          LDA 3 ITEMN        ;LOAD ADDRESS OF NEXT ITEM
29 00141'020413          LDA 0 PROG         ;LOAD SCHEDULED PROGRAM
30 00142'041400          STA 0 0 3          ;STORE IT FOR SCHEDULER
31 00143'175400          INC 3 3
32 00144'020767          LDA 0 STRT         ;LOAD WHEN TO START PROGRAM
33 00145'041400          STA 0 0 3          ;STORE IT
34 00146'175400          INC 3 3
35 00147'020770          LDA 0 STP          ;LOAD WHEN TO STOP PROGRAM
36 00150'041400          STA 0 0 3          ;STORE IT
37 00151'175400          INC 3 3            ;PREPARE ADDRESS INDICATOR FOR NEXT I
38 00152'054403          STA 3 ITEMN
39 00153'000660          JMP JOBS           ;RETURN FOR NEXT INSTRUCTION
40
41 00154'000124'PROG:    SAMPL
42 00155'177777 ITEMN:   ITEM1
43
44 00156'054435 GTTME:   STA 3 AC3
45 00157'006675          JSR @POINT+3       ;GET TIME IN MINUTES
46 00160'125113          MOVL# 1 1 SNC      ;CHECK SIGN
47 00161'000403          JMP .+3            ;SIGN POSITIVE - SKIP
48 00162'124400          NEG 1 1            ;MAKE POSITIVE
49 00163'044426          STA 1 SIGN         ;INDICATE SIGN ACTUALLY NEGATIVE
50 00164'127120          ADDZL 1 1          ;N*4
51 00165'121000          MOV 1 0            ;AC0=N*4
52 00166'127120          ADDZL 1 1          ;N*16.
53 00167'127120          ADDZL 1 1          ;N*64.
54 00170'106400          SUB 0 1            ;N*60.
55 00171'044421          STA 1 TEMP         ;STORE NUMBER IN SECONDS
56 00172'006663          JSR @POINT+4       ;OUTPUT
57 00173'000046'         MESS+3             ;"MIN"
58 00174'006660          JSR @POINT+3       ;GET SECONDS PART OF TIME
59 00175'020415          LDA 0 TEMP         ;FIND MINUTE PART
```

```
  0004  INTER
01 00176'107000          ADD 0 1          ;GET CORRECT TIME
02 00177'020412          LDA 0 SIGN       ;CHECK SIGN
03 00200'101004          MOV 0 0 SZR
04 00201'124400          NEG 1 1
05 00202'046411          STA 1 @AC3
06 00203'102400          SUB 0 0          ;MAKE SURE POSITIVE FOR NEXT ENTRY
07 00204'040405          STA 0 SIGN
08 00205'006650          JSR @POINT+4     ;OUTPUT
09 00206'000047'         MESS+4           ;"SEC"
10 00207'010404          ISZ AC3
11 00210'002403          JMP @AC3         ;RETURN
12
13 00211'000000 SIGN:    0
14 00212'000000 TEMP:    0
15 00213'000000 AC3:     0
16
17                AA:    .TXT @<15><12><40>@
   00214'005015
   00215'000040
18                BB:    .TXT @TO START AT@
   00216'047524
   00217'051440
   00220'040524
   00221'052122
   00222'040440
   00223'000124
19                CC:    .TXT @ NO@
   00224'047040
   00225'000117
20                DD:    .TXT @MIN@
   00226'044515
   00227'000116
21                EE:    .TXT @SEC@
   00230'042523
   00231'000103
22                FF:    .TXT @ AND STOP AT@
   00232'040440
   00233'042116
   00234'051440
   00235'047524
   00236'020120
   00237'052101
   00240'000000
23
24        000000'        .END TMIN
```

```
  0001  INPUT
01                      .TITL INPUT
02                      .ENT INPTP INPTD INPTT
03                      .EXTN .IOX .QUIT
04                      .EXTN MONMR INITR EMULS RDOCT ATOD ATOD1
05                      .EXTN SAMPL LOG CHNGE TO FROM READ ERROR
06                      .EXTN TCTRL TSP TCINT BEGIN TYPEC
07                      .EXTN HOUR MINIT SECND SMINT LGINT SMVOL STOP
08                      .EXTN SCHED RTIME ITEM1 I1SRT I1STP
09                      .EXTN ITEM2 I2SRT I2STP ITEM3 I3SRT I3STP
10                      .EXTN ITEM4 I4SRT I4STP ITEM5 I5SRT I5STP
11                      .EXTN ITEM6 I6SRT I6STP ITEM7 I7SRT I7STP
12                      .EXTN MAXMF MAXIF MAXEF MFLOW IFLOW EFLOW
13                      .NREL
14
15                  ;PROGRAM GETS A STRING OF LETTERS FROM TELETYPE
16                  ;   CHECKS CODE OF STRING FOR ACCEPTABLE ADDRESS
17                  ;   IF ACCEPTABLE - RETURNS WITH AC0 CONTAINING ADDRESS
18                  ;   IF NOT - ERROR MESSAGE IS PRINTED
19
20 00000'020463 INPTT:  LDA 0 LISTT     ;ENTRY POINT FOR TASK INPUT
21 00001'040461         STA 0 LIST
22 00002'020555         LDA 0 TABLT
23 00003'040553         STA 0 TABLE
24 00004'000412         JMP INPUT
25
26 00005'020465 INPTP:  LDA 0 LISTP     ;ENTRY POINT FOR PROGAM INPUT
27 00006'040454         STA 0 LIST
28 00007'020556         LDA 0 TABLP
29 00010'040546         STA 0 TABLE
30 00011'000405         JMP INPUT
31
32 00012'020473 INPTD:  LDA 0 LISTD     ;ENTRY POINT FOR DATA ADRESS INPUT
33 00013'040447         STA 0 LIST
34 00014'020563         LDA 0 TABLD
35 00015'040541         STA 0 TABLE
36
37 00016'054437 INPUT:  STA 3 AC3       ;SAVE RETURN ADDRESS
38 00017'152400         SUB 2 2         ;CLEAR AC2 FOR CODING INPUT
39 00020'177777         .IOX
40 00021'000000         0               ;TELETYPE
41 00022'020000         020000          ;INPUT AND ECHO
42 00023'000054'        CHARR           ;ADDRESS FOR INPUT
43 00024'000001         1               ;READ 1 CHARACTER
44 00025'177777         ERROR           ;ERROR RETURN ADDRESS
45 00026'020426         LDA 0 CHARR     ;LOAD INPUT CHARACTER
46 00027'024430         LDA 1 C101      ;LOAD ASCII "A"
47 00030'034430         LDA 3 C132      ;LOAD ASCII "Z"
48 00031'106033         ADCZ# 0 1 SNC   ;CHECK IF INPUT
49 00032'116033         ADCZ# 0 3 SNC   ;IS A CHARACTER
50 00033'000404         JMP CHECK       ;NO - END OF INPUT
51 00034'151120         MOVZL 2 2       ;YES DO CODE
52 00035'113000         ADD 0 2         ;COMPLETE CODE
53 00036'000762         JMP INPUT+2     ;GO BACK FOR NEXT CHARACTER
54 00037'034423 CHECK:  LDA 3 LIST      ;GET ADDRESS OF LIST OF ACCEPTABLE AD
55 00040'126400         SUB 1 1         ;CLEAR ADDRESS INDICATOR (AC1)
56 00041'175400         INC 3 3         ;GET ADDRESS
57 00042'021400         LDA 0 0 3       ;GET CODE
58 00043'101005         MOV 0 0 SNR     ;END OF LIST ?
59 00044'002415         JMP @POINT      ;YES AND CODE NOT ACCEPTABLE
```

```
  0002  INPUT
01 00045'125400            INC 1 1          ;NO - INCREMENT ADDRESS LOCATION INDIC
02 00046'112414            SUB# 0 2 SZR     ;IS THIS THE CODE ?
03 00047'000772            JMP CHECK+2      ;NO - GO BACK TO CHECK NEXT CODE
04 00050'034506            LDA 3 TABLE      ;GET LOCATION OF TABLE OF ADDRESSES
05 00051'137000            ADD 1 3          ;GET ADDRESS OF ADDRESS
06 00052'021400            LDA 0 0 3        ;LOAD ADDRESS INDICATED BY CODE
07 00053'002402            JMP @AC3         ;RETURN TO CALLING PROGRAM
08
09 00054'000000 CHARR:     0
10 00055'000000 AC3:       0
11 00056'000177 MASK:      177
12 00057'000101 C101:      101
13 00060'000132 C132:      132
14 00061'000025'POINT:     ERROR
15 00062'000062'LIST:      LIST
16
17 00063'000063'LISTT:     LISTT
18 00064'011365            11365            ;SAMPLE
19 00065'023210            23210            ;MONOMER
20 00066'112630            112630           ;INITIATOR
21 00067'023640            23640            ;EMULSIFIER
22 00070'044370            44370            ;CONTROLT
23 00071'000000            0
24
25 00072'000072'LISTP:     LISTP
26 00073'001025            1025             ;LOG
27 00074'010363            10363            ;CHANGE
28 00075'002143            2143             ;FROM
29 00076'000367            367              ;TO
30 00077'002152            2152             ;READ
31 00100'002326            2326             ;STOP
32 00101'045741            45741            ;SCHEDULE
33 00102'022546            22546            ;READOCT
34 00103'004714            4714             ;START
35 00104'000000            0
36
37 00105'000105'LISTD:     LISTD
38 00106'114322            114322           ;TYPECONTROL
39 00107'002170            2170             ;HOUR
40 00110'011341            11341            ;MINUTE
41 00111'011344            11344            ;SECOND
42 00112'023724            23724            ;SMPLINT
43 00113'024002            24002            ;SMPLVOL
44 00114'011274            11274            ;LOGINT
45 00115'011556            11556            ;TEMPSP
46 00116'047014            47014            ;TEMPCINT
47 00117'045463            45463            ;REACTIME
48 00120'002072            2072             ;ATOD
49 00121'004266            4266             ;ATODB
50 00122'045267            45267            ;MAXFLOWN
51 00123'045263            45263            ;MAXFLOWI
52 00124'045257            45257            ;MAXFLOWE
53 00125'004367            4367             ;FLOWN
54 00126'004363            4363             ;FLOWI
55 00127'004357            4357             ;FLOWE
56 00130'004437            4437             ;ITEMA
57 00131'011731            11731            ;STARTA
58 00132'004755            4755             ;STOPA
59 00133'004440            4440             ;ITEMB
```

