# THE APPLE II/STOPPED-FLOW

# INTERFACE PACKAGE

by

Daniel M. Laby

* * * * * * *

Submitted in partial fulfillment

of the requirements for

Honors in the Department of Chemistry

UNION COLLEGE

May, 1983

## ABSTRACT

LABY, DANIEL M. The Apple II/Stopped-Flow Interface Package. Department of Chemistry, May 1983

The Apple II+ microcomputer was interfaced to an Aminco-Morrow stopped-flow Apparatus. This interface package can be broken into two parts; the hardware, or electrical connections necessary to link the two instruments, and secondly, the software, or programs written to control the interface package.

A circuit was constructed as part of the interface hardware. This circuit consisted of several operational amplifiers connected in such a way as to transform the output voltage of the stopped-flow into one acceptable by the Apple computer.

The software for this interface package is extensive. Programs were written that allowed the user to initialize the correct interface conditions, save any data from a particular run, and draw a rough plot of the data collected. Programs are also included for reactions that follow first order kinetics. These programs calculate a rate constant and intercept for the best fit straight line of a first order kinetic plot.

## TABLE OF CONTENTS

This project has involved the interfacing of the stopped-flow instrument to the Apple II+ computer. There are several reasons for undertaking this project. The primary reason is the ease in data collection offered by the use of a computer. Without the computer to aid in data collection and manipulation, the user must record all data and perform any calculations by hand. With it the user can ask the computer to record and calculate all necessary information. Another reason for desiring an interface is the ability to obtain greater accuracy in recording data. The human eye is limited in its ability to read a value from a screen, while the computer, which operates electronically is limited only by the word size used to record the signal.

This interface project can be broken down into two main sections: hardware and software. The hardware section includes: the instrument itself, the interface circuit, the analog to digital (A/D) converter, and the clock. The second section deals solely with the software or programs used in the interface package. These programs include: the primary menu, the set up program, the data collection and storage programs, and finally, the calculation and ploting programs. This paper will deal with each of these sections individually and will include a user's manual for the interface package itself.

Before discussing the hardware section of the interface, it is important to gain a general understanding of what happens during a run on the stopped-flow. By depressing the plunger on the stopped-flow apparatus the user, causes the mixing of two reagents; these reagents have a characteristic transmitance of monochromatic electromagnetic radiation. As the reaction continues, this transmitance value changes. As the transmitance changes so does the output of the photometer. Since the photometer is an electrical instrument its output consists of an electrical signal. This signal is first offset and then brought into the computer.

By setting the power supply voltage of the photometer we can create an output range between ground (zero) and negative ten volts. This signal must then be converted into a signal that is understandable by the Apple computer. The computer only understands quantities that have fixed values (digital signals), while the output of the photometer is a continuously varyiable electrical signal (analog signal); therefore, an analog to digital converter is needed. The A/D is mounted on a computer board and can be found within the computer. There is one important restriction of the A/D board: the input to the board must be between minus five and plus five volts. We stated earlier that the output of the photometer was zero to negative ten volts. Since this does not match the input requirements of the A/D board, an electrical circuit was developed that changed the output of the photometer to one satisfying the input needs of the A/D board.

The computer is directed to begin collecting data when it receives a trigger signal. This signal is produced by the stopped flow instrument when a run is started. The signal passes through the A/D board and into the computer. Whithin the software of this interface package is a program loop that waits for the trigger signal before collecting any data.

Once the data are in the computer we must find a way to regulate how often a data point is recorded. The best way to accomplish this is by using a clock which has readable values in the millisecond range. On a second board mounted within the computer is a clock which can be read from within a program.

The final step in collecting data is saving it permanently on the disk. This is accomplished with the BASIC-DOS computing language. Since the stopped-flow instrument is designed for use with relatively fast reactions, the actual data collection program was written in 6502 Assembly language. BASIC instructions are executed more slowly than Assembly language instructions.

The final programs in the interface package involve a rough plot of the data and a calculation based on the assumption that the user is working with reactions that are first order with respect to a given reactant.

We have given a short description of the different functions of the interface package, and now we will explore, in greater depth, each part of this package to see how it contributes to the interface package as a whole.

# I. HARDWARE

## 1)INSTRUMENTATION

The instrument used for this project was the Aminco-Morrow Stopped-Flow Apparatus. Associated with this instrument are several supporting devices: a Beckman Spectrophotometer which is used to follow the reaction, a High Performance Kinetic Photometer, and a Dual power supply.

The stopped flow apparatus is diagramed in figure 1. The two main parts of interest here are the trigger switch and the mixing chamber/observation cell. The observation cell is a transparent compartment through which passes monochromatic light of a pre-set wavelength. This light is then transmitted to a photomultiplier detector. The trigger switch is used to start the computer's data collection. This switch operates by closing the battery circuit when the plunger rises after the reaction is started. The battery is a fourteen volt transistor battery; we have slightly modified the original design of the switch by introducing a potentiometer into the trigger circuit. This was necessary in order not to overload the A/D board with more than the maximum five volts.

Several parts of the related instrumentation should

FIGURE 1 - Schematic of Stopped Flow Instrument

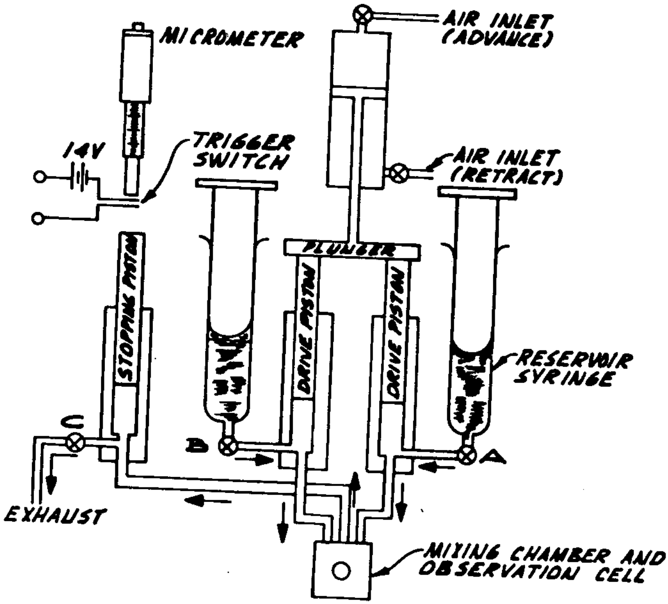
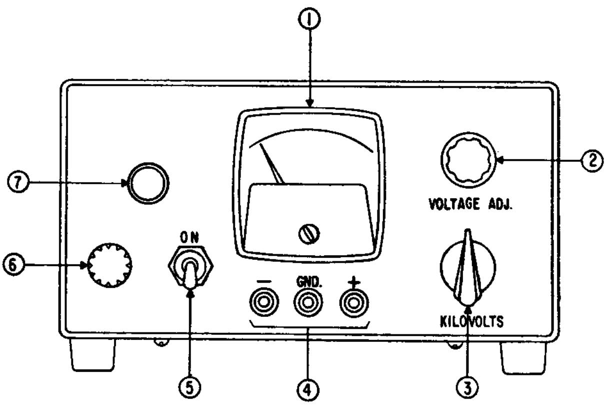



Figure 1. Schematic diagram, Stopped-flow Apparatus

be mentioned at this point. In order to offset any dark current produced by the photomultiplier tube (pmt), it is necessary to add a fixed voltage to the output of the pmt. This is accomplished by turning the fine offset knob on the High Performance Kinetic Photometer until the output of the pmt is zero volts (with the shutter closed in front of the pmt). Once the dark current is offset, the user must set the ten volt maximum on the dual power supply. This is accomplished by opening the shutter and placing fresh water in the observation cell. By turning the coarse and fine voltage adjust until the power supply voltage (see figure 2) is ten volts, the user has set the maximum electronic output of the stopped flow instrument. The user should be sure that the ten volt range is set correctly (it should recalibrated every thirty minutes). If the ten volt difference decreases, the user will see a marked effect on the results.

FIGURE 2 - Stopped Flow Power Supply

POWER SUPPLY



## 2)INTERFACE CIRCUIT AND POWER SUPPLY

As stated earlier, the output of the photometer needs to be offset to match the input requirements of the A/D board. This task is accomplished by building an electronic circuit using operational amplifiers (op-amps) that will perform the necessary offset. Op-amps have several characteristics that make them very useful in this application. They have a high input impedance while maintaining a low output impedance; this allows the user to build an op-amp into a circuit without loading the circuit. In operation, op-amps tend to keep their two inputs at the same potential; in other words, there is no potential difference between the two inputs. The final characteristic of an op-amp is the fact that the user does not need to know what is happening within the op-amp itself to use it. The external connections are all that is needed to make proper use of the amplifier.

Our circuit consists of two op-amps (see figure 3). The first op-amp encountered by the incoming signal is designed in the follower configuration. The output from the follower is unchanged from its input, but the high input impedance of this configuration prevents the interface circuit from loading the photometer in the stopped flow instrument.

The second op-amp offsets the signal from the stopped

FIGURE 3 - Interface Circuit

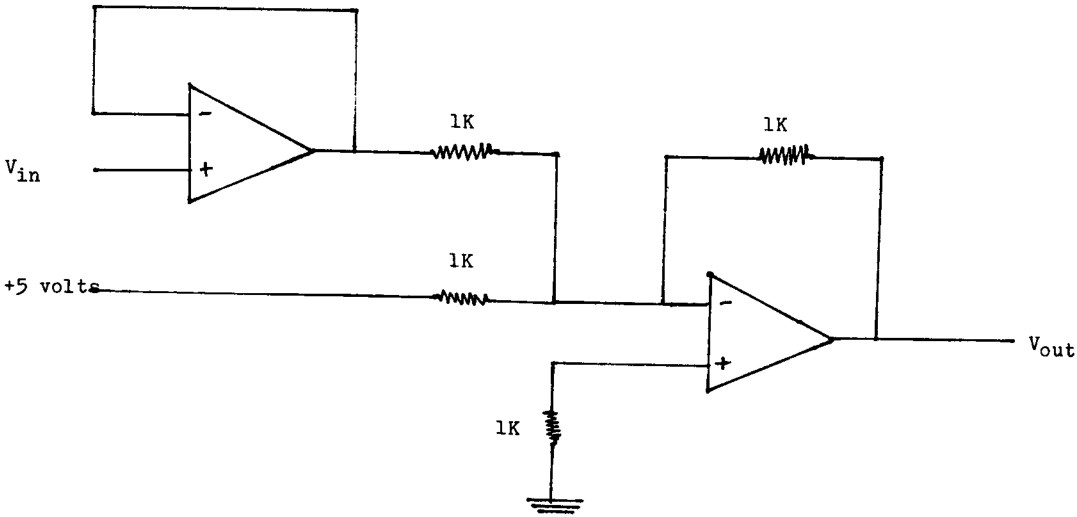



FIGURE 3

$$V_{out} = -(V_{in}+5)$$

$V_{out}$=Output from Op Amp circuit (-5 to +5 volts)

$V_{in}$ =Input to Op Amp circuit (0 to -10 volts) flow by a positive five volts and inverts its sign. Thus, these two op-amps in unison change a minus ten to zero volt signal into a plus five to minus five volt signal, while at the same time limiting the amount of current drawn by the circuit.