```
  0003  INPUT
01 00134'011732            11732              ;STARTB
02 00135'004756            4756               ;STOPB
03 00136'004441            4441               ;ITEMC
04 00137'011733            11733              ;STARTC
05 00140'004757            4757               ;STOPC
06 00141'004442            4442               ;ITEMD
07 00142'011734            11734              ;STARTD
08 00143'004760            4760               ;STOPD
09 00144'004443            4443               ;ITEME
10 00145'011735            11735              ;STARTE
11 00146'004761            4761               ;STOPE
12 00147'004444            4444               ;ITEMF
13 00150'011736            11736              ;STARTF
14 00151'004762            4762               ;STOPF
15 00152'004445            4445               ;ITEMG
16 00153'011737            11737              ;STARTG
17 00154'004763            4763               ;STOPG
18 00155'000000            0
19
20 00156'000156'TABLE:     TABLE
21
22 00157'000157'TABLT:     TABLT
23 00160'177777            SAMPL
24 00161'177777            MONMR
25 00162'177777            INITR
26 00163'177777            EMULS
27 00164'177777            TCTRL
28
29 00165'000165'TABLP:     TABLP
30 00166'177777            LOG                ;ADDRESS OF LOG
31 00167'177777            CHNGE              ;ADDRESS OF CHNGE
32 00170'177777            FROM
33 00171'177777            TO
34 00172'177777            READ
35 00173'177777            STOP
36 00174'177777            SCHED
37 00175'177777            RDOCT
38 00176'177777            BEGIN
39
40 00177'000177'TABLD:     TABLD
41 00200'177777            TYPEC
42 00201'177777            HOUR
43 00202'177777            MINIT
44 00203'177777            SECND
45 00204'177777            SMINT
46 00205'177777            SMVOL
47 00206'177777            LGINT
48 00207'177777            TSP
49 00210'177777            TCINT
50 00211'177777            RTIME
51 00212'177777            ATOD
52 00213'177777            ATOD1
53 00214'177777            MAXMF
54 00215'177777            MAXIF
55 00216'177777            MAXEF
56 00217'177777            MFLOW
57 00220'177777            IFLOW
58 00221'177777            EFLOW
59 00222'177777            ITEM1
```

```
  0004   INPUT
01 00223'177777          I1SRT
02 00224'177777          I1STP
03 00225'177777          ITEM2
04 00226'177777          I2SRT
05 00227'177777          I2STP
06 00230'177777          ITEM3
07 00231'177777          I3SRT
08 00232'177777          I3STP
09 00233'177777          ITEM4
10 00234'177777          I4SRT
11 00235'177777          I4STP
12 00236'177777          ITEM5
13 00237'177777          I5SRT
14 00240'177777          I5STP
15 00241'177777          ITEM6
16 00242'177777          I6SRT
17 00243'177777          I6STP
18 00244'177777          ITEM7
19 00245'177777          I7SRT
20 00246'177777          I7STP
21
22                       .END
```

```
                        .TITL SAMPL
                        .ENT SAMPL SMINT SMVOL SMIN SSEC
                        .EXTN  .PTY .FORK .WAIT RTIME .XMIT .QUIT TTOO
                        .EXTN RELYS
                        .NREL

                    ; THIS PROGRAM HANDLES ALL THE VALVES, INFUSION PUMP, AND
                    ; STIRRER FOR THE SAMPLING DILUTION SYSTEM.
                    ;
                    ; THIS IS A TASK PROGRAM
                    ;  - USES .XMIT OVER CHANNEL 3 FOR UV ANALYSER ROUTINE

00000'000454 SMINT:   300.                ;SAMPLING INTERVAL(SECS)
00001'000003 PURGE:   3.                  ;SAMPLE LINE PURGE TIME
00002'000005 STINT:   5.                  ;STIRRING INTERVAL
00003'000017 EMINT:   15.                 ;TIME REQUIRED TO EMPTY
00004'000024 SMVOL:   20.                 ;SAMPLE VOLUME
00005'000010 SLFSH:   10                  ;SAMPLE LINE FLUSH TIME
00006'000012 FSHTM:   10.                 ;FLUSHING TIME FOR DILURION VESSEL
00007'000031 UVINT:   25.                 ;DELAY IN OPENING VALVE
00010'000043 FILLR:   35.                 ;MINIMUM TIME TO FILL VESSEL
00011'000170 MINTM:   120.                ;TOTAL TIME TO COMPLETE A SAMPLE
00012'000055 MAXST:   45.                 ;MAXIMUM SYRINGE WITHDRAWAL TIME
                    ;
                    ; RELAY 1 = WATER VALVE IN
                    ; RELAY 2 = VALVE TO DRAIN
                    ; RELAY 3 = OVERFLOW VALVE
                    ; RELAY 4 = POWER TO SYRINGE MACHINE
                    ; RELAY 5 = 0 - WITHDRAW SYRINGE  1 - INFUSE SYRINGE
                    ; RELAY 6 = SAMPLE INLET VALVE TO DILUTER
                    ; RELAY 7 = VALVE TO UV ANALYSER
                    ; RELAY 8 = STIRRER CONTROL
                    ; RELAY 9 = PURGE VALVE
                    ; RELAY 10= REACTOR VALVE
                    ;
00013'012500 RELAY:   012500              ;TABLE OF RELAYS
00014'000400          000400              ;FOR SWITCHING VALVES
00015'021000          021000
00016'061000          061000
00017'060000          060000
00020'140000          140000
00021'134000          134000
00022'102200          102200
00023'000700          ,000700
00024'120000          120000

00025'177777 POINT:   TTOO
00026'177777          RTIME
00027'000000 SMIN:    0
00030'000000 SSEC:    0
00031'000000 DRAW:    0
00032'000767 STATUS:  JMP .-9.

00033'000535 STOPR:   JMP END

                    ;
                    ;  THE FOLLOWING SECTION PREPARES FOR SAMPLING
                    ;
00034'020776 SAMPL:   LDA 0 STATUS      ;THIS INSTRUCTION ALLOWS PROGRAM TO R
```

```
  0002  SAMPL
01 00035'040414          STA 0 PLACE
02 00036'177777          .PTY              ;SET PRIORITY
03 00037'000100          100
04                ;
05                ;   SAMPLE CONTROL
06                ;
07 00040'177777 FORK:    .FORK             ;TAKE SAMPLE
08 00041'000101          101
09 00042'000052'         DO
10 00043'020735          LDA 0 SMINT       ;WAIT FOR SPECIFIED TIME INTERVAL
11 00044'100400          NEG 0 0
12 00045'177777          .WAIT
13 00046'000144          100.
14 00047'101404          INC 0 0 SZR
15 00050'000775          JMP .-3
16 00051'000767 PLACE:   JMP .-9.
17                ;
18                ;       THIS SECTION TAKES A SAMPLE
19                ;
20 00052'020726 DO:      LDA 0 SMINT       ;CHECK SAMPLE INTERVAL
21 00053'024731          LDA 1 SMVOL
22 00054'125120          MOVZL 1 1         ;BOTH INFUSE AND WITHDRAW
23 00055'030734          LDA 2 MINTM       ;LOAD MINIMUM SAMPLE TIME
24 00056'147000          ADD 2 1           ;COMPUTE TOTAL TIME TO SAMPLE
25 00057'106513          SUBL# 0 1 SNC     ;ENOUGH TIME TO SAMPLE ?
26 00060'000506          JMP SEROR         ;NO - PRINT ERROR AND QUIT
27 00061'024723          LDA 1 SMVOL       ;LOAD SAMPLE VOLUME (*2.)
28 00062'044747          STA 1 DRAW        ;SAVE VALUE FOR LATER ON
29 00063'030727          LDA 2 MAXST       ;LOAD MAXIMUM SYRINGE TIME
30 00064'146513          SUBL# 2 1 SNC     ;VALUE TOO BIG ?
31 00065'000501          JMP SEROR         ;YES - PRINT ERROR AND QUIT
32 00066'024713          LDA 1 PURGE       ;LOAD SAMPLE PURGE TIME
33 00067'020734          LDA 0 RELAY+8.
34 00070'004450          JSR RESET
35 00071'022735          LDA 0 @POINT+1
36 00072'040736          STA 0 SSEC
37 00073'020720          LDA 0 RELAY       ;LOAD RELAYS TO TAKE SAMPLE
38 00074'024735          LDA 1 DRAW
39 00075'004443          JSR RESET
40 00076'020716          LDA 0 RELAY+1     ;STIR ONLY
41 00077'024703          LDA 1 STINT
42 00100'004440          JSR RESET
43 00101'020714          LDA 0 RELAY+2     ;SEND SAMPLE TO UVANS
44 00102'024705          LDA 1 UVINT
45 00103'177777          .XMIT
46 00104'000003          3                 ;TELL ANALYSER ROUTINE SAMPLE COMING
47 00105'004433          JSR RESET
48 00106'020710          LDA 0 RELAY+3     ;CONTINUE EMPTYING VESSEL
49 00107'024700          LDA 1 UVINT
50 00110'004430          JSR RESET
51 00111'020706          LDA 0 RELAY+4     ;COMPLETE EMPTYING
52 00112'024671          LDA 1 EMINT
53 00113'004425          JSR RESET
54 00114'020704          LDA 0 RELAY+5     ;FLUSH VESSEL
55 00115'024671          LDA 1 FSHTM
56 00116'004422          JSR RESET
57 00117'020702          LDA 0 RELAY+6     ;FILL WITH WATER & EMPTY SYRINGE
58 00120'024711          LDA 1 DRAW
59 00121'004417          JSR RESET
```

```
  0003  SAMPL
01 00122'020702          LDA 0 RELAY+9.   ;COMPLETE FILLING IF NOT YET FULL
02 00123'024706          LDA 1 DRAW
03 00124'030664          LDA 2 FILLR
04 00125'146422          SUBZ 2 1 SZC
05 00126'000403          JMP .+3
06 00127'124400          NEG 1 1
07 00130'004410          JSR RESET
08 00131'020671          LDA 0 RELAY+7    ;FLUSH SAMPLE LINE
09 00132'024653          LDA 1 SLFSH
10 00133'004405          JSR RESET
11 00134'102400          SUB 0 0          ;CLOSE ALL VALVES
12 00135'126520          SUBZL 1 1
13 00136'004402          JSR RESET
14 00137'177777          .QUIT
15                ;
16                ;   SUBROUTINE INDICATES RELAYS FOR RELAY PROGRAM
17
18 00140'130405 RESET:   NEG 1 2 SNR      ;TEST IF NO WAITING
19 00141'001400          JMP 0 3          ;RETURN
20 00142'054412          STA 3 RETRN      ;MUST SAVE RETURN ADDRESS
21 00143'000040'         .FORK            ;SET RELAYS
22 00144'000001          1                ;MUST USE HIGHEST PRIORITY
23 00145'177777          RELYS
24 00146'000045'         .WAIT            ;WAIT ONE SECOND
25 00147'000144          100.
26 00150'151404          INC 2 2 SZR
27 00151'000775          JMP .-3
28 00152'002402          JMP @RETRN       ;RETURN
29
30 00153'000077 SETRL:   77
31 00154'000000 RETRN:   0
32
33                ;ERROR MESSAGE
34
35 00155'000334"AA:      A*2
36                A:     .TXT @SAMPLING ERROR@
   00156'040523
   00157'050115
   00160'044514
   00161'043516
   00162'042440
   00163'051122
   00164'051117
   00165'000000
37 00166'006637 SERROR:  JSR @POINT
38 00167'000155'         AA
39                ;
40                ;       END ROUTINE
41                ;
42 00170'020402 END:     LDA 0 .+2        ;THIS INSTRUCTION WILL STOP SAMPLE RO
43 00171'040660          STA 0 PLACE
44 00172'000137'         .QUIT
45
46                       .END
```

```
01                      .TITL RTDBO
02                      .ENT RTDBO
03                      .EXTN .IOX TTOO
04                      .NREL
05
06                ; PROGRAM RECIEVES UP TO 5 DECIMAL DIGITS PLUS SIGN FROM TELET
07                ;         RETURNS WITH INPUT CONVERTED TO BINARY IN AC1
08                ;         "+" OR "-" ACCEPTED AS SIGN ("+" OPTIONAL)
09                ;         SPACE INDICATES END OF INPUT
10                ;         ANY OTHER CHARACTER CAUSES DELETION OF INPUT
11                ;         AND REQUEST FOR NEW NUMBER
12
13 00000'054454 RTDBO:   STA 3 AC3        ;SAVE RETURN ADDRESS
14 00001'006452           JSR @POINT       ;PRINT WAITING FOR NUMBER MESSAGE.. #
15 00002'000055'          AA
16 00003'102400           SUB 0 0          ;CLEAR AC0
17 00004'040452           STA 0 SIGN       ;ENSURE STARTING SIGN IS "+"
18 00005'040452           STA 0 RESLT      ;CLEAR RESULT REGISTER
19 00006'177777 GETC:     .IOX
20 00007'000000           0                ;TELETYPE
21 00010'020000           020000           ; ECHO ON INPUT - WORD FORMAT
22 00011'000060'          INPUT            ;ADDRESS OF STORAGE
23 00012'000001           1                ;ONE CHARACTER
24 00013'000001'          RTDBO+1          ;ERROR RETURN ADDRESS
25 00014'020444           LDA 0 INPUT      ;LOAD INPUT CHARACTER
26 00015'024444           LDA 1 C60        ;LOAD ASCII "0"
27 00016'034444           LDA 3 C71        ;LOAD ASCII "9"
28 00017'162033           ADCZ# 3 0 SNC    ;TEST IF INPUT CHARACTER
29 00020'106032           ADCZ# 0 1 SZC    ;IS A DIGIT
30 00021'000402           JMP .+2          ;NOT A DIGIT
31 00022'000414           JMP DIGIT
32 00023'024440           LDA 1 PSIGN      ;LOAD ASCII "+"
33 00024'106415           SUB# 0 1 SNR     ;TEST FOR "+"
34 00025'000761           JMP GETC         ;YES - GO BACK FOR NEXT CHARACTER
35 00026'024436           LDA 1 SP         ;LOAD ASCII SPACE
36 00027'106415           SUB# 0 1 SNR     ;TEST FOR SPACE
37 00030'000416           JMP END
38 00031'024434           LDA 1 NSIGN      ;LOAD ASCII "-"
39 00032'106414           SUB# 0 1 SZR     ;TEST FOR "-"
40 00033'000746           JMP RTDBO+1      ;NO
41 00034'040422           STA 0 SIGN       ;SET SIGN INDICATOR
42 00035'000751           JMP GETC         ;GO FOR NEXT CHARACTER
43
44 00036'122400 DIGIT:    SUB 1 0          ;CONVERT INPUT TO BINARY
45 00037'024420           LDA 1 RESLT      ;LOAD PREVIOUS INPUT
46 00040'135000           MOV 1 3          ;MULTIPLY PREVIOUS INPUT BY 10.
47 00041'127120           ADDZL 1 1        ;N*4
48 00042'167120           ADDZL 3 1        ;N*10.
49 00043'123000           ADD 1 0          ;OBTAIN NEW NUMBER
50 00044'040413           STA 0 RESLT      ;STORE NEW NUMBER
51 00045'000741           JMP GETC
52
53 00046'024411 END:      LDA 1 RESLT      ;LOAD ANSWER
54 00047'020407           LDA 0 SIGN       ;LOAD SIGN INDICATOR
55 00050'101004           MOV 0 0 SZR      ;TEST SIGN OF NUMBER
56 00051'124400           NEG 1 1          ;NEGATIVE NUMBER SO CHANGE SIGN
57 00052'002402           JMP @AC3         ;RETURN TO CALLING PROGRAM
58
59 00053'177777 POINT:    TTOO
```

```
  0002  RTDBO
01 00054'000000 AC3:    0
02 00055'000154"AA:     A*2
03 00056'000000 SIGN:   0
04 00057'000000 RESLT:  0
05 00060'000000 INPUT:  0
06 00061'000060 C60:    60
07 00062'000071 C71:    71
08 00063'000053 PSIGN:  53
09 00064'000040 SP:     40
10 00065'000055 NSIGN:  55
11
12              A:      .TXT "# "
   00066'020043
   00067'000000
13
14                      .END
```