It is fairly simple to analyze the circuit in figure 3. As stated previously, the input voltage (Vin) from the stopped-flow instrument first encounters an op-amp in the follower configuration. If we label the output from this op-amp Vo' we can write an expression for this voltage:

$$Vo' = Vin$$

After passing the first op-amp the signal continues through the circuit to the second op-amp. The expression for the output voltage (Vo) at the second op-amp is:

$$Vo = -(Vo'(Rfb/Rin) + 5(Rfb/Rin))$$

If we let the resistor in the feedback loop equal the input resistance this expression simplifies to :

$$Vo = -(Vo' + 5)$$

By substituting for the output of the first op-amp, we derive the final expression for this circuit :

$$V_o = -(V_{in} + 5)$$

It is clear from the final expression that an input range of zero to minus ten volts will produce an output range of minus five to plus five volts.

The final piece of hardware important to this section is a standard fifteen volt power supply. The fifteen volt terminals are used to power the op-amps while the five volt terminal is used to offset the input voltage from the stopped-flow instrument.

3)Analog to Digital converter

This piece of hardware is the heart of the interface package. The A/D converter links the outside world to the world of the computer. The converter installed in the Apple computer is manufactured by the Mountain Computer company. The A/D board does not require any memory or input/output devices except for its two ribbon connectors. There are a total of sixteen channels available to the user for analog to digital conversion.

As mentioned in the previous section, the A/D board is used in the interface package to convert the analog signal produced by the photometer into a digital signal which is understandable by the computer. The converter accomplishes this by using a successive approximation register. When a program looks at a memory location tied to one of the A/D channels, the converter begins to make a digital approximation of the analog signal. Since the process of conversion is one of successive approximation, the longer the A/D is allowed to equilibrate the closer the digital value will be to the correct analog signal. In an assembly language program this is done by introducing several NOP commands. These commands have the effect of passing time without executing any instructions.

The A/D converter is able to convert any analog signal within plus five to minus five volts. If a signal outside this range is entered several consequences may follow. The converter will attempt to handle this overflow by passing the excess signal to another channel; this may have a diliterious effect on the conversion process. The overflow is passed to the channels immediately surrounding the overloaded channel. For example, in this interface project, we attempted to input ten volts to channel 13, as we monitored channel 14 we noticed that the digital signal from this channel was changing as we varied the input to channel 13. The second effect that overload can have is to burn out a channel.

While testing the A/D board for use in this interface we noticed that channel zero was not operating correctly: it was responding outside the specified plus to minus five volt range. Mountain Computer Corporation supplies several test programs along with its board. One of these programs is supposed to test the validity of any A/D channel. When this program was used to test channel zero it was found to be operating correctly. When channel zero was tested manually, it operated incorrectly. The apparent problem with the test programs is in the way they test each channel. Instead of physically applying a potential to the channel and testing the output of that same channel, they seem to place a value in the memory location assigned to that channel and then, later, look to see if that value has changed. Extreme care should be given in using the test programs supplied by the manufacturer. The best way to test the board is by manually applying a potential to a channel and checking the output of that channel with the computer.

As mentioned above, the A/D board can convert any analog signal within the range of minus five to plus five volts. This conversion is accomplished with good speed and accuracy. Nine microseconds are required for the conversion with an error of plus or minus one in the least significant bit. The board operates with eight bit registers in order to produce a digital output within the range of zero to 255 (see figure 4). For example, if we do a run on the stopped flow which sends a potential of 2.50 volts to the converter, the value 192 should be placed in memory.

Within this interface package two channels of the A/D board are used. One channel is for data (channel 14), and the second channel is used to mark the trigger (channel 13). The 16 different channels can be referenced by loading specific memory locations into the host program. These memory locations are dependant upon what slot the A/D board is plugged into in the computer. There is a simple formula which can be used to determine the correct memory location:

ADDR = 49280 + (slot# * 16) + channel#

FIGURE 4 - A/D Conversion Chart

## A/D CONVERSION CHART

| (OUTPUT) DAC | NUMBER | (INPUT) ADC |
|---|---|---|
| -5.00 V | 0 | -5.00 V |
| | . | |
| | . | |
| -2.50 V | 64 | -2.50 V |
| | . | |
| | . | |
| 0.00 V | 128 | 0.00 V |
| | . | |
| | . | |
| +2.50 V | 192 | +2.50 V |
| | . | |
| | . | |
| +5.00 V | 255 | +5.00 V |

The channel number can vary between zero and 15, the slot number varies between one and seven, and the desired address is ADDR. In this application the conversion board is in slot two, and, thus, channels 13 and 14 correspond to locations 49325 and 49326, respectively (in base sixteen these locations are $C0AD and $C0AE, respectively).

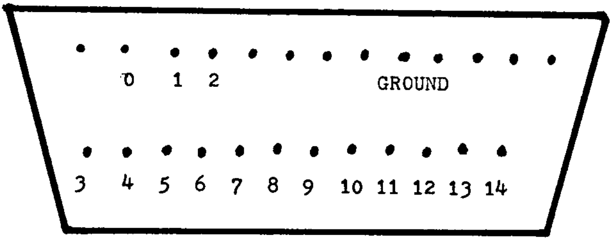
The final point concerning the A/D converter deals with the input and output of the unit. There are two cables supplied with the unit. One of these cables connects to the interface box while the other is left un-connected. The cables consist of sixteen wires, each representing one channel. The wires end in small pins used to connect the converter to other hardware. In the thesis entitled "Microcomputer Interfacing With Chemical Instrumentation", J. Meyer corrects the pin plan supplied by the manufacturer. These corrections are valid with one small change: channels zero and two should be switched; an updated pin plan is shown in figure 5.

We now have described how the data signal enters the computer. The next step in the hardware aspect of this package is data collection. Two hardware components are required for this. A clock is needed to regulate the interval for data collection, and a computer is needed to store and collect the data. The clock will be discussed in the next section and the Apple computer in the final section.

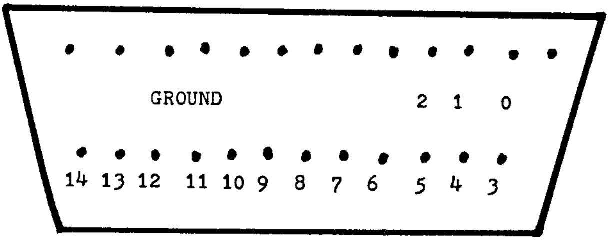
FIGURE 5 - Pin Plan for A/D Board

Pin Plan for A/D Board

DIGITAL TO ANALOG




ANALOG TO DIGITAL



4) APPLE CLOCK

When collecting kinetic data, it is important to have some knowledge of the time that a specific piece of data is recorded. The Apple clock allows us to keep track of time as we collect data during a reaction. Like the A/D converter, the clock is in one of the slots in the rear of the Apple. The clock, manufactured by Mountain Hardware, Inc. and fully assembled, can be used from both BASIC and Assembly language programs. Since we are interested in using the clock to regulate rapid data collection, we will use it from an Asssembly language program.

Before discussing how our program makes use of the clock, it is important to understand how the clock itself operates. The clock is controlled by clock counters; these counters simply count from zero to the next higher counter and then back again. These counters are regulated by a one mega-hertz crystal. The counters for time fractions of less than a second are in Binary Coded Decimal format (BCD) while the other counters are twelve bit binary counters and supply time values of over a second. The clock can be stopped and started from within a program by simply loading the appropriate memory location. Besides being able to keep track of time, the Apple clock also has some built in functions that allow the user access to the time displayed in certain formats. This is useful for applications requiring more than a few seconds; however, in our case, with the stopped flow, we are dealing with reactions that are over in a matter of seconds, and thus the built-in function is not applicable.

The clock can be read by simply loading the time from the correct memory location. Once again these memory locations depend on the slot in which the Apple clock is installed. On the Apple used in this interface package, the clock is in slot four. There is a simple formula which allows the user to calculate the correct memory locations based on the slot number:

$$ADDR = -16256 + (16*N) + X$$

Here X may have any value between zero and nine. The chosen value for X will determine what part of the clock is read (ex. four will contain the millisecond and tens of milliseconds time bits). The variable N is the slot number of the clock, and ADDR is the memory location containing the time information. Since the clock is used by the Assembly language program to regulate data collection, all future references to the clock's memory locations will be in base sixteen ( the number system used in Assembly language programming). Since the clock is in slot four, the range of usable memory locations is $C0C0 to $C0C6 (the remaining three locations are not used by the interface programs). Figure 6 shows what is contained in each of these locations.

Before data can be collected, the program must stop the clock. This has two significant effects, it stops the clock if it is on (has no effect if it is already stopped) and then sets all time bits less than a second to zero (this is done regardless of the previous status of the clock). Once we have stopped the clock and zeroed the proper locations, we may re-start the clock at any time. The interface package is designed to start the clock when the trigger from the stopped-flow is tripped. Once a reaction is started, the battery in the trigger circuit will send a voltage signal to the A/D board. The program monitors the voltage signal from a channel on the A/D board and starts the clock once the trigger line reaches the pre-set trigger level (which is 191), indicating the fact that a run has started. Two different locations on the clock are used for these tasks. The clock is stopped simply by executing the Assembly language command:

```
LDA $C0C6
```

The LDA command and its effect on the clock will be

FIGURE 6 - Clock Addresses

# APPLE CLOCK ADDRESS/COMMAND TABLE

ADDRESS

| HEX. | DECIMAL | COMMAND |
|---|---|---|
| C0C0 | -16192 | Reads $2^{20}$ - $2^{24}$ time bits |
| C0C1 | -16191 | Reads $2^{12}$ - $2^{19}$ time bits |
| C0C2 | -16190 | Reads $2^{4}$ - $2^{11}$ time bits |
| C0C3 | -16189 | Reads 100 msec. - $2^{3}$ time bits |
| C0C4 | -16188 | Reads 1 msec. - 10 msec. time bits |
| C0C5 | -16187 | Start Clock |
| C0C6 | -16186 | Stop Clock |

discussed in the SOFTWARE section of this paper. A similar command is used to start the clock. Once the clock has been started, we must be able to read the different time bits. We are only interested in time values of less than a second (locations $C0C3 and $C0C4). The time information within these locations is stored in BCD format. Once again, the time value handling will be reserved for the discussion of the Assembly language program.

During the development of the interface package, the clock was tested several times. The clock manual states that the crystal controlling the time on the clock board may at some point need re-calibration. The clock was tested by starting it at the same time as a dependable stop watch. After twenty four hours both were checked and found to agree within experimental error. We are now in a position to record data; all that remains to be examined is the role of the Apple computer in this interface package.