```
01                      .TITL RTBDO
02                      .ENT RTBDO
03                      .EXTN .IOX BIDEC
04                      .NREL
05
06                  ; PROGRAM TO OUTPUT A NUMBER ON TELETYPE
07                  ; CALL WITH AC0 - NUMBER TO BE PRINTED
08                  ;           AC1 - NUMBER OF CHARACTERS (OCTAL IF NEGATIVE) (MA
09                  ;           AC2 - NOT USED AND SAVED
10                  ; SPACE PRINTED AT END OF NUMBER
11
12 00000'050511 RTBDO:  STA 2 AC2       ;SAVE AC2
13 00001'054511         STA 3 AC3       ;SAVE RETURN ADDRESS
14 00002'125112         MOVL# 1 1 SZC   ;TEST IF OCTAL NUMBER
15 00003'000516         JMP OCTAL       ;YES GO TO THAT ROUTINE
16 00004'034507         LDA 3 MASK      ;LOAD TO GET # OF CHARACTERS
17 00005'131300         MOVS 1 2        ;MOVE AND SWAP WORD
18 00006'167400         AND 3 1         ;MASK OUT UNWANTED BITS
19 00007'044501         STA 1 NUMBR     ;STORE NUMBER OF CHARACTERS
20 00010'173400         AND 3 2         ;MASK OUT TO GET DECIMAL PLACE
21 00011'050503         STA 2 DECML
22 00012'006466         JSR @POINT+1    ;CONVERT TO DECIMAL
23       000006 RESLT:  .BLK 6          ;SPACES FOR RESULT
24 00021'000040 C40:    40              ;ASCII SPACE
25 00022'034455         LDA 3 POINT     ;GET ADDRESS OF RESULT
26 00023'102400         SUB 0 0         ;CLEAR AC0
27 00024'040463         STA 0 NVAL      ;CLEAR NVAL
28 00025'030774         LDA 2 C40       ;LOAD ASCII SPACE
29 00026'024456         LDA 1 C60       ;LOAD ASCII ZERO
30 00027'021401         LDA 0 1 3       ;LOAD DIGIT
31 00030'106414         SUB# 0 1 SZR    ;TEST IF ASCII ZERO
32 00031'000405         JMP .+5         ;NO - HAVE FIRST NON ZERO DIGIT
33 00032'010455         ISZ NVAL        ;INCREMENT ZERO COUNTER
34 00033'051401         STA 2 1 3       ;STORE SPACE IN DIGIT
35 00034'175400         INC 3 3         ;INCREMENT ADDRESS INDICATOR
36 00035'000772         JMP .-6         ;GO BACK TO TEST NEXT DIGIT
37 00036'024451         LDA 1 NVAL      ;LOAD NUMBER OF LEADING ZEROS
38 00037'020754         LDA 0 RESLT     ;GET SIGN OF NUMBER
39 00040'050753         STA 2 RESLT     ;STORE SPACE IN HERE
40 00041'034442         LDA 3 NSIGN     ;GET ASCII "-"
41 00042'116414         SUB# 0 3 SZR    ;TEST IF NEGATIVE NUMBER
42 00043'000406         JMP .+6         ;NO SKIP TO NEXT TEST
43 00044'034433         LDA 3 POINT     ;GET ADDRESS OF RESULT
44 00045'137000         ADD 1 3         ;GET ADDRESS WHERE SIGN SHOULD BE
45 00046'041400         STA 0 0 3       ;STORE SIGN
46 00047'014440         DSZ NVAL        ;DECREMENT NUMBER OF LEADING ZEROS
47 00050'000401         JMP .+1
48 00051'020435         LDA 0 C6        ;LOAD 6
49 00052'024435         LDA 1 NVAL      ;LOAD # OF UNECESSARY PRINTS
50 00053'125400         INC 1 1         ;GET ACTUAL REDUNDANT PRINTS
51 00054'122404         SUB 1 0 SZR     ;OBTAIN NUMBER OF SPACES REQU'D FOR PR
52 00055'000404         JMP .+4         ;THESE NEXT THREE INSTRUCTIONS
53 00056'030426         LDA 2 C60       ;STORE 0 FOR ALL ZERO NUMBER
54 00057'050741         STA 2 RESLT+5
55 00060'101400         INC 0 0
56 00061'024427         LDA 1 NUMBR     ;GET NUMBER OF SPACE SPECIFIED
57 00062'040425         STA 0 NVAL      ;SAVE NUMBER OF ACTUAL DIGITS
58 00063'125005         MOV 1 1 SNR
59 00064'000433         JMP MIN         ;WANT MINIMUM # OF SPACES PRINTED
```

```
01 00065'122033          ADCZ# 1 0 SNC    ;SKIP IF NUMBR<NVAL
02 00066'000452          JMP OUTPT
03
04 00067'020413          LDA 0 I          ;LOAD ASCII "*"
05 00070'024415          LDA 1 N6         ;LOAD # OF DIGITS
06 00071'034406          LDA 3 POINT      ;LOAD ADDRES OF RESULT
07 00072'041400          STA 0 0 3        ;STORE *'S TO INDICATE OVERFLOW
08 00073'175400          INC 3 3          ;INCREMENT ADDRESS INDICATOR
09 00074'125404          INC 1 1 SZR      ;TEST COMPLETION
10 00075'000775          JMP .-3          ;GO BACK FOR NEXT STORE
11 00076'000442          JMP OUTPT
12
13              ; DATA FOR RTBDO
14
15 00077'000013'POINT:   RESLT
16 00100'177777          BIDEC
17 00101'000021'         RESLT+6
18 00102'000052 I:       052
19 00103'000055 NSIGN:   55
20 00104'000060 C60:     60
21 00105'177772 N6:      -6
22 00106'000006 C6:      6
23 00107'000000 NVAL:    0
24 00110'000006 NUMBR:   6
25 00111'000000 AC2:     0
26 00112'000000 AC3:     0
27 00113'000007 MASK:    7
28 00114'000000 DECML:   0
29 00115'000116'DOTAD:   DOTAD+1
30 00116'000056          056
31
32 00117'040771 MIN:     STA 0 NUMBR
33 00120'000420          JMP OUTPT
34
35 00121'024765 OCTAL:   LDA 1 C6         ;ALTER TO POSITIVE NUMBER
36 00122'044766          STA 1 NUMBR
37 00123'030762          LDA 2 N6         ;LOAD COUNTER
38 00124'024767          LDA 1 MASK       ;LOAD OCTAL MASK
39 00125'107400          AND 0 1          ;GET OCTAL DIGIT
40 00126'034756          LDA 3 C60        ;GET ASCII CONVERSION NUMBER
41 00127'167000          ADD 3 1          ;CONVERT OCTAL DIGIT TO ASCII
42 00130'034747          LDA 3 POINT      ;GET RESULT ADDRESS
43 00131'156400          SUB 2 3          ;GET CORRECT POSITION
44 00132'045777          STA 1 -1 3       ;STORE IN CORRECT DIGIT REGISTER
45 00133'101220          MOVZR 0 0        ;MOVE INPUT TO CORRECT POSITION
46 00134'101220          MOVZR 0 0        ;FOR GETTING NEXT DIGIT
47 00135'101220          MOVZR 0 0
48 00136'151404          INC 2 2 SZR      ;TEST IF FINISHED
49 00137'000765          JMP OCTAL+3
50
51 00140'020754 OUTPT:   LDA 0 DECML      ;LOAD # OF DIGITS AFTER DECIMAL
52 00141'024746          LDA 1 NVAL       ;LOAD # DIGITS IN RESULT
53 00142'106433          SUBZ# 0 1 SNC    ;TEST IF ANY BLANKS AFTER "."
54 00143'000425          JMP ADZRO        ;A BLANK - REPLACE BY "0"
55 00144'020735          LDA 0 POINT+2    ;GET ADDRESS OF LAST DIGIT
56 00145'024743          LDA 1 NUMBR      ;LOAD # OF DIGITS TO PRINT
57 00146'122400          SUB 1 0          ;GET POSITION OF FIRST DIGIT
58 00147'030745          LDA 2 DECML      ;LOAD # OF DIGITS AFTER DECIMAL
59 00150'146405          SUB 2 1 SNR      ;GET # DIGITS BEFORE DECIMAL
```

```
  0003  RTBDO
01 00151'000402           JMP .+2           ;NO DIGITS - SO DON'T PRINT ANY
02 00152'004427           JSR PRINT         ;PRINT LEADING DIGITS
03 00153'030741           LDA 2 DECML       ;LOAD # DIGITS AFTER DECIMAL
04 00154'151005           MOV 2 2 SNR       ;TEST IF DECIMAL WANTED
05 00155'000404           JMP FINIS         ;NO - SO DON'T PRINT ANY
06 00156'020737           LDA 0 DOTAD       ;LOAD ADDRESS OF "."
07 00157'126520           SUBZL 1 1         ;PRINT ONE CHARACTER
08 00160'004421           JSR PRINT         ;PRINT "."
09 00161'024733 FINIS:    LDA 1 DECML       ;GET REMAINING DIGITS
10 00162'020717           LDA 0 POINT+2     ;GET ADDRESS OF FINAL DIGIT
11 00163'122400           SUB 1 0           ;GET ADDRESS FOR PRINT
12 00164'125400           INC 1 1           ;INCREMENT TO INCLUDE FINAL SPACE
13 00165'004414           JSR PRINT         ;PRINT REMAINING CHARACTERS
14 00166'030723           LDA 2 AC2         ;RESTORE AC2
15 00167'002723           JMP @AC3          ;RETURN
16
17 00170'034711 ADZRO:    LDA 3 POINT+2     ;GET ADDRESS OF LAST DIGIT
18 00171'125400           INC 1 1           ;INCREMENT DIGIT COUNTER
19 00172'136400           SUB 1 3           ;GET CORRECT POSITION FOR "0"
20 00173'030711           LDA 2 C60         ;LOAD "0"
21 00174'051400           STA 2 0 3         ;REPLACE BLANK WITH "0"
22 00175'030713           LDA 2 NUMBR
23 00176'132433           SUBZ# 1 2 SNC     ;SKIP IF NUMBR >OR= NVAL
24 00177'010711           ISZ NUMBR         ;INCREMENT # PRINTS
25 00200'000742           JMP OUTPT+2
26
27 00201'054412 PRINT:    STA 3 RET
28 00202'040405           STA 0 PPP
29 00203'044405           STA 1 NNN
30 00204'177777           .IOX
31 00205'000000           0
32 00206'120000           120000
33 00207'000013'PPP:      RESLT
34 00210'000006 NNN:      6
35 00211'000212'          .+1
36 00212'002401           JMP @RET
37 00213'000000 RET:      0
38
39                        .END
```