5) THE APPLE COMPUTER

The computer is an essential part of this interface package. The Apple computer has several significant parts that are useful for this interfacing project. Besides the video monitor and disk drive, the computer has several different graphics packages. A graphics package provides the user with the ability to draw pictures on the Apple screen. In both plotting programs of the interface, the high resolution screen is used. The high resolution screen has dimensions of 280 dots by 192; dots, each dot represents a point on the screen. This allows for a relatively sharper picture than the low resolution screen on the Apple.

The Apple is programmable in several different languages. The "home" language of the Apple is a form of BASIC called APPLESOFT. Applesoft is very similar to Basic except for some minor variations that make the Apple more useful. Most of the interface programs are written in Applesoft with the exception of the data collection program which is written in Assembly language. There is one other language that is specific to the Apple computer: DOS. This stands for disk operating system and is the part of the Apple that keeps track of what is in memory, what files are on the disk, and what the user wants to do with the current file. An example of DOS is the DELETE or BLOAD commands. The delete command allows the user to remove a file from the disk while the bload command loads from a disk a given binary file. This command is used in the BASIC portion of the data collection program to load the Assembly language program. Once the program is loaded, it is placed into active memory to be executed later in the interface process.

The interface program is intended for use with some sort of printing device. As of now the Epson MX-80 printer is the one programmed for in the interface. If, in the future, other printers are desired, there should be no problem in incorporating them.

Up to this point we have examined all of the pieces necessary for a successful interface. We have provided a basic understanding of what occurs when one runs a reaction on the stopped flow. With this information, we can now move onto the software section. This section will focus on the several different programs developed for the stopped flow interface.

## II. SOFTWARE

The second half of this report will deal with the software, or computer programs, written for the stopped flow interface package. These programs can be broken down into several different categories. The first three programs are used to initialize the system, load the menu program, and set the proper voltages on the stopped flow instrument. The second set of programs are designed to allow the user to record a run on the stopped flow and permanently save it on a floppy disk. The final set of programs are data manipulation programs. These programs include the plotting programs and the first order kinetic calculations.

### 1) INITIALIZATION PROGRAMS

When the user first turns on the computer with the interface disc in drive number one, it executes several programs stored permanently within the Apple. These programs set the different parameters necessary for interaction with the user. The final initialization program run by the computer is the program entitled HELLO. The 'hello' program, then, is the first program in the stopped flow interface package (see figure 7). Although this program is short and straight forward, it illustrates a very useful technique essential to this interface package. Applesoft Basic includes a set of instructions that allow the user to execute certain control commands from within a program. This command takes the form :

CHR$(4)

The CHR$(X) function has the effect of returning the ASCII character that corresponds to the variable or number placed within parentheses. In this case we use this function to return the ASCII code equivalent of the number four. The number four in the ASCII code represents control-D. By typing a line such as line 70 in figure 7, we are able to execute the string as a direct command to the operating system. This technique of issuing operating system commands from within a program is very useful. In this interface package, some of the operating system commands used are : RUN, BLOAD, OPEN, and CLOSE. These commands will be discussed in the appropriate section.

Since the hello program is the first program executed by the computer, using the CHR$ command allows us to call any other program on the disk. The program called by the

FIGURE 7 - Hello Program

```
]LIST

10  REM  **********************************
11  REM  * PROGRAM : HELLO               *
20  REM  * HELLO PROGRAM FOR STOPPED     *
30  REM  * FLOW INTERFACE.               *
40  REM  **********************************
50  REM  THIS PROGRAM IS EXECUTED WHEN THE COMPUTER IS TURNED ON.
60  REM  ITS PURPOSE IS TO CALL THE FIRST MENU IN THE INTERFACE.
70  PRINT  CHR$ (4);"RUN SF-INTFCE"
80  END

]
```

hello program is the first menu program called SF-INTFCE (see figure 8). This is the first of two menus included in the package.

From this menu the user can call any part of the system. The key to this program is the use of a command that allows a different program to be run depending upon the user's choice. The program first prints the four possible choices on the screen followed by a prompt soliciting the user's choice. At this point two things can happen. If the user inputs a number not corresponding to one of the four choices, the question will be repeated. If the user enters a number from one to four, the program of his choice will be executed. At this point the user should enter the number one for the set-up program.

The SET-UP program (see figure 9) is designed to allow the user to eliminate any voltage difference between the pmt and the A/D board. As discussed earlier, the A/D board has certain memory locations associated with it. By executing the Applesoft command PEEK(X) we can examine the result of any analog to digital conversion (line 80). For example, in line 80 the setup program makes use of this command to display the voltage difference between the pmt and the A/D board. In order to display a voltage value between zero and ten volts, the program must convert the digital result of the conversion into the appropriate value (line 90). Line 80 reads the

FIGURE 8 - SF-INTFCE Program

```
]LIST

0  REM  **********************************
1  REM  * PROGRAM : SF-INTFCE            *
2  REM  * FIRST MENU IN STOPPED FLOW     *
3  REM  *          INTERFACE             *
4  REM  **********************************
5  REM  THIS PROGRAM CONTAINS THE PRIMARY MENU OF THE INTERFACE
20 D$ =  CHR$ (4): REM  THIS ALLOWS FOR USE OF CTRL-D FROM WITHIN THE PRO
    GRAM.
30  HOME : REM  CLEAR SCREEN
40  HTAB 5: PRINT "MENU FOR STOPPED FLOW INTERFACE"
50  PRINT : PRINT
60  INVERSE : REM  PRINT DARK LETTERS ON LIGHT BACKGROUND
70  PRINT "1.SET UP"
80  PRINT : PRINT
90  PRINT "2.DATA COLLECTION"
100  PRINT : PRINT
110  PRINT "3.MANIPULATE DATA"
120  PRINT : PRINT
130  PRINT "4.QUIT"
140  PRINT : PRINT
150  NORMAL : REM  PRINT LIGHT LETTERS ON A DARK BACKGROUND
180  PRINT : PRINT : PRINT
190  INPUT "ENTER A NUMBER AND PRESS RETURN. ";NB
200  PRINT
210  ON NB GOSUB 260,280,300
220  IF NB = 4 GOTO 320
250  GOTO 30
260  PRINT D$;"RUN SETUP": REM  EXECUTE INITIALIZATION PROGRAM
270  RETURN
280  PRINT D$;"RUN CONTROLLER": REM  EXECUTE DATA ACQUISITION PROGRAM
290  RETURN
300  PRINT D$;"RUN MENU2": REM  EXECUTE DATA MANIPULATION PROGRAMS
310  RETURN
320  END

]
```

FIGURE 9 - SET UP Program

```
]

 LIST

0  REM  ***********************************
1  REM  * PROGRAM : SETUP                 *
2  REM  * SET UP PROGRAM OF STOPPED       *
3  REM  * FLOW INTERFACE PACKAGE.         *
4  REM  ***********************************
5  REM  THIS PROGRAM IS CALLED FROM THE FIRST MENU.
6  REM  IT IS DESIGNED TO READ THE DATA CHANNEL OF THE A/D BOARD
7  REM  AND DISPLAY A VOLTAGE WITHIN THE LIMITATIONS OF THE BOARD
8  REM  THE PROGRAM IS TERMINATED WHEN THE USER PRESSES ANY
9  REM  KEY FROM THE KEYBOARD. THE PROGRAM USES LOCATION -16384 TO SEE IF

10  REM  KEY HAS BEEN PRESSED, THIS FUNCTION IS RESET BY POKING -16368
20 D$ =  CHR$ (4): REM  CNTRL-D
30  HOME
40  PRINT : PRINT
41  HTAB 12: PRINT "SET-UP PROCEDURE": PRINT
42  PRINT : PRINT
50  PRINT "PROCEDURE TO COUPLE COMPUTER TO PMT."
60  PRINT : PRINT "  INCREASE OFFSET ON PHOTO-"
70  PRINT "    METER UNTIL VOLTAGE ON COMPUTER"
75  PRINT "          IS 0 TO .1 VOLTS (FLASHING)."
77  REM ----------READ A/D DATA LINE----------
80 X =  PEEK (49326): REM  THIS IS CHANNEL 14 SLOT 2
90 X = X * (10 / 255): REM  ALLOW X TO VARY FROM 0 TO 10
91  REM  PRINT ONLY 3 DIGITS OF VOLTAGE USING STRING COMMANDS
100 X$ =  STR$ (X)
109  INVERSE
110  VTAB 17: HTAB 10: PRINT "   VOLTAGE : "; LEFT$ (X$,3);"   "
111  VTAB 17: PRINT
112  NORMAL
113  VTAB 23: PRINT "PRESS SPACE BAR TO RETURN TO MENU"
115  REM  CHECK TO SEE IF KEY HAS BEEN PRESSED
120 Y =  PEEK ( - 16384)
121  POKE  - 16368,0
130  IF Y `  = 127 GOTO 80
140  PRINT D$;"RUN SF-INTFCE"
150  END : REM  RETURN TO MENU 1 WHEN DONE

]
```

value from the board using the PEEK command while line 90 makes the conversion. The value must be divided by 255 since that is the maximum value the A/D board will produce; this result must then be multiplied by ten to place it within the desired ten volt range. The program loops until a key is depressed on the keyboard. At this time the program executes the operating system command; RUN SF-INTFCE, which has the effect of returning the user to the first menu.

The code central to the functioning of this program can be found in lines 120 and 121. Two locations in memory are used in these lines. Location -16384 is a flag that is set any time a key is depressed on the keyboard. The second location, -16368; has the opposite effect, since it re-sets the computer's ability to determine whether or not a key has been pressed. It is important to re-set location -16384 after using it by POKING a zero into location -16368. The poke command, like the peek command allows the user direct access to memory locations. The poke command allows the user to set a specific location to any desired value by poking that location with the desired value. Once again, after the user has completed the set up program he is returned to the first menu.

## 2) DATA COLLECTION

Once all of the necessary initialization has been completed, the user may record data from the stopped flow instrument. Entering the number two from the main menu will call the program written for this purpose. The program entitled CONTROLLER (see figure 10) will be loaded in place of the menu program. This program has three main functions : to determine the run length desired by the user, to determine the amount of time to delay before taking an infinity reading, and, finally, to load and execute the Assembly language program that actually collects the reaction data. Each of these three tasks will be discussed separately below.