```
01                      .TITL MATHS
02                      .ENT BIDEC MULTY DIVID
03                      .NREL
04
05                  ;THESE PROGRAMS ARE ALL RE-ENTRANT
06                  ;EACH WILL BE DESCRIBED SEPARATELY BELOW
07
08                  ;PROGRAM 1  -  BINARY TO DECIMAL CONVERSION
09                  ;   CALL SEQUENCE
10                  ;       JSR BIDEC
11                  ;       .BLK 7
12                  ;       <RETURN>
13                  ;               CALL WITH AC0 CONTAINING BINARY
14                  ;               DECIMAL NUMBER RETURNED IN FIRST SIX
15                  ;               LOCATIONS FOLLOWING CALL STATEMENT
16                  ;               SIGN GIVEN IN FIRST LOCATION IN ASCII
17
18 00000'024437 BIDEC:  LDA 1 C60       ;LOAD ASCII "0"
19 00001'045400         STA 1 0 3       ;STORE IN SIGN INDICATOR
20 00002'101113         MOVL# 0 0 SNC   ;TEST SIGN OF NUMBER
21 00003'000404         JMP .+4         ;POSITIVE NUMBER - JUMP
22 00004'100400         NEG 0 0
23 00005'024431         LDA 1 NSIGN     ;LOAD ASCII "-"
24 00006'045400         STA 1 0 3       ;STORE IN SIGN ADDRESS
25 00007'030421         LDA 2 TEST      ;LOAD ADDRESS OF POWERS OF TEN TABLE
26 00010'151400 NEXT:   INC 2 2         ;INCREMENT DIVISOR ADDRESS
27 00011'175400         INC 3 3         ;INCREMENT DIGIT ADDRESS
28 00012'024425         LDA 1 C60       ;LOAD ASCII "0"
29 00013'045400         STA 1 0 3       ;STORE AS START FOR DIGIT
30 00014'025000         LDA 1 0 2       ;GET DIVISOR
31 00015'125212         MOVR# 1 1 SZC   ;TEST IF LAST DIVISOR
32 00016'000406         JMP END         ;YES
33 00017'122512         SUBL# 1 0 SZC   ;TEST IF DIVISOR GOES IN
34 00020'000770         JMP NEXT        ;NO - GET NEXT DIVISOR
35 00021'122400         SUB 1 0         ;SUBTRACT DIVISOR
36 00022'011400         ISZ 0 3         ;INCREMENT DIGIT
37 00023'000774         JMP .-4         ;TRY AGAIN
38 00024'025400 END:    LDA 1 0 3       ;LOAD LAST "0"
39 00025'107000         ADD 0 1         ;GET LAST NUMBER
40 00026'045400         STA 1 0 3       ;STORE IT
41 00027'001402         JMP 2 3         ;RETURN TO CALLING PROGRAM
42
43 00030'000030'TEST:   TEST
44 00031'023420         10000.
45 00032'001750         1000.
46 00033'000144         100.
47 00034'000012         10.
48 00035'000001         1.
49 00036'000055 NSIGN:  55
50 00037'000060 C60:    60
51
52                  ; PROGRAM 2 - SINGLE PRECISION MULTIPLY
53                  ;   CALL SEQUENCE
54                  ;       JSR (MULTY)
55                  ;       16.
56                  ;       <RETURN>
57                  ;               CALL WITH NUMBERS IN AC0 AND AC1
58                  ;               HIGHER ORDER RESULT IN AC0
59                  ;               LOWER ORDER RESULT IN AC1 RETURNED
```

```
01
02 00040'020422 MULTY:   LDA 0 C16        ;LOAD COUNTER
03 00041'041400          STA 0 0 3        ;STORE COUNTER
04 00042'102400          SUB 0 0          ;CLEAR AC0
05 00043'125203          MOVR 1 1 SNC     ;CHECK NEXT MULTIPLIER BIT
06 00044'101201          MOVR 0 0 SKP     ;BIT=0, SHIFT ONLY
07 00045'143220          ADDZR 2 0        ;BIT=1, ADD MULTIPLIER AND SHIFT
08 00046'015400          DSZ 0 3          ;COUNT STEP
09 00047'001401          JMP 1 3          ;RETURN
10
11                ; PROGRAM 3 - SINGLE PRECISION DIVIDE
12                ;    CALL SEQUENCE
13                ;         JSR (DIVID)
14                ;         16.
15                ;         <RETURN>
16                ;                  AC0 AND AC1 CONTAIN HIGHER AND LOWER
17                ;                  ORDER NUMERATOR RESPECTIVELY
18                ;                  AC2 CONTAINS DENOMINATOR
19                ;                  RETURNS WITH RESULT IN AC1
20
21 00050'125120 DIVID:   MOVZL 1 1        ;SHIFT DIVIDEND LOW PART
22 00051'101100          MOVL 0 0         ;SHIFT DIVIDEND HIGH PART
23 00052'142412          SUB# 2 0 SZC     ;DOES DIVISOR GO IN
24 00053'142400          SUB 2 0
25 00054'125100          MOVL 1 1
26 00055'015400          DSZ 0 3          ;COUNT STEP
27 00056'000773          JMP DIVID+1
28 00057'030403          LDA 2 C16        ;RELOAD COUNTER
29 00060'051400          STA 2 0 3        ;STORE COUNTER
30 00061'001401          JMP 1 3          ;RETURN
31
32 00062'000020 C16:     16.
33
34                       .END
```

```
 0001  INFCE
01                      .TITL INFCE
02                      .ENT RELYS TTOO ATOD TRANS ATOD1 SRLYS TTR TTC
03                      .EXTN .IOX .WAIT
04                      .EXTN DIVID .QUIT
05                      .NREL
06
07                ; THIS ROUTINE SETS THE RELAY PATTERN
08                ; MUST BE HIGHEST PRIORITY IN SYSTEM
09                ; CALL SEQUENCE
10                ;       .FORK
11                ;       1,
12                ;       RELYS
13                ;               AC0 SHOULD CONTAIN DESIRED RELAYS
14                ;               AC1 SHOULD CONTAIN RELAYS THAT SHOULD
15                ;               NOTBE CHANGED
16
17 00000'000000 SRLYS:  0                       ;CURRENT STATUS OF RELAYS
18
19 00001'030777 RELYS:  LDA 2 SRLYS,            ;LOAD CURRENT RELAY STATUS
20 00002'133400         AND 1 2                 ;KEEP RELAYS NOT TO BE CHANGED
21 00003'113000         ADD 0 2                 ;ADD NEW RELAY CONFIGERATION
22 00004'071025         DOA 2 25                ;OUTPUT RELAY CONFIGERATION
23 00005'050773         STA 2 SRLYS             ;UPDATE RELAY STATUS
24 00006'177777         .QUIT
25
26                ; THIS ROUTINE OUTPUTS A STRING OF CHARACTERS TO TELETYPE
27                ;       CALLING SEQUENCE
28                ;          JSR TTOO
29                ;          ADDRESS OF BYTE POINTER
30                ;       RETURN
31                ; AC1, AC2 AND CARRY REMAIN UNCHANGED
32
33 00007'023400 TTOO:   LDA 0 @0 3              ;LOAD ADDRESS OF BYTE POINTER
34 00010'040406         STA 0 A                 ;STORE IN CORRECT POSITION
35 00011'175400         INC 3 3                 ;GET RETURN ADDRESS
36 00012'054410         STA 3 AA                ;SAVE RETURN ADDRESS
37 00013'177777         .IOX
38 00014'000000         0                       ;TELETYPE
39 00015'102000         102000                  ;OUTPUT
40 00016'000000 A:      0                       ;ADDRESS OF BYTE POINTER
41 00017'000454         300.                    ;# OF CHARACTERS (RTOS STOPS AT FIRST
42 00020'000021'        .+1                     ;ERROR RETURN ADDRESS
43 00021'002401         JMP @AA                 ;RETURN TO CALLING PROGRAM
44 00022'000000 AA:     0
45
46                ; THIS ROUTINE CONTINUALLY READS THE A/D CHANNELS
47
48 00023'000013'TRANS:  .IOX
49 00024'000007         7                       ;A/D
50 00025'000000         0
51 00026'000075'        ATOD                    ;ADDRESS OF DATA STORAGE
52 00027'000020         16.                     ;READ ALL 16 CHANNELS
53 00030'000031'        .+1                     ;ERROR RETURN ADDRESS
54
55                ;THE FOLLOWING IS A DIGITAL FILTER
56
57 00031'024444         LDA 1 ATOD              ;LOAD REACTOR TEMPERATURE
58 00032'020437         LDA 0 TR                ;LOAD RUNNING SUM
59 00033'123000         ADD 1 0
```

```
  0002  INFCE
01 00034'040435          STA 0 TR
02 00035'034441          LDA 3 ATOD1
03 00036'030432          LDA 2 TC
04 00037'173000          ADD 3 2
05 00040'050430          STA 2 TC
06 00041'014433          DSZ CONTR
07 00042'000761          JMP TRANS
08 00043'050423          STA 2 SAVE
09 00044'105000          MOV 0 1
10 00045'102400          SUB 0 0
11 00046'030421          LDA 2 NUMBR
12 00047'006416          JSR @POINT
13 00050'000020          16.
14 00051'044421          STA 1 TTR
15 00052'024414          LDA 1 SAVE
16 00053'102400          SUB 0 0
17 00054'006411          JSR @POINT
18 00055'000020          16.
19 00056'044415          STA 1 TTC
20 00057'102400          SUB 0 0          ;CLEAR AC0
21 00060'040411          STA 0 TR         ;RESET SUMMING REGISTERS
22 00061'040407          STA 0 TC
23 00062'020405          LDA 0 NUMBR      ;RESET COUNTER
24 00063'040411          STA 0 CONTR
25 00064'000737          JMP TRANS        ;REPEAT ROUTINE
26
27 00065'177777 POINT:   DIVID
28 00066'000000 SAVE:    0
29 00067'000037 NUMBR:   37
30 00070'000000 TC:      0
31 00071'000000 TR:      0
32 00072'000000 TTR:     0
33 00073'000000 TTC:     0
34 00074'000010 CONTR:   8.
35 00075'000000 ATOD:    0
36        000017 ATOD1:  .BLK 15.
37
38                       .END
```

```
  0001  CLOCK
01                      .TITL CLOCK
02                      .ENT CLOCK HOUR MINIT SECND RTIME
03                      .EXTN RTDBO .WAIT .FORK .XMIT .RCV .QUIT
04                      .NREL
05
06              ; THIS PROGRAM KEEPS TRACK OF TIME
07
08 00000'130740 RTIME:  -20000.
09 00001'000000 HOUR:   0
10 00002'000000 MINIT:  0
11 00003'000000 SECND:  0
12 00004'000074 C60:    60.
13 00005'000015 C13:    13.
14
15 00006'177777 CLOCK:  .FORK
16 00007'000005          5
17 00010'000045'         TIME
18 00011'020773          LDA 0 C60
19 00012'030773          LDA 2 C13
20 00013'177777 RCVE:    .RCV              ;WAIT FOR INDICATION THAT A SECOND HAS
21 00014'000004          4
22 00015'010763          ISZ RTIME
23 00016'000401          JMP .+1
24 00017'177777          .XMIT
25 00020'000001          1
26 00021'024762          LDA 1 SECND
27 00022'125400          INC 1 1           ;INCREMENT SECOND COUNTER
28 00023'106415          SUB# 0 1 SNR      ;TEST IF MINUTE HAS PASSED
29 00024'126400          SUB 1 1           ;YES - SET SECOND COUNTER TO ZERO
30 00025'044756          STA 1 SECND
31 00026'125004          MOV 1 1 SZR       ;TEST SECOND COUNTER
32 00027'000764          JMP RCVE
33 00030'024752          LDA 1 MINIT       ;MINUTE HAS PASSED
34 00031'125400          INC 1 1           ;INREMENT MINUTE COUNTER
35 00032'106415          SUB# 0 1 SNR      ;TEST IF HOUR HAS PASSED
36 00033'126400          SUB 1 1           ;YES - ZERO HOUR COUNTER
37 00034'044746          STA 1 MINIT
38 00035'125004          MOV 1 1 SZR       ;TEST MINUTE COUNTER
39 00036'000755          JMP RCVE
40 00037'024742          LDA 1 HOUR        ;HOUR HAS PASSED
41 00040'125400          INC 1 1           ;INCREMENT HOUR COUNTER
42 00041'146415          SUB# 2 1 SNR      ;TEST IF 12 HOURS HAVE PASSED
43 00042'126520          SUBZL 1 1         ;YES - SET HOUR COUNTER = 1
44 00043'044736          STA 1 HOUR
45 00044'000747          JMP RCVE
46
47 00045'177777 TIME:    .WAIT             ;WAIT FOR ONE SECOND
48 00046'000144          100.
49 00047'000017'         .XMIT             ;TELL CLOCK A SECOND HAS PASSED
50 00050'000004          4
51 00051'000774          JMP TIME          ;GO WAIT ANOTHER SECOND
52
53                      .END
```

```
01                      .TITL PINIT
02                      .ENT PINIT ITEM1 I1SRT I1STP ITEM2 I2SRT I2STP
03                      .ENT ITEM3 I3SRT I3STP ITEM4 I4SRT I4STP
04                      .ENT ITEM5 ITEM6 I5SRT I6SRT I5STP I6STP
05                      .ENT ITEM7 I7SRT I7STP
06                      .EXTN .RCV .FORK ERROR RTIME
07                      .NREL
08
09                ;
10                ;   PROGRAM INITIATES AND TERMINATES PROGRAMS AT DESIRED TIME
11                ;   AS SCHEDULED
12                ;
13
14 00000'177777 PINIT:  .RCV              ;WAIT FOR SECOND COUNTER FROM CLOCK
15 00001'000001          1
16 00002'022424          LDA 0 @POINT      ;FIND CURRENT TIME
17 00003'034424          LDA 3 ITEMS       ;LOAD ITEM ADDRESS
18 00004'054020          STA 3 20          ;STORE IN AUTOINREMENT REGISTER
19 00005'026020 CHECK:   LDA 1 @20         ;LOAD PROGRAM
20 00006'125005          MOV 1 1 SNR       ;CHECK IF THERE IS A PROGRAM THERE
21 00007'000771          JMP PINIT         ;NO - END OF TABLE
22 00010'044414          STA 1 PROG+3      ;STORE LOCATION OF PROGRAM
23 00011'026020          LDA 1 @20         ;LOAD STARTING TIME FOR THIS PROGRAM
24 00012'106415          SUB# 0 1 SNR      ;READY TO BE STARTED ?
25 00013'004406          JSR PROG          ;YES - GO START IT
26 00014'014410          DSZ PROG+3                ;GET ADDRESS TO STOP PROGRAM
27 00015'026020          LDA 1 @20         ;LOAD STOPPING TIME
28 00016'106415          SUB# 0 1 SNR      ;READY TO BE STOPPED ?
29 00017'004402          JSR PROG          ;YES - GO START IT
30 00020'000765          JMP CHECK
31 00021'171000 PROG:    MOV 3 2
32 00022'177777          .FORK
33 00023'000050          50
34 00024'177777          ERROR
35 00025'001000          JMP 0 2
36
37 00026'177777 POINT:   RTIME
38
39 00027'000027'ITEMS:   ITEMS
40 00030'000000 ITEM1:   0
41 00031'000000 I1SRT:   0
42 00032'000000 I1STP:   0
43 00033'000000 ITEM2:   0
44 00034'000000 I2SRT:   0
45 00035'000000 I2STP:   0
46 00036'000000 ITEM3:   0
47 00037'000000 I3SRT:   0
48 00040'000000 I3STP:   0
49 00041'000000 ITEM4:   0
50 00042'000000 I4SRT:   0
51 00043'000000 I4STP:   0
52 00044'000000 ITEM5:   0
53 00045'000000 I5SRT:   0
54 00046'000000 I5STP:   0
55 00047'000000 ITEM6:   0
56 00050'000000 I6SRT:   0
57 00051'000000 I6STP:   0
58 00052'000000 ITEM7:   0
59 00053'000000 I7SRT:   0
```

```
  0002  FINIT
01 00054'000000 I7STP:   0
02 00055'000000          0
03
04                       . END
```

```
  0001  UVANS
01                      .TITL UVANS
02                      .ENT UVANS
03                      .EXTN .RCV .WAIT FLAGL TTOO RTBDO LOG
04                      .EXTN SMIN SSEC .FORK .QUIT
05                      .NREL
06
07                ; THIS PROGRAM SUPERVISES THE OPERATION OF THE UV
08                ; SPECTROPHOTOMETER, ANALYSIS AND PRINTING OF
09                ; RESULTS
10
11 00000'177777 UVANS:  .RCV              ;WAIT FOR SAMPLE INDICATION FROM
12 00001'000003          3                ;SAMPLE PROGRAM
13 00002'022413          LDA 0 @POINT+3   ;GET TIME WHEN SAMPLE WAS TAKEN
14 00003'040415          STA 0 STIME      ;SAVE TIME IN MINUTES
15 00004'022412          LDA 0 @POINT+4
16 00005'040414          STA 0 STIME+1    ;SAVE TIME IN SECONDS
17 00006'177777          .FORK
18 00007'000100          100
19 00010'000042'         PRINT
20 00011'000767          JMP UVANS
21
22                ; DATA SECTION FOR UVANS
23
24 00012'177777 POINT:   TTOO
25 00013'177777          RTBDO
26 00014'177777          FLAGL
27 00015'177777          SMIN
28 00016'177777          SSEC
29 00017'177777          LOG
30
31 00020'000000 STIME:   0
32 00021'000000          0
33 00022'000000 POSTN:   0
34 00023'000023'WAVE:    WAVE
35 00024'001130 WAVE1:   600.
36 00025'000000 TURB1:   0
37 00026'000764 WAVE2:   500.
38 00027'000000 TURB2:   0
39 00030'000620 WAVE3:   400.
40 00031'000000 TURB3:   0
41 00032'000536 WAVE4:   350.
42 00033'000000 TURB4:   0
43 00034'000454 WAVE5:   300.
44 00035'000000 TURB5:   0
45 00036'000000          0
46
47 00037'000000 SNO:     0
48 00040'000006 C6:      6
49 00041'001406 P6:      1406
50
51                ; THIS SECTION PRINTS RESULTS OF UV ANALYSIS
52
53 00042'022752 PRINT:   LDA 0 @POINT+2   ;LOAD LOGGING FLAG
54 00043'101004          MOV 0 0 SZR      ;TEST IF LOG ROUTINE RUNNING
55 00044'000404          JMP CONT         ;YES - SO CAN PRINT
56 00045'177777          .WAIT
57 00046'000144          100.
58 00047'000773          JMP PRINT
59 00050'102400 CONT:    SUB 0 0          ;CLEAR AC0
```