In lines 50-54 the user is asked to enter the desired run time. This value is then divided by 250 (the number of data points to be taken). The result of this division is the interval time between data points, in other words, the amount of time the computer must wait before recording the next reading. This variable can have a maximum value of 0.999 msec., and thus the longest run time allowed by this package is 249 (250 * 0.999) seconds. This should be more than adequate for applications requiring the stopped flow instrument. Once a value for the interval length has been determined, it

FIGURE 10 - CONTROLLER Program

```
]LIST 0-214

0   REM  ************************************
1   REM  * PROGRAM : CONTROLLER             *
2   REM  * BASIC CONTROL PROGRAM FOR        *
3   REM  * TAKING DATA WITH ASSEMBLY        *
4   REM  *      LANGUAGE PROGRAM            *
5   REM  ************************************
6   REM  THIS PROGRAM SETS THE RUN TIME OVER WHICH DATA
7   REM  SHOULD BE TAKEN AND THE AMOUNT OF DELAY TIME
8   REM  BEFORE THE INFINITY READING IS RECORDED
9   REM  ONCE THIS INFORMATION IS SOLICITED FROM THE USER
10  REM  IT IS PASSED ONTO THE ASSEMBLY LANGUAGE PROGRAM
11  REM  VARIABLES:LE=length of run
12  REM            D =amount of time to delay
13  REM            IVL=time interval between points
14  REM            HUND=hundreds digit of delay time
15  REM            TENS=tens digit of delay time
16  REM            O=ones digit of delay time
17  REM            IVF=infinity value from A/D board
18  REM            SCRAP=file to save interval length
19  REM  ALL VALUES ARE poked INTO LOCATIONS LINKED TO ASSEMBLY PGM.
20  HOME
30  HTAB 12: PRINT "DATA ACQUISITION PROGRAM"
40  PRINT : PRINT
45  REM  ----------GET RUN TIME----------
50  PRINT "ENTER LENGTH OF RUN IN SECONDS"
51  INPUT "  (A MULTIPLE OF .250 SEC.) : ";LE
52  REM  IF USER ENTERS INVALID RUN TIME, RE-ASK QUESTION
54  IF LE ` .250 GOTO 20
60  HOME
65  REM  ----------GET INFINITY DELAY TIME----------
70  PRINT "INFINITY VALUE OPTION"
71  PRINT : PRINT : PRINT
80  HTAB 4: PRINT "1.ENTER 0 FOR MANUAL INFINITY VALUE"
90  PRINT : PRINT
100  HTAB 4: PRINT "2.ENTER DELAY TIME (1-999 MILLISEC.)"
110  PRINT : PRINT : PRINT
120  INPUT "INPUT DESIRED TIME (0-999): ";D
121 D = D / 1000: REM  CHANGE DELAY TO MSEC
170  REM  DIVIDE INTERVAL INTO PROPER LOCATIONS.
180 IVL = LE / 250: REM  DIVIDE RUN LENGTH EQUALLY BETWEEN 250 POINTS
181  IF (IVL * 1000 -  INT (IVL * 1000)) ~ .5 THEN IVL = IVL + .001
190 HUND =  INT (IVL * 10)
200 TENS =  INT (((IVL * 10) - HUND) * 10)
210 O =  INT (((((IVL * 10) - HUND) * 10) - TENS) * 10)
212 IVL = (HUND * 100 + TENS * 10 + O) / 1000
213 LVI = IVL
214 IVL = IVL * 1000

]
```

```
LIST 214-400

214 IVL = IVL * 1000
215  REM  PUT INTERVAL LENGTH IN LOCATION $303
220  POKE 771,IVL: REM  771=$303
221  REM  SEPARATE DELAY TIME AND poke INTO APPROPRIATE LOCATIONS
251 HUND =  INT (D * 10)
252 TENS =  INT (((D * 10) - HUND) * 10)
253 O =  INT (((((D * 10) - HUND) * 10) - TENS) * 10)
254  POKE 775,HUND: REM  775=$307
255  POKE 776,TENS: REM  776=$308
256  POKE 777,O: REM  777=$309
300  HOME : FLASH : PRINT "START RUN WHEN RED LIGHT GOES OFF": NORMAL
305  REM  ----------LOAD ASSEMBLY PGM. AND COLLECT DATA----------
306 D$ =  CHR$ (4)
310  PRINT D$;"BLOAD READER1.OBJ0"
320  CALL 778: REM  COLLECT DATA
321  IF D ˜ 0 GOTO 332: HOME
322  REM  ----------TAKE INFINITY VALUE----------
323  HTAB 4: VTAB 12: PRINT "PRESS ANY KEY TO TAKE INFINITY VALUE"
324  GET G$
325 IVF =  PEEK (49326): REM  DO DUMMY READ OF A/D BOARD AND MAKE AVERAGE
    .
326 IVF = 0
327  FOR X = 1 TO 10
328 IVF = IVF +  PEEK (49326)
329  NEXT X
330 IVF = IVF / 10
331  POKE 24826,IVF: REM  POKE INFINITY VALUE
332  PRINT D$;"OPEN SCRAP"
340  PRINT D$;"DELETE SCRAP"
350  PRINT D$;"OPEN SCRAP"
360  PRINT D$;"WRITE SCRAP"
370  PRINT LVI
380  PRINT D$;"CLOSE SCRAP"
390  PRINT D$;"RUN MENU2"
400  END : REM  END PROGRAM AND RETURN TO MENU

]
```

must be communicated to the assembly language program. This is accomplished with the help of the poke command discussed earlier. By simply poking the interval length into a pre-determined memory location, that value is stored for use by the assembly language program.

The second part of the controller program deals with the amount of time to delay before taking an infinity reading. Lines 65-121 solicit this information. The user has two choices in taking an infinity reading. He may enter any number from 1 to 999 which will signify the number of milliseconds to delay after the end of the run before taking the infinity value; or he may enter the number 0 to record the infinity value manually. If the user wants to record a manual infinity value, the computer will prompt him to do so at the end of the run. Once the user enters a value for the delay time, the computer immediately divides this number by 1000 in order to convert it into a millisecond value. Lines 221-253 are necessary to divide the delay time into the proper units. By making use of the INT command (which effectively erases the decimal portion of a number) we can separate the delay time into units of hundreds, tens, and ones of milliseconds. These steps are necessary for the correct functioning of the assembly language program. If the user enters zero for a delay time (signifying a desire to record a manual infinity value) the program will place three zeros into the locations assigned to the delay time causing the assembly language program to take an infinity reading immediately, only to have this value replaced by the user in the controller program.

After the run has been completed, the controller takes care of a housekeeping matter. Lines 332-380 store the interval length in a file entitled SCRAP. This allows the assignment of the correct time to the data points stored in the computer. The SCRAP file will be read in the saving program (data storage), and will be used for a different purpose later on in the interface package.

The commands that cause the assembly language program to be executed can be found in lines 310 and 320. The first line makes use of the ability to execute an operating system command from within an Applesoft program. The command BLOAD READER1.OBJ0 causes the computer to load the binary file READER1.OBJ0 starting in location $30A. This command only loads the file; it does not execute it. Line 320 executes the file with the command CALL 778. This Applesoft command causes the computer to move to the given memory location and begin executing the program at that location. The hexadecimal (base sixteen) number $30A corresponds to 778 in base ten. Thus, by calling location 778, we are actually executing the assembly language data collection program.

The assembly language program (see figure 11) can also be broken down into several sections. The first section causes the computer to wait for the stopped flow

FIGURE 11 - READER1.OBJO Program

LIST 1-61

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;         PROGRAM :        READER1.OBJ0     ;
;         ASSEMBLY-LANGUAGE-PROGRAM-FOR     ;
;         STOPPED-FLOW-INTERFACE-PACKAGE ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;         PROGRAM LOADED AT $30A, ENDS AT 3E7.
          ORG  $30A        ;PROGRAM STARTS AT $30A
          LDA  $C0C6       ;STOP THE CLOCK
          LDA  #$BF        ;TRIGGER LEVEL IS 191
          STA  $302        ;STORE TRIGGER LEVEL IN $302
;----------TRIGGER-WAIT-LOOP----------
          LDA  $C0AD       ;LOAD TRIGGER VALUE FROM A/D
TOP1      LDA  $C0AD       ;LOAD TRIGGER VALUE(TRUE) FROM A/D
          CMP  $302        ;COMPARE VALUE WITH TRIGGER LEVEL
          BCC  TOP1
          LDX  #00         ;INITIALIZE POINT COUNTER TO ZERO
          LDY  #250        ;INITIALIZE MAXIMUM NUMBER OF POINTS
          LDA  #00         ;ZERO ACCUMULATOR
          STA  $304        ;ZERO LOCATION $304(HUND DIGIT)
          STA  $305        ;ZERO L OCATION $305(TENS DIGIT)
          STA  $306        ;ZERO LOCATION $306(ONES DIGIT)
          STA  $302        ;ZERO LOCATION $302(SCRAP LOCATION)
          LDA  $C0C5       ;START THE CLOCK
;----------DATA-POINT-READING-LOOP----------
TOP       LDA  $C0AE       ;DUMMY READ OF A/D DATA LINE
;ALLOW-A/D-TO-REACH-CORRECT-VALUE
          NOP
          NOP
          NOP
          NOP
          NOP
          LDA  $C0AE       ;REAL READING OF A/D DATA LINE
          STA  $6000,X     ;STORE POINTS STARTING AT LOCATION $6000
          CLC
;INCREASE-TIME-TARGET-BY-INTERVAL-LENGTH
          SED              ;SET THE DECIMAL MODE (BCD)
          LDA  $306        ;LOAD ONES DIGIT
          ADC  $303        ;ADD INTERVAL LENGTH STORED IN $303
          STA  $302        ;STORE IN SCRAP LOCATION
          AND  #$0F        ;ERASE UPPER NIBBLE OF ACCUMULATOR
          STA  $306        ;STORE NEW ONES DIGIT
;SHIFT-ANY-CARRY-TO-RIGHT-BIT-OF-LOCATION
          LSR  $302
          LSR  $302
          LSR  $302
          LSR  $302
          CLC
          LDA  $305        ;LOAD TENS DIGIT
          ADC  $302        ;ADD ANY CARRY FROM PREVIOUS ADDITION
          STA  $302        ;STORE IN SCRAP LOCATION
          AND  #$0F        ;ERASE UPPER NIBBLE OF ACCUMULATOR
          STA  $305        ;STORE TENS DIGIT
;SHIFT-ANY-CARRY-TO-RIGHT-BIT-OF-LOCATION
          LSR  $302
          LSR  $302
          LSR  $302
          LSR  $302
          CLC
          LDA  $304        ;LOAD HUNDREDS DIGIT
          ADC  $302        ;ADD ANY CARRY FROM PREVIOUS ADDITION
          AND  #$0F        ;ERASE UPPER NIBBLE OF ACCUMULATOR
```

LIST 62-117