```
  0002  UVANS
01 00051'042743          STA 0 @POINT+2  ;STOP LOGGING ROUTINE
02 00052'000045'         .WAIT           ;MAKE SURE LOGGING HAS STOPPED
03 00053'000120          80.
04 00054'006736          JSR @POINT      ;PRINT SAMPLE #
05 00055'000112'         MESSA
06 00056'010761          ISZ SNO         ;INCREMENT SAMPLE #
07 00057'020760          LDA 0 SNO
08 00060'126400          SUB 1 1         ;INDICATE MINIMUM PRINTING
09 00061'006732          JSR @POINT+1    ;PRINT NUMBER
10 00062'006730          JSR @POINT      ;PRINT "TAKEN AT (SEC)"
11 00063'000113'         MESSB
12 00064'020735          LDA 0 STIME+1
13 00065'126400          SUB 1 1         ;MINIMUM PRINTING
14 00066'006725          JSR @POINT+1    ;PRIT TIME IN SECONDS
15 00067'006723          JSR @POINT      ;PRINT WAVELENGTH AND TURBIDITY
16 00070'000114'         MESSC
17 00071'030732          LDA 2 WAVE      ;LOAD ADDRESS OF WAVELENGTH
18 00072'050730          STA 2 POSTN     ;STORE IN POSITION INDICATOR
19 00073'006717 REPET:   JSR @POINT      ;PRINT CR AND LF
20 00074'000115'         MESSD
21 00075'010725          ISZ POSTN       ;INCREMENT POSITION INDICATOR
22 00076'022724          LDA 0 @POSTN    ;LOAD VALUE OF WAVELENGTH
23 00077'024741          LDA 1 C6
24 00100'101005          MOV 0 0 SNR     ;TEST IF AT END OF TABLE
25 00101'000407          JMP ENDP
26 00102'006711          JSR @POINT+1    ;PRINT WAVELENGTH VALUE
27 00103'010717          ISZ POSTN       ;INCREMENT POSITION INDICATOR
28 00104'022716          LDA 0 @POSTN    ;LOAD TURBIDITY
29 00105'024734          LDA 1 P6
30 00106'006705          JSR @POINT+1
31 00107'000764          JMP REPET
32
33 00110'126520 ENDP:    SUBZL 1 1
34 00111'002706          JMP @POINT+5    ;RESTART LOGGING ROUTINE
35
36 00112'000234"MESSA:   A*2
37 00113'000252"MESSB:   B*2
38 00114'000274"MESSC:   C*2
39 00115'000322"MESSD:   D*2
40
41                A:     .TXT @<15><12><12>SAMPLE # @
   00116'005015
   00117'051412
   00120'046501
   00121'046120
   00122'020105
   00123'020043
   00124'000000
42                B:     .TXT @  TAKEN AT (SEC) @
   00125'020040
   00126'040524
   00127'042513
   00130'020116
   00131'052101
   00132'024040
   00133'042523
   00134'024503
   00135'000040
43                C:     .TXT @<15><12><12>W.L. (NM) TURBIDITY@
```

```
  0003  UVANS
   00136'005015
   00137'053412
   00140'046056
   00141'024056
   00142'046516
   00143'020051
   00144'052524
   00145'041122
   00146'042111
   00147'052111
   00150'000131
01                 D:        .TXT @<15><12> @
   00151'005015
   00152'000040
02
03                           .END
```

```
01                      .TITL LOG
02                      .ENT LOG LOGST FLAGL LGINT
03                      .EXTN SRLYS RTBDO HOUR MINIT SECND TTOO RTIME
04                      .EXTN TEMPC TEMPR
05                      .EXTN .QUIT .FORK .WAIT
06                      .NREL
07
08              ; THIS ROUTINE LOGS VALUES ON TTOO
09              ; PROGRAM CONTINUES TO RUN BUT DOES NOT LOG WHEN FLAGL=0
10              ; PROGRAM OUTPUTS TO TTO AGAIN WHEN FLAGL=0
11
12 00000'177777 POINT:  RTBDO
13 00001'177777          TTOO
14
15 00002'000002'LIST:    LIST
16 00003'177777          HOUR
17 00004'000002          2
18 00005'177777          MINIT
19 00006'000002          2
20 00007'177777          SECND
21 00010'000002          2
22 00011'177777          RTIME
23 00012'000006          6
24 00013'177777          SRLYS
25 00014'177772          -6
26 00015'177777          TEMPC
27 00016'000405          405
28 00017'177777          TEMPR
29 00020'000405          405
30 00021'000000          0                ;END OF TABLE INDICATION
31
32 00022'000000 FLAGL:   0
33 00023'000154"AA:      A*2
34 00024'000224"BB:      B*2
35
36 00025'006754 LOG:     JSR @POINT+1     ;TYPE LOGGER HEADING
37 00026'000023'         AA
38 00027'102520          SUBZL 0 0        ;SET FLAG=1 FOR LOGGER
39 00030'040772          STA 0 FLAGL
40 00031'177777          .QUIT
41
42 00032'000012 LGINT:   10.
43
44 00033'024777 LOGST:   LDA 1 LGINT      ;LOAD LOGGING INTERVAL
45 00034'124400          NEG 1 1
46 00035'177777          .WAIT            ;WAIT ONE SECOND
47 00036'000144          100.
48 00037'125404          INC 1 1 SZR      ;TEST IF INTERVAL IS UP
49 00040'000775          JMP .-3          ;NO - WAIT ANOTHER SECOND
50 00041'020761          LDA 0 FLAGL      ;LOAD FLAG
51 00042'101005          MOV 0 0 SNR      ;TEST FLAG
52 00043'000770          JMP LOGST        ;FLAG=0 SO DON'T PRINT
53 00044'177777          .FORK            ;START LOGGING ROUTINE
54 00045'000100          100
55 00046'000050'         LOGER
56 00047'000764          JMP LOGST        ;GO WAIT FOR NEXT LOG
57
58 00050'006731 LOGER:   JSR @POINT+1     ;TYPE CR AND LF
59 00051'000024'         BB
```

```
  0002  LOG
01 00052'030730          LDA 2 LIST       ;LOAD ADDRESS OF LIST
02 00053'151400          INC 2 2          ;INCREMENT ADDRESS
03 00054'035000          LDA 3 0 2       ;LOAD ADDRESS OF VALUE TO BE PRINTED
04 00055'175005          MOV 3 3 SNR      ;TEST IF END OF TABLE
05 00056'000031'         .QUIT            ;YES
06 00057'021400          LDA 0 0 3        ;NO - LOAD VALUE TO BE PRINTED
07 00060'151400          INC 2 2          ;GET ADDRESS OF TYPE OF PRINTING
08 00061'025000          LDA 1 0 2        ;LOAD VALUE
09 00062'034740          LDA 3 FLAGL      ;LOAD FLAG
10 00063'175004          MOV 3 3 SZR      ;TEST IF FLAG=1
11 00064'006714          JSR @POINT       ;YES GO PRINT VALUE
12 00065'000766          JMP LOGER+3      ;GO GET NEXT NUMBER
13
14                A:     .TXT @<15><12>   TIME  RTIME  RELAYS  TEMPC  TEMPR@
   00066'005015
   00067'020040
   00070'052040
   00071'046511
   00072'020105
   00073'051040
   00074'044524
   00075'042515
   00076'020040
   00077'042522
   00100'040514
   00101'051531
   00102'020040
   00103'042524
   00104'050115
   00105'020103
   00106'052040
   00107'046505
   00110'051120
   00111'000000
15                B:     .TXT @<15><12>@
   00112'005015
   00113'000000
16
17                       .END
```

```
01                      .TITL FEEDS
02                      .ENT MONMR INITR EMULS
03                      .ENT MAXMF MAXIF MAXEF MFLOW IFLOW EFLOW
04                      .EXTN .PTY .FORK .WAIT .QUIT RELYS
05                      .NREL
06
07                ; THIS IS A TASK PROGRAM TO CAUSE PSEUDO-VARIABLE
08                ; FEED RATES TO REACTOR
09
10                ; SECTION A - MONOMER FEED
11
12 00000'000436           JMP ENDM          ;ENTER HERE TO STOP TASK
13 00001'177777 MONMR:    .PTY              ;SET HIGH PRIORITY FOR ACCURACY
14 00002'000006           6
15 00003'020410           LDA 0 GO          ;SET PROGRAM TO RUN
16 00004'040406           STA 0 NOGOM
17 00005'177777           .FORK             ;OPEN AND CLOSE VALVE
18 00006'000050           50
19 00007'000014'          DOMON
20 00010'177777           .WAIT
21 00011'000764 MAXMF:    500.
22 00012'000773 NOGOM:    JMP .-5
23 00013'000773 GO:       JMP .-5
24
25 00014'020413 DOMON:    LDA 0 MFLOW       ;LOAD FLOWRATE
26 00015'101005           MOV 0 0 SNR       ;TEST IF 0 FLOW
27 00016'177777           .QUIT
28 00017'101112           MOVL# 0 0 SZC     ;TEST IF NEGATIVE FLOW
29 00020'000016'          .QUIT
30 00021'024414           LDA 1 MRELY       ;OPEN MONOMER VALVE
31 00022'120000           COM 1 0           ;SET RELAY ON
32 00023'000005'          .FORK
33 00024'000001           1
34 00025'177777           RELYS
35 00026'000010'          .WAIT
36 00027'000310 MFLOW:    200.
37 00030'102400           SUB 0 0           ;CLOSE MONOMER VALVE
38 00031'000023'          .FORK
39 00032'000001           1
40 00033'000025'          RELYS
41 00034'000020'          .QUIT
42
43 00035'177737 MRELY:    177737
44
45 00036'020402 ENDM:     LDA 0 STOP
46 00037'040753           STA 0 NOGOM
47 00040'000034'STOP:     .QUIT
48
49                ; SECTION B - INITIATOR FEED
50
51 00041'000435           JMP ENDI          ;ENTER HERE TO STOP TASK
52 00042'000001'INITR:    .PTY              ;SET HIGH PRIORITY FOR ACCURACY
53 00043'000006           6
54 00044'020747           LDA 0 GO          ;SET PROGRAM TO RUN
55 00045'040406           STA 0 NOGOI
56 00046'000031'          .FORK
57 00047'000007           7
58 00050'000054'          DOIN
59 00051'000026'          .WAIT             ;WAIT REQUIRED PERIOD
```

```
  0002  FEEDS
01 00052'000454 MAXIF:  300.              ;MAXIMUM INITIATOR FLOW
02 00053'000773 NOGOI:  JMP .-5
03
04 00054'020413 DOIN:   LDA 0 IFLOW       ;LOAD DESIRED INITIATOR FLOW
05 00055'101005         MOV 0 0 SNR       ;TEST IF FLOW IS 0
06 00056'000040'        .QUIT
07 00057'101112         MOVL# 0 0 SZC     ;TEST IF NEGATIVE FLOW
08 00060'000056'        .QUIT
09 00061'024414         LDA 1 IRELY       ;OPEN INITIATOR VALVE
10 00062'120000         COM 1 0
11 00063'000046'        .FORK
12 00064'000001         1
13 00065'000033'        RELYS
14 00066'000051'        .WAIT             ;WAIT REQUIRED INTERVAL
15 00067'000310 IFLOW:  200.              ;INITIATOR FLOW
16 00070'102400         SUB 0 0           ;CLOSE INITIATOR VALVE
17 00071'000063'        .FORK
18 00072'000001         1
19 00073'000065'        RELYS
20 00074'000060'        .QUIT
21 00075'177757 IRELY:  177757            ;INDICATION FOR RELAY PROGRAM
22
23 00076'020742 ENDI:   LDA 0 STOP
24 00077'040754         STA 0 NOGOI
25 00100'000074'        .QUIT
26
27              ; SECTION C - EMULSIFIER FEED
28
29 00101'000435         JMP ENDE
30 00102'000042'EMULS:  .PTY              ;SET HIGH PRIORITY FOR ACCURACY
31 00103'000006         6
32 00104'020707         LDA 0 GO
33 00105'040406         STA 0 NOGOE
34 00106'000071'        .FORK
35 00107'000007         7
36 00110'000114'        DOEM
37 00111'000066'        .WAIT
38 00112'000310 MAXEF:  200.
39 00113'000773 NOGOE:  JMP .-5
40
41 00114'020413 DOEM:   LDA 0 EFLOW
42 00115'101005         MOV 0 0 SNR       ;TEST IF FLOW 0
43 00116'000100'        .QUIT
44 00117'101112         MOVL# 0 0 SZC     ;TEST FOR NEGATIVE FLOW
45 00120'000116'        .QUIT
46 00121'024414         LDA 1 ERELY
47 00122'120000         COM 1 0
48 00123'000106'        .FORK
49 00124'000001         1
50 00125'000073'        RELYS
51 00126'000111'        .WAIT
52 00127'000226 EFLOW:  150.
53 00130'102400         SUB 0 0
54 00131'000123'        .FORK
55 00132'000001         1
56 00133'000125'        RELYS
57 00134'000120'        .QUIT
58
59 00135'177767 ERELY:  177767
```

```
01
02 00136'020702 ENDE:    LDA 0 STOP
03 00137'040754          STA 0 NOGOE
04 00140'000134'         .QUIT
05
06                       .END
```

```
01                      .TITL TCTRL
02                      .ENT TCTRL TSP TEMPC TEMPR TCINT TYPEC
03                      .EXTN .QUIT .WAIT .FORK TTR TTC RELYS
04                      .NREL
05
06                  ; THIS PROGRAM CONTROLS REACTOR TEMPERATURE
07                  ;  IT IS A TASK PROGRAM
08
09 00000'000472             JMP END
10 00001'020414 TCTRL:      LDA 0 GO
11 00002'040412             STA 0 PLACE
12 00003'177777             .FORK
13 00004'000051             51
14 00005'000027'            FFORM
15 00006'024412             LDA 1 TCINT         ;FIND CONTROL INTERVAL
16 00007'124400             NEG 1 1
17 00010'177777             .WAIT
18 00011'000144             100.                ;WAIT ONE SECOND
19 00012'125404             INC 1 1 SZR
20 00013'000775             JMP .-3
21 00014'000767 PLACE:      JMP .-9.
22 00015'000767 GO:         JMP .-9.
23 00016'000764 STOP:       JMP .-12.
24
25 00017'000000 TYPEC:      0
26 00020'000001 TCINT:      1
27 00021'000764 TSP:        500.
28 00022'000620 TEMPC:      400.
29 00023'000763 TEMPR:      499.
30 00024'000144 LOWER:      100.
31 00025'177777 POINT:      TTR
32 00026'177777             TTC
33
34 00027'024775 FFORM:      LDA 1 LOWER         ;LOAD ABSOLUTE 0 DEG CENT
35 00030'022775             LDA 0 @POINT        ;LOAD RECTOR TEMP.  (A/D UNITS)
36 00031'122400             SUB 1 0
37 00032'101120             MOVZL 0 0           ;MULT. * 2.
38 00033'040770             STA 0 TEMPR
39 00034'022772             LDA 0 @POINT+1      ;LOAD COOLANT TEMPERATURE
40 00035'122400             SUB 1 0
41 00036'101120             MOVZL 0 0
42 00037'040763             STA 0 TEMPC         ;STORE TEMP. CONVERTED TO CENTIGRADE
43 00040'020757             LDA 0 TYPEC         ;CHECK CONTROL DESIRED
44 00041'101004             MOV 0 0 SZR         ;CHECK FOR ANY CONTROL
45 00042'000403             JMP ONOFF           ;ON - OFF CONTROL
46 00043'177777             .QUIT               ;NO CONTROL DESIRED
47
48 00044'000001 RANGE:      1
49
50 00045'020756 ONOFF:      LDA 0 TEMPR         ;GET TEMPERATURE OF REACTOR
51 00046'024753             LDA 1 TSP           ;GET DESIRED SET POINT
52 00047'030775             LDA 2 RANGE         ;GET RANGE OF NO ACTION
53 00050'122400             SUB 1 0
54 00051'142512             SUBL# 2 0 SZC       ;IS TEMPERATURE TOO HIGH
55 00052'100401             NEG 0 0 SKP         ;NO - TEST FOR TOO COLD
56 00053'000406             JMP COOL            ;YES - COOL REACTOR
57 00054'142512             SUBL# 2 0 SZC       ;IS TEMPERATURE TOO LOW
58 00055'000043'            .QUIT               ;IN BETWEEN LEAVE AS IS
59 00056'000401             JMP HEAT            ;YES - HEAT REACTOR
```

```
  0002  TCTRL
01
02 00057'020410 HEAT:   LDA 0 RHEAT     ;LOAD HOT WATER VALVE RELAY
03 00060'000402         JMP .+2
04
05 00061'020407 COOL:   LDA 0 RCOOL     ;LOAD COLD WATER RELAY
06 00062'024407         LDA 1 TRELY
07 00063'000003'        .FORK
08 00064'000001         1
09 00065'177777         RELYS
10 00066'000055'        .QUIT
11
12 00067'000004 RHEAT:  4
13 00070'000006 RCOOL:  6
14
15 00071'177771 TRELY:  177771
16
17 00072'020724 END:    LDA 0 STOP      ;THIS INSTRUCTION STOPS CONTROLLER
18 00073'040721         STA 0 PLACE
19 00074'102400         SUB 0 0         ;D/A 0
20 00075'000765         JMP COOL+1
21
22                      .END
```