```
            STA   $304         ;STORE NEW HUNDREDS DIGIT
            CLD                ;CLEAR DECIMAL MODE
;ALLOW-CLOCK-TO-CATCH-UP-TO-INTERVAL
BOT         LDA   $C0C3        ;LOAD HUNDREDS TIME BIT
            AND   #$0F         ;ISOLATE HUND TIME BIT
            CMP   $304         ;COMPARE CLOCK TO HUND VALUE
            BNE   BOT          ;READ CLOCK AGAIN IF CLOCK IS SMALLER
            LDA   $C0C4        ;READ TENS AND ONES MSEC DIGITS OF CLOCK
            STA   $302         ;SAVE TIME IN SCRAP LOCATION
            AND   #240         ;ISOLATE TENS OF MSEC VALUE
;MOVE-10MSEC-VALUE-TO-LOWER-NIBBLE
            LSR   A
            LSR   A
            LSR   A
            LSR   A
            CMP   $305         ;COMPARE CLOCK WITH INTERVAL LENGTH
            BCC   BOT          ;READ CLOCK AGAIN IF CLOCK IS SMALLER
            LDA   $302         ;LOAD PREVIOUS TIME FROM SCRAP LOCATION
            AND   #$0F         ;ISOLATE ONES OF MSEC DIGIT
            CMP   $306         ;COMPARE CLOCK WITH INTERVAL LENGTH
            BCC   BOT          ;READ CLOCK AGAIN IF CLOCK IS SMALLER
            INX                ;INCREMENT NO. OF POINTS COUNTER
            DEY                ;DECREMENT NO. OF POINTS ALREADY READ
            BNE   TOP          ;READ ANOTHER POINT UNTIL ALL 250 ARE READ
;----------TAKE-INFINITY-VALUE----------
            LDA   $C0C6        ;STOP CLOCK
            LDA   $C0C5        ;START CLOCK
BOT1        LDA   $C0C3        ;LOAD HUND OF MSEC DIGIT
            AND   #$0F         ;SAVE LOWER NIBBLE
            CMP   $307         ;COMPARE TO HUND MSEC OF DELAY INTERVAL
            BCC   BOT1         ;READ CLOCK UNTIL CLOCK IS LARGER
            LDA   $C0C4        ;READ TENS AND ONE MSEC TIME BITS
            STA   $302         ;SAVE TIME IN SCRAP LOCATION
            AND   #240         ;ISOLATE UPPER NIBBLE
;MOVE-10-MSEC-VALUE-TO-LOWER-NIBBLE
            LSR   A
            LSR   A
            LSR   A
            LSR   A
            CMP   $308         ;COMPARE TIME WITH TENS MSEC DELAY DIGIT
            BCC   BOT1         ;BRANCH TO BOT1 IF TIME IS SMALLER
            LDA   $302         ;LOAD TIME FROM SCRAP LOCATION
            AND   #$0F         ;ISOLATE ONES OF MSEC DIGIT
            CMP   $309         ;COMPARE TIME WITH ONES MSEC DELAY DIGIT
            BCC   BOT1         ;BRANCH TO BOT1 IF TIME IS SMALLER
            LDA   $C0AE        ;DUMMY READ OF A/D LINE
;ALLOW-A/D-TO-REACH-CORRECT-VALUE
            NOP
            NOP
            NOP
            NOP
            NOP
            LDA   $C0AE        ;TRUE READING OF A/D LINE-INFINITY VALUE
            STA   $60FA        ;STORE VALUE AFTER 250 DATA POINTS
            RTS
            BRK                ;END OF PROGRAM
```

:

trigger to be set. The second section records the data points from the run, and the final section records the infinity reading.

The trigger wait loop is the first section to be executed by the computer. Before starting this loop the trigger level (digital value of 191) is stored in location $302. This is done so that the computer does not trigger immediately once the program is executed, but instead waits until the stopped flow is started. The A/D board is then read (lines 10 and 11). The first time it is read is actually a dummy read to allow equilibration of the conversion process. The second LDA command loads the accumulator with the digital representation of the trigger potential (channel 13 slot 2). If this value is equal to or less than 191 (the trigger level, about 2.50 volts) the computer goes back to the top of the loop and reads the A/D board again. This process continues until the A/D board produces a value above 191. This can only occur when the trigger level goes above the target voltage signifying the begining of a run. Once the trigger has been set, the computer makes some preliminary initializations and then moves into the data collection section of the program.

The data collecting loop begins with line 23. Once again, a dummy reading of the A/D board is made, and five NOP commands are used to allow the converter to equilibrate before the actual reading of the board is made. The A/D is read by the instruction LDA $C0AE. This memory location corresponds to channel 14 of slot two. Immediately after the data are read they are stored in the set of memory locations reserved for data points. The first data point is stored in location $6000 (decimal 24571). All the other points follow it with the infinity reading (point 251) being stored at the end. The actual location is calculated (line 31) by adding a constant 6000 to the point counter (index registor X).

After storing a data point, we must increment the time interval target by the interval length. This is most easily done by executing the command SED (set decimal mode). The advantage of this mode over the normal binary mode is the fact that each of the two halves of a memory location (nibble) can only contain values from zero to nine. Thus it is possible to add the interval length, which may be four msec. (if the entire run time is one minute), to the memory location that contains the ones digit of the time target. This addition will be done in base ten causing a carry to the next higher position if the result is ten or greater (the entire process is diagrammed in figure 12). The ability to add the interval length in the decimal mode allows for a much more concise and efficient program. Once the time target has been set, the clock must be read in order to determine whether or not it is time to take another data reading. In most cases by the time the computer reads the clock, there

**FIGURE 12 - Addition in Decimal Mode**

# ADDITION IN DECIMAL MODE

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $303 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | (4) |
| $305 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | (8) |
| $306 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | (8) |

Accumulator

| Instruction | Register | | | | | | | | | | Memory | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | A | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | (0) | | | | | | | | | | |
| LDA $306 | A | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | (8) | | | | | | | | | | |
| ADC $303 | A | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | (12) | | | | | | | | | | |
| STA $302 | A | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | | $302 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | (12) |
| AND #$0F | A | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | (2) | | | | | | | | | | |
| STA $306 | A | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | | $306 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | (2) |
| LSR $302 four times | A | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | | $302 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | |
| LDA $305 | A | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | (8) | | | | | | | | | | |
| ADC $302 | A | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | (9) | | | | | | | | | | |
| STA $302 | A | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | | $302 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | |

will be quite a long time before it is time to take another reading. This is the purpose of the clock catch up loops. There are three of these loops in this section. The first loop waits for the hundreds of milliseconds target digit to match the clock's time: after these have matched the computer moves into the second loop and waits for the tens of milliseconds time digit to match the clock's tens of milliseconds digit. The final loop is the same except that it waits for the single milliseconds readings to match. Once the computer falls out of the last loop, all of the time digits have matched and the clock has caught up to the target time. It is now time to record another data point. Before the next point is recorded, the appropriate counters must be updated and set. This data collection loop continues for all 250 points.

After the last point is recorded, the clock is stopped, and then started again before moving into the infinity delay loop. This loop is identical to the loops described above except for the fact that it is only executed once. After the clock matches the delay time entered by the user, a reading is made of the A/D board and an average is taken for the infinity reading. This value is stored in location $60FA which is 251 locations after $6000. At this point all the data have been taken and the flow of control returns to the calling program, in this case the Applesoft controller program.

If the user entered a value of zero for the infinity delay, he is now prompted to indicate the desired time to record a value. Once the user does this, the A/D board is read and an average reading is stored in location 49326 (this is the same location as $60FA). This section is skipped if the user entered a non zero delay time. As stated earlier, the controller program finally stores information in the housekeeping file and then executes MENU2.

This program is similar to the SF-INTFCE program except that it contains four different choices (figure 13). Now that the reaction data are stored in the computer thet must be transferred to the floppy disk for permanent storage. This is accomplished by pressing the number one from the second menu. A program entitled SAVER (see figure 14) is executed.

This program has one task: to save the data of the run on the floppy disk. Each data point has two parts. The first part is the time at which a certain piece of data was recorded and the second part is the actual data reading. In order to assign a specific time to the data, this program must know what the interval length was. Thus, the first section of the saver program retrieves the interval length stored in the SCRAP file by the controller program. Once the interval length is known, it is possible to calculate the time associated with each specific data point by simply multiplying the interval by

FIGURE 13 - MENU2 Program

```
]LIST

0  REM  ***********************************
1  REM  * PROGRAM : MENU2                 *
2  REM  * MENU FOR DATA MANIPULATION      *
3  REM  *          PROGRAMS               *
4  REM  ***********************************
5  REM  SECOND MENU FOR STOPPED FLOW INTERFACE PACKAGE-DATA MANIPULATION
6  REM  FOLLOWS SAME PRINCIPLES AS FIRST MENU IN PACKAGE
7  REM  CHOICES ARE PRINTED IN inverse AND LATER SET BACK TO normal.
20 D$ =  CHR$ (4): HOME
40  HTAB 5: PRINT "MANIPULATION OF DATA PROGRAMS"
50  PRINT : PRINT : INVERSE
90  PRINT "1.STORE NEW DATA"
100  PRINT : PRINT
110  PRINT "2.PLOT RAW DATA"
120  PRINT : PRINT
130  PRINT "3.CALCULATE FIRST ORDER RATE CONSTANT"
140  PRINT : PRINT
150  PRINT "4.GO BACK TO MENU 1"
160  PRINT : PRINT
170  PRINT "5.QUIT"
180  PRINT : PRINT : PRINT
190  NORMAL
200  INPUT "ENTER A NUMBER AND PRESS RETURN. ";NUM
210  ON NUM GOSUB 250,270,290,300
211  IF NUM = 5 GOTO 320
220  GOTO 20
250  PRINT D$,"RUN SAVER": REM  PROGRAM TO SAVE DATA
260  RETURN
270  PRINT D$,"RUN RPLOT": REM  PROGRAM TO MAKE ROUGH PLOT OF DATA
280  RETURN
290  PRINT D$,"RUN CALC": REM  PROGRAM TO CALCULATE RATE CONSTANT(1st ORD
     ER)
295  RETURN
300  PRINT D$,"RUN SF-INTFCE": REM  RETURN TO FIRST MENU
320  END : REM  IF NONE OF CHOICES ABOVE END EXECUTION OF INTERFACE PACKA
     GE.

]
```

FIGURE 14 - SAVER Program

```
]LIST

0   REM  ***********************************
1   REM  * PROGRAM : SAVER                 *
2   REM  * PROGRAM TO SAVE DATA ON DISK    *
3   REM  * DRIVE AFTER COMPLETION OF RUN   *
4   REM  ***********************************
5   REM  THIS PROGRAM USES A FILE(SCRAP) TO REMEMBER THE INTERVAL(IVL)
6   REM  LENGTH OF THE RUN.AFTER THIS PROGRAM IS COMPLETED THE USER IS
7   REM  RETURNED TO THE SECOND MENU.
10  REM  NOTE:line 12 eliminates messages on I/O status.
11 D$ =  CHR$ (4)
12  PRINT D$;"NOMON C,I,O"
21  PRINT D$;"OPEN SCRAP"
22  PRINT D$;"READ SCRAP"
23  INPUT IVL: REM  GET LENGTH OF INTERVAL BETWEEN POINTS STORED IN SCRAP

24  PRINT D$;"CLOSE SCRAP"
30  HOME
35  REM  USER SHOULD NOW INPUT FILE NAME BY WHICH DATA WILL BE STORED
40  INPUT "INPUT FILE NAME TO SAVE DATA : ";N$
49  PRINT "SAVING: ",N$
50  PRINT D$;"OPEN ",N$
70  PRINT D$;"WRITE ",N$
80  FOR X = 1 TO 251
89  REM  DATA STARTS IN LOCATION 24575 AND CONTINUES FOR 250 LOCATIONS
90  PRINT IVL * (X - 1), PEEK (24575 + X)
95  NEXT X
99  REM  AT END OF PROGRAM USER IS RETURNED TO SECOND MENU
100  PRINT D$;"CLOSE ",N$
110  PRINT D$;"RUN MENU2"
115  END

]
```

the data point number minus one. Before the data are stored, the user is asked to specify the file name under which the data will be saved on the disk (line 40). Once this is done, the file is opened and the 250 data points, along with the times in the run they were taken, are saved. These data points are followed by the infinity reading on the disk. This is the end of the SAVER program, and control is again passed back to a menu, in this case, the second menu containing the data manipulation programs.

3) DATA MANIPULATION

This interface package includes two programs for data manipulation. The first program simply makes a plot on the screen of the user specified data file, while the second program calculates a first order rate constant and makes the appropriate plot.

The program entitled RPLOT (see figure 15) is used to make a rough plot of the data. The program is called from the data manipulation menu. This program has a very interesting feature. Not only does it use the high resolution screen of the Apple computer, but it also makes use of another program which allows letters and numbers to be printed on that screen. This program is called the High Resolution Character Generator (HRCG) and is called immediately after entering RPLOT. Once the HRCG is loaded into the Apples's memory and executed (line 10000), the user is asked to specify which file he would like printed (line 900). Once the data have been read into the computer, the screen is cleared and the axes are drawn and labeled (the task of labeling the axis would have been much more complex without the HRCG program). Lines 210 through 292 draw the actual plot with the horizontal infinity line printed at the end. The data from the file must be converted into percent transmitance

FIGURE 15 - RPLOT Program

LIST 0-360

```
0   REM  **********************************
1   REM  * PROGRAM : RPLOT                *
2   REM  * PROGRAM TO MAKE A ROUGH PLOT   *
3   REM  * OF DATA TAKEN USING THE        *
4   REM  * STOPPED FLOW INTERFACE PACKAGE*
5   REM  **********************************
6   REM  VARIABLES : TIME=time data point was taken
7   REM              DT=actual data point, on a scale of 0 to 255
8   REM              IVL=actual infinity value reading
9   REM              N$=name of data file
10   REM             LEHI=length of interval between data points
11   REM  THIS PROGRAM READS A DATA FILE AND PRINTS A PLOT OF
12   REM  TRANSMIT.vs.TIME FOR THE DATA. FILE NAME AND INTERVAL LENGTH ARE

13   REM  ALSO INCLUDED ON THE DISPLAY. THE USER HAS THE ABILITY
14   REM  TO PRINT A HARD COPY OF THE PLOT IF SO DESIRED.
15   REM  DIMENSION ARRAYS TO NUMBER OF DATA POINT PLUS INFINITY READING
16   DIM TIME(251)
17   DIM DT(251)
20 D$ =  CHR$ (4)
25   REM  ----------LOAD HRCG----------
26   REM  USE HIGH RES.CHAR.GEN. TO PRINT ALPHANUMERICS ON GRAPHIC PAGE
30   GOSUB 1000: REM  LOAD HRCG
40   HOME
60   REM  ----------PLOT X + Y AXIS----------
70   HPLOT 28,21 TO 28,171 TO 278,171
80   REM  MARKS OFF AXIS BY FIVES
90   FOR I = 21 TO 171 STEP 15
100   FOR M = 28 TO 278 STEP 10
110   HPLOT M,I
120   NEXT M
130   NEXT I
140   REM  PLOT X HATCH MARKS
150   FOR L = 28 TO 278 STEP 10
160   HPLOT L,171 TO L,176
170   NEXT L
210   REM  ----------PLOT DATA----------
211   REM  PLOT START AT LOCATION 28,21 ON SCREEN
220 Y1 = (100 * (DT(1) / 255) * 3 / 2) + 21
230 X1 = 28
240   FOR R = 2 TO 250
250 Y2 = (100 * (DT(R) / 255) * 3 / 2) + 21
260 X2 = R + 28
270   HPLOT X1,Y1 TO X2,Y2
280 X1 = X2:Y1 = Y2
290   NEXT R
291 IVL = (100 * (DT(251) / 255) * 3 / 2) + 21
292   HPLOT 28,IVL TO 278,IVL
299   REM  ----------SEE IF USER WANTS HARD COPY---------
300   GET G$
301   PRINT  CHR$ (15); CHR$ (2)
310   HOME : TEXT
311 D$ =  CHR$ (4)
320   VTAB 12: HTAB 8: INPUT "WOULD YOU LIKE A HARD COPY ? :";Y$
330   ON Y$ = "YES" GOSUB 350
331   POKE ADRS + 10,0: POKE ADRS + 11,198
332   PRINT  CHR$ (15); CHR$ (25)
340   PRINT D$;"RUN MENU2"
```

```
LIST 340-1120

340  PRINT D$;"RUN MENU2"
350  PRINT D$,"RUN EPSON PLOT"
360  RETURN
900  REM  LOADS DESIRED DATA FILE FOR PLOT
910  HOME
940  INPUT "ENTER DESIRED DATA FILE. ";N$
941 A$ = N$
950  PRINT CK$;"LOADING DATA IN FILE : ";N$
960  PRINT D$;"OPEN ",N$
970  PRINT D$;"READ ",N$
980  FOR X = 1 TO 251
982  INPUT TIME(X),DT(X)
983  NEXT X
984  PRINT D$;"CLOSE ",N$
985  PRINT  CHR$ (16): RETURN
986  RETURN
1000  REM  ----------LOAD DATA AND BRING IN HRCG----------
1001  HGR : POKE  - 16302,0
1010  GOSUB 10000: REM  LOAD HIGH RES. CHAR. GEN.
1011  GOSUB 900: REM  LOAD IN DESIRED DATA FILE TO PLOT
1012  PRINT  CHR$ (15); CHR$ (1)
1015 CL =  - 16336
1020 FLAG = 0
1021  REM  ----------LABEL AXIS----------
1070 D$ =  CHR$ (4):G$ =  CHR$ (7)
1100 CL$ =  CHR$ (12): REM  LOWER CASE
1110 CK$ =  CHR$ (11): REM  UPPER CASE
1120 CS$ =  CHR$ (25): REM  SHIFT


]
```

LIST 1120-10250

```
1120 CS$ =  CHR$ (25): REM  SHIFT
1121  PRINT  CHR$ (16)
1122  VTAB 24: PRINT "     1     5     10     15      20 TIME"
1123  VTAB 22: HTAB 2: PRINT "100"
1125  VTAB 18: HTAB 3: PRINT "80"
1127  VTAB 14: HTAB 3: PRINT "60"
1129  VTAB 11: HTAB 3: PRINT "40"
1131  VTAB 7: HTAB 3: PRINT "20"
1133  VTAB 3: HTAB 4: PRINT "0"
1134  VTAB 4: PRINT CL$;"%"
1135  VTAB 6: PRINT CK$;"T"
1136  VTAB 7: PRINT CK$;"R"
1137  VTAB 8: PRINT CK$;"A"
1138  VTAB 9: PRINT CK$;"N"
1139  VTAB 10: PRINT CK$;"S"
1140  VTAB 11: PRINT CK$;"M"
1141  VTAB 12: PRINT CK$;"I"
1142  VTAB 13: PRINT CK$;"T"
1143  VTAB 14: PRINT CK$;"A"
1144  VTAB 15: PRINT CK$;"N"
1145  VTAB 16: PRINT CK$;"C"
1146  VTAB 17: PRINT CK$;"E"
1147  VTAB 1: HTAB 14: PRINT CK$;"DATA FILE : ";N$
1148 LEHI = (TIME(3) - TIME(2)) * 10
1149  VTAB 2: HTAB 23: PRINT CK$;"(I";CL$;"VL :";LEHI;" SEC.)"
1150  RETURN
10000  REM  PROGRAM TO LOAD HRCG
10010  ONERR  GOTO 10230
10020  HOME :ADRS = 0
10030  PRINT D$;"BLOAD RBOOT"
10040  CALL 520: REM  EXECUTE RBOOT
10050 ADRS =  USR (0),"HRCG"
10060  REM  BRING IN HRCG
10070 A = 1
10080  IF ADRS <  = 0 THEN ADRS = ADRS + 65536
10120 CS = ADRS - 768 * A: HIMEM: CS
10130 CH =  INT (CS / 256):CL = CS - 256 * CH
10140  POKE ADRS + 7,CL: POKE ADRS + 8,CH
10200  CALL ADRS + 3
10210  POKE 216,0
10220  RETURN
10230  TEXT : PRINT "UNABLE TO LOAD"
10240  POKE 216,0
10250  END : REM  END OF RPLOT

]
```

values before they can be plotted. This is accomplished in line 250. After the graph has been made, the user must push any key on the keyboard to continue. A question concerning the user's desire for a hard copy is then asked. If the user answers yes, he will be taken into the EPSON PLOT program; any other answer will return the user to the data manipulation menu. The EPSON PLOT program is a piece of commercial software, incorporated into the interface package. Its operation is via a self-explanatory menu.

There are several points to be made before moving on to the calculation program. First, the HRCG program is lengthy and not only takes time to load but also takes up a fair amount of the computer's memory. Due to this fact, the user should not make a plot of every run, but instead only of a representative sampling. The second point deals with the plot itself. Due to the way the HRCG prints on the screen, it is not possible to align all of the axis labels with the appropriate tick mark on the axis. For example, with the HRCG the user can only place a letter in any one of 24 positions on the screen in the vertical direction. With the graphics screen the user can place a point at any one of close to 200 locations. This difference in flexibility causes the non-alignment of the labels. The final point to be noted about HRCG concerns its ability to print both upper and lower case characters. By typing certain control symbols, the user can type in either character set. Lines 1100 to 1120 set certain string variables to the appropriate control characters; note the use again of the CHR$ command.

After being returned to the data manipulation menu, the user may choose choice three, the calculation of a first order rate constant. The program executed by the menu is named CALC (see figure 16). This program takes a data file and performs a linear least squares calculation to determine the best fit straight line. The program first asks the reader for the desired data file (line 900). Once the file is loaded, the program begins the calculation which takes approximately thirty seconds. Again, since the data stored in the file are not in terms of percent transmitance, the appropriate conversions must be made. After the best line is calculated, the results are displayed on the screen and the program prompts the user to press any key to see a first order plot of the data. However, before the plot can be made, two tasks are performed by CALC. The first concerns the second use of the scrap file. Originally, this file was used to store the interval length between data points; now it will be used for a different purpose. Not only does the first order plotting program need to know the time interval between points but it also must know the name of the data file being used. To this end, lines 431 to 436 erase the old values of SCRAP and record the name of the file being used and the interval time between points. The second

FIGURE 16 - CALC Program

```
]LIST 0-290

0   REM  ***********************************
1   REM  * PROGRAM : CALC                  *
2   REM  * PROGRAM TO CALCULATE RATE       *
3   REM  * CONSTANT AND INTERCEPT USING    *
4   REM  * LINEAR LEAST SQUARES FIT        *
5   REM  ***********************************
6   REM  VARIABLES :     TI=infinity transmitance value
7   REM                  AI=infinity absorbance value
8   REM                  T,A=transmitance and absorbance of point x
9   REM                  N=number of points to use in calculation
10  REM  CALCULATE BEST FIT STRAIGHT LINE FOR DATA POINTS IN A FIRST
11  REM  ORDER KINETIC PLOT. SLOPE IS -RATE CONSTANT AND INTERCEPT
12  REM  IS Ln(Ai-A0).AFTER PROGRAM IS FINISHED IT CALLS THE PLOTTING PGM

13  DIM C(251),R(251):D$ =  CHR$ (4): GOSUB 900
14  HOME :N = 0:D$ =  CHR$ (4)
15  HOME : VTAB 13: HTAB 14: PRINT "CALCULATING"
110  REM  ----------CALCULATE DATA VALUES----------
111 TI = (R(251) / 255) * 100
112 AI = 0 - ( LOG (TI) /  LOG (10))
120  FOR X = 1 TO 250
140 T = (R(X) / 255) * 100
141  IF  ABS (TI - T) ` 2 GOTO 180
142 N = N + 1
150 A = 0 - ( LOG (T) /  LOG (10))
160 R(X) =  LOG (AI - A)
170  NEXT X
175  REM  ----------CALCULATE BEST FIT LINE----------
176  REM  VARIABLES :     C(X):time of point x
177  REM                  R(X):transmitance of point x
178  REM                  M:best slope
179  REM                  B:best intercept
180  REM                  CC:correlation coeficient
181  REM                  VX,VY:variance in x and y data points
182  REM  USE LINEAR LEAST SQUARES TO FIND BEST FIT LINE.
183  REM  MAKE USE OF STRING COMMANDS TO PRINT ONLY THREE MOST
184  REM  SIGNIFICANT DIGITS.N IS THE NUMBER OF DATA POINTS TO BE
185  REM  USED IN THE ANALYSIS. N IS FOUND BY COLLECTING ALL
186  REM  OF THE DATA POINTS THAT DIFFER BY 2 TRANSMITANCE
187  REM  UNITS FROM THE INFINITY READING.
188 SY = R(1)
190 SX = C(1)
200 XY = R(1) * C(1)
210 Y2 = R(1) * R(1)
220 X2 = C(1) * C(1)
230  FOR J = 2 TO N
240 SX = SX + C(J)
250 SY = SY + R(J)
260 XY = XY + R(J) * C(J)
270 X2 = X2 + C(J) * C(J)
280 Y2 = Y2 + R(J) * R(J)
290  NEXT J

]
```

```
LIST 290-980

290  NEXT J
300 M = (N * XY - SX * SY) / (N * X2 - SX * SX)
310 B = (X2 * SY - SX * XY) / (N * X2 - SX * SX)
320 VX = (N * X2 - SX * SX) / (N * N)
330 VY = (N * Y2 - SY * SY) / (N * N)
340 CC = (M *  SQR (VX)) / ( SQR (VY))
345  REM  ----------PRINT RESULTS----------
350  HOME : PRINT : PRINT
360 M$ =  STR$ (M)
370 B$ =  STR$ (B)
380 CC$ =  STR$ (CC)
390  PRINT "THE SLOPE IS :"; LEFT$ (M$,5)
400  PRINT "THE INTERCEPT IS :"; LEFT$ (B$,5)
410  PRINT "THE CORRELATION COEF. IS :"; LEFT$ (CC$,5)
420  PRINT : PRINT
425  REM  ----------MOVE INTO PLOTTING PROGRAM----------
426  REM  PLACE DATA FILE NAME AND INTERVAL LENGTH IN SCRAP FILE
427  REM  THIS INFORMATION WILL BE USED BY CALCLABEL PROGRAM
430  PRINT "PRESS ANY KEY TO SEE PLOT": GET G$
431  PRINT D$;"OPEN SCRAP"
432  PRINT D$;"DELETE SCRAP"
433  PRINT D$;"OPEN SCRAP"
434  PRINT D$;"WRITE SCRAP"
435  PRINT N$: PRINT (C(4) - C(3)) * 10
436  PRINT D$;"CLOSE SCRAP"
437  PRINT D$;"RUN CALCLABEL"
438  END : REM  END OF SLOPE CALCULATION PROGRAM
890  REM  ----------GET DATA ----------
900  HOME : INPUT "ENTER DESIRED DATA FILE : ";N$
901 D$ =  CHR$ (4)
910  PRINT "LOADING DATA FILE : ";N$
920  PRINT D$;"OPEN ";N$
930  PRINT D$;"READ ";N$
940  FOR X = 1 TO 251
950  INPUT C(X),R(X)
960  NEXT X
970  PRINT D$;"CLOSE ";N$
980  RETURN

]
```

task of the calculation program is to call its sister program (CALCLABEL) which actually draws the plot.

CALCLABEL (see figure 17) is loaded in place of CALC and can only be executed from the CALC program. This program is very similar to the rough plotting program discussed earlier. The high resolution character generator is also used to label the axes. The first task in this program is to read the SCRAP file in order to determine which data file to plot and the interval length between points. This is accomplished in lines 900 through 930. After using the SCRAP file, the program reads the correct data file and converts the data into absorbance units. Since this is a first order plot, the natural log of the difference between the absorbance at time infinity minus the absorbance at time t must be calculated (ln(Aif-At)).

The program then clears the screen and plots all of the points that fall within a range of zero and negative three on the Y-axis. This usually includes at least 80% of the points. As in the other plotting program, the user has the option of obtaining a hard copy of the plot. If the user answers yes, the EPSON PLOT program is executed; all other answers will cause the return of control to the data manipulation menu.

There is one final program which does not appear on any of the menus but is useful to the interface package. It is entitled DATA PRINT (see figure 18) and can only be

FIGURE 17 - CALCLABEL Program

```
LIST 0-610

0   REM  ***********************************
1   REM  * PROGRAM : CALCLABEL              *
2   REM  * PROGRAM TO DRAW FIRST ORDER      *
3   REM  * KINETIC PLOT OF DATA FILE N$     *
4   REM  ***********************************
5   REM  VARIABLES :          R(X)=data point x
6   REM                       LEHI=length of interval, from SCRAP
7   REM                       N$=file name of data, from SCRAP
8   REM                       F=filler variable for time data
9   REM  THIS PROGRAM USES THE HIGH RESOLUTION CHARACTER GENERATOR
10  REM  PROGRAM IN ORDER TO PLACE ALPHANUMERICS ON THE
11  REM  GRAPHIC SCREEN.SEVERAL CONTROL KEYS ARE USED IN THAT SECTION
12  REM  AND ARE EXPLAINED THERE.PROGRAM FIRST DRAWS AXIS AND
13  REM  THEN PLOTS POINTS.THIS PROGRAM IS LINKED TO CALC
14  REM  AND CAN ONLY BE CALLED THROUGH IT.
15  DIM R(251)
59  REM  READ IN DATA AND HIGH RESOLUTION CHARACTER GEN.
60  GOSUB 1000
70  HOME :N = 250
440  REM  ----------LABEL AXIS----------
441  VTAB 22: PRINT "      1     5     10     15      20"
442  VTAB 2: HTAB 3: PRINT "0"
443  VTAB 8: HTAB 2: PRINT "-1"
444  VTAB 14: HTAB 2: PRINT "-2"
445  VTAB 21: HTAB 2: PRINT "-3"
446  VTAB 23: HTAB 17: PRINT "TIME"
447  VTAB 1: HTAB 16: PRINT "DATA FILE : ";N$
448  VTAB 4: PRINT " L"
449  VTAB 5: PRINT " O"
450  VTAB 6: PRINT " G"
451  VTAB 10: PRINT "(A";CL$;"I";CK$
452  VTAB 11: PRINT " -"
453  VTAB 12: PRINT "A";CL$;"T";CK$;")"
454  VTAB 23: HTAB 23: PRINT "(I";CL$;"VL :";LEHI;" SEC.)"
460  REM  PLOT X AND Y AXIS
470  HPLOT 28,10 TO 28,160 TO 278,160
480  REM  MARK OFF AXIS BY FIVES
490  FOR I = 10 TO 160 STEP 5
510  HPLOT 27,I TO 29,I
530  NEXT I
531  FOR I = 10 TO 160 STEP 5
532  FOR M = 28 TO 278 STEP 10
533  HPLOT M,I
534  NEXT M
535  NEXT I
540  REM  PLOT X AXIS HATCH MARKS
550  FOR L = 28 TO 278 STEP 10
560  HPLOT L,160 TO L,165
570  NEXT L
580  REM  PLOT 1 UNIT HATCH MARKS ON Y AXIS
590  FOR I = 10 TO 160 STEP 50
600  HPLOT 25,I TO 28,I
610  NEXT I

]
```

```
LIST 620-980

620  REM  ----------PLOT DATA POINTS----------
621  REM  IF A DATA POINT IS LESS THAN ZERO OR GREATER THAN 3
622  REM  DON'T PLOT IT. IT IS OUTSIDE AXIS OF GRAPH.
630  FOR X = 1 TO N
640  IF R(X) `  = 0 GOTO 660
650  NEXT X
655  REM  GRAPH IS AT LOCATION 10,28 ON SCREEN
660 Y1 =  - ((150 / 3) * R(X)) + 10
670 X1 = 27 + X
680  FOR Y = X + 1 TO N
690 Y2 =  - ((150 / 3) * R(Y)) + 10
691  IF R(Y) `  - 3.0 GOTO 740
700 X2 = 27 + Y
710  HPLOT X1,Y1 TO X2,Y2
720 X1 = X2:Y1 = Y2
730  NEXT Y
732  REM  USER IS FINISHED LOOKING AT PLOT
740  GET G$
750  PRINT  CHR$ (15); CHR$ (2)
760  HOME : TEXT
770 D$ =  CHR$ (4)
775  REM  ----------ALLOW USER TO GET HARD COPY----------
780  VTAB 12: HTAB 6: INPUT "WOULD YOU LIKE A HARD COPY? : ";Y$
790  ON Y$ = "YES" GOSUB 830
800  POKE ADRS + 10,0: POKE ADRS + 11,198
810  PRINT  CHR$ (15); CHR$ (25)
820  PRINT D$;"RUN MENU2"
830  PRINT D$;"RUN EPSON PLOT"
840  END : REM  END OF CALC PLOTTING PROGRAM
900  REM  ----------LOAD DATA INTO PROGRAM----------
901  PRINT D$;"OPEN SCRAP"
902  PRINT D$;"READ SCRAP"
903  INPUT N$: INPUT LEHI
904  PRINT D$;"CLOSE SCRAP"
920  PRINT D$;"OPEN ";N$
930  PRINT D$;"READ ";N$
940  FOR X = 1 TO 251
950  INPUT F,R(X)
960  NEXT X
961  PRINT D$;"CLOSE ";N$
962  HOME : VTAB 13: HTAB 14: PRINT "CALCULATING"
969  REM  ----------CALCULATE POINTS TO PLOT----------
970  PRINT D$;"CLOSE ";N$
971 TI = (R(251) / 255) * 100
972 AI =  - ( LOG (TI) /  LOG (10))
973  FOR X = 1 TO 250
974 T = (R(X) / 255) * 100
975  IF  ABS (TI - T) ` 2 GOTO 979
976 A =  - ( LOG (T) /  LOG (10))
977 R(X) =  LOG (AI - A)
978  NEXT X
979 N = X - 1
980  RETURN


]
```

```
LIST 980-10250

980  RETURN
999  REM  ----------LOAD H.R.C.G.----------
1000  HGR : POKE  - 16302,0
1010  GOSUB 10000
1011  GOSUB 900
1012  PRINT  CHR$ (15); CHR$ (1)
1013 CL =  - 16336
1020 FLAG = 0
1030 D$ =  CHR$ (4)
1031 G$ =  CHR$ (7)
1039  REM  CL$ AND CK$ CAUSE LOWER AND UPPER CASE LETTERS TO BE PRINTED
1040 CL$ =  CHR$ (12)
1050 CK$ =  CHR$ (11)
1060  PRINT  CHR$ (16)
1120  RETURN
10000  REM  PROGRAM TO LOAD HRCG
10010  ONERR  GOTO 10230
10020  HOME :ADRS = 0
10030  PRINT D$;"BLOAD RBOOT"
10040  CALL 520: REM  EXECUTE RBOOT
10050 ADRS =  USR (0),"HRCG"
10060  REM  BRING IN HRCG
10070 A = 1
10080  IF ADRS `  = 0 THEN ADRS = ADRS + 65536
10120 CS = ADRS - 768 * A: HIMEM: CS
10130 CH =  INT (CS / 256):CL = CS - 256 * CH
10140  POKE ADRS + 7,CL: POKE ADRS + 8,CH
10200  CALL ADRS + 3
10210  POKE 216,0
10220  RETURN
10230  TEXT : PRINT "UNABLE TO LOAD"
10240  POKE 216,0
10250  END

]
```

FIGURE 18 - DATAPRINT Program

```
]LIST

1   REM  ***********************************
2   REM  * PROGRAM : DATAPRINT             *
3   REM  * PROGRAM TO PRINT OUT RAW DATA   *
4   REM  * OF A RUN. USER CAN INPUT        *
5   REM  * DESIRED RANGE TO PRINT.         *
6   REM  ***********************************
7   REM  VARIABLES:      TIME :Array to hold time of data point
8   REM                  DT : Array to hold actual data point
9   REM                  N$ : Name of data file
10  REM  PROGRAM TO READ AND PRINT DATA DEPENDING ON USER PARAMETERS
20 D$ =  CHR$ (4)
30  DIM TIME(251),DT(251)
31  REM  GET NAME OF DATA FILE
32  REM  ----------LOAD DATA----------
40  HOME : INPUT "ENTER DESIRED DATA FILE :";N$
50  HOME : VTAB 12: HTAB 10: FLASH
60  PRINT "LOADING FILE:";N$
61  NORMAL
69  REM  LOAD DATA INTO COMPUTER
70  PRINT D$;"OPEN ";N$
80  PRINT D$;"READ ";N$
90  FOR X = 1 TO 251
100  INPUT TIME(X),DT(X)
110  NEXT X
120  PRINT D$;"CLOSE ";N$
125  REM  ----------PRINT DATA----------
130  HOME : PRINT "ENTER STARTING & ENDING PTS."
140  HTAB 4: PRINT "AND STEP SIZE"
141  REM  A,B AND C ARE THE STARTING,ENDING AND STEP SIZE
150  VTAB 8: INPUT "STARTING POINT :";A
160  VTAB 10: INPUT "ENDING POINT :";B
170  VTAB 12: INPUT "STEP SIZE :";C
171  HOME : VTAB 4
172  PRINT "POINT #    TIME(SEC)    %TRANS"
173  PRINT "-------    ---------    ------"
174  PRINT
180  FOR Q = A TO B STEP C
185  REM  CONVERT DATA POINTS TO TRANSMITANCE
189 T = (DT(Q) / 255) * 100
190  PRINT "  ";Q;"          ";TIME(Q);"     ";T
199  PRINT : PRINT
200  NEXT Q
210  END : REM  END OF DATAPRINT PROGRAM

]
```

run from the cursor mode of the Apple. The purpose of this program is to allow the user to obtain a listing of the actual data points of the run. The program initially asks the user for the name of the data file to use. This is followed by three questions. The first question asks the user to specify the data point the computer should use to start the listing; the second question asks how many points to skip between those listed, and, finally, the computer must be told the number of the point ending the listing. Once this is done, the computer makes the appropriate listing of points. This listing allows the user to check how well the interface package is operating or to use these data for non-first order fits.

## III. SAMPLE RUN

In order to test the validity of the stopped flow interface package, the kinetics of formation of blue peroxychromic acid was studied. This reaction has been studied extensively,[11] and is known to follow first order kinetics. The reaction envolves the mixing of the two solutions shown below :

SOLUTION A

10 ml. 0.005 M K Cr O

5.0 ml. 0.50 M $HNO_3$

SOLUTION B

1.0 ml. 1.0 M $H_2O_2$

5.0 ml. 0.50 M $HNO_3$

Each of these solutions were then diluted to a volume of 50.0 ml with water. The two solutions were each placed in a syringe on the stopped flow instrument. When these two solutions are mixed, they undergo the following reaction :

HCrO4 + 2 H2O2 + H+ --- CrO5.H2O + 2 H2O

This reaction is followed at 580 nm. where the Peroxychromate complex has a strong absorption band.

Four runs were made with data recorded by the computer and the oscilloscope. The collection time given the computer varied from 0.250 seconds to two seconds and the infinity delay time varied accordingly. The data taken from the four runs are shown in figure 19. The results of these runs are summarized in figure 20. All of the runs show very good agreement, with a maximum absolute difference of 1.2 transmitance units and with many of the readings agreeing exactly.

The error associated with reading the oscilloscope is at least one percent transmitance unit; the error in the A/D reading of the data signal is one unit in 255. If we keep in mind the fact that the power supply has a consistent tendency to drift to lower voltages, the values determined by the interface package are equal to those derived from the data displayed on the oscilloscope. This data cannot be compared to previously published literature values due to several reasons. Among these reasons is the fact that the temperature of the solutions was not kept at a constant 25.0 degrees centigrade, and the concentration of the acid solution was not standardized. Although a comparison to literature is not possible, the validity of the data is not jeopardized by this fact. As long as the computer matches

FIGURE 19 - Data from Test Run

RUN TIME : 0.250 seconds

INFINITY DELAY : 0.999 seconds

| TIME (msec) | TRANSMITANCE (%) | |
|---|---|---|
| | SCOPE | COMPUTER |
| 0 | 97 | 96.1 |
| 20 | 90 | 90.2 |
| 40 | 82 | 82.4 |
| 60 | 76 | 75.7 |
| 80 | 71 | 71.2 |
| 100 | 66.5 | 65.9 |
| 120 | 63 | 62.4 |
| 140 | 60 | 59.2 |
| 160 | 57.5 | 56.5 |
| 180 | 55 | 54.5 |
| 200 | 53 | 52.5 |
| INFINITY | 39 | 38.4 |

RUN TIME : 0.500 seconds

INFINITY DELAY : 0.750 seconds

| TIME (msec) | TRANSMITANCE (%) | |
|---|---|---|
| | SCOPE | COMPUTER |
| 0 | 97.3 | 96.1 |
| 20 | 90 | 90.2 |
| 40 | 82.5 | 82.4 |
| 60 | 76 | 75.7 |
| 80 | 70.5 | 70.6 |
| 100 | 66.5 | 65.9 |
| 120 | 63 | 62.4 |
| 140 | 59.8 | 59.2 |
| 160 | 57.5 | 56.5 |
| 180 | 55.0 | 54.1 |
| 200 | 53.5 | 52.2 |
| INFINITY | 39 | 38.0 |

RUN TIME : 1.00 second

INFINITY DELAY : 0.250 seconds

| TIME (msec) | TRANSMITANCE (%) | |
|---|---|---|
| | SCOPE | COMPUTER |
| 0 | 96.5 | 95.3 |
| 20 | 90 | 89.8 |
| 40 | 82 | 82.0 |
| 60 | 75.8 | 75.7 |
| 80 | 70.5 | 70.2 |
| 100 | 66 | 65.9 |
| 120 | 62.5 | 62.4 |
| 140 | 59.5 | 59.2 |
| 160 | 57 | 56.1 |
| 180 | 54.5 | 54.1 |
| 200 | 52.5 | 52.2 |
| INFINITY | 39 | 38.0 |

RUN TIME : 2.00 seconds

INFINITY DELAY : manual

| TIME (msec) | ANSMITANCE (%) | |
|---|---|---|
| | SC[illegible] | COMPUTER |
| 0 | 96.4 | 94.9 |
| 20 | 89.3 | ----* |
| 40 | 82 | 82.0 |
| 60 | 75.3 | ---- |
| 80 | 70 | 70.2 |
| 100 | 66 | ---- |
| 120 | 62.2 | 62.0 |
| 140 | 59 | ---- |
| 160 | 56.4 | 55.7 |
| 180 | 54 | ---- |
| 200 | 52.3 | 51.8 |
| INFINITY | 39 | 37.6 |

* Due to interval time, data was not available.

FIGURE 20 - Results of Test Run

RESULTS

| RUN NUMBER : | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| **OSCILLOSCOPE** | | | | |
| Slope | -5.48 | -5.43 | -5.64 | -5.73 |
| Intercept | -0.912 | -0.915 | -0.910 | -0.912 |
| Corr. Coeff. | -0.999 | -0.999 | -0.999 | -0.999 |
| **COMPUTER** | | | | |
| Slope | -5.56 | -5.28 | -5.38 | -5.34 |
| Intercept | -0.890 | -0.902 | -0.898 | -0.893 |
| Corr. Coeff. | -0.999 | -0.999 | -0.999 | -0.999 |

what is displayed on the oscilloscope, it is performing correctly (assuming of course that the oscilloscope is correct).

Although this interface package appears to be complete, there are several enhancements which may be desirable. There are four specific tasks that if pursued would make this interface package complete. Several of these are trivial while the fourth is fairly substantial.

At this point in the interface package, the only printer that can be used to print a hard copy of either plot is the Epson MX-80. The department has in its possesion an Apple Silentype printer. This printer will allow the high resolution screens to be printed as long as the proper printing program is active. In the future the user should be given the option of specifying which printer, if any, is connected to the system. This requires minor alteration in the software of the interface package and should be easy to accomplish.

The second improvment can be found at the end of the CALC program. At the end of this program the user is asked to press any key to see a first order kinetic plot. Instead of automatically being sent into the plotting program, the user should have a choice of making a plot or not. This enhancement also is fairly trivial and should be incorporated into the package.

At no point in the program is the user given the option of listing the raw data points. The DATA PRINT program should be an option in the data manipulation menu. There may be some point were the user may be studying a reaction that does not follow first order kinetics, and he may want to use the stopped flow interface to take data. At the present time the user does not have the ability to simply list the data of a given run. Also, this improvment should not be too difficult.

The final enhancement is much more difficult to incorporate into the package. As stated earlier, the stopped flow interface package may be used in the future for reactions that are not first order. The interface package could be made much more powerful if it could calculate rate constants for non first order reactions. This would require a large amount of additional programming, but should fit easily into the scheme used in this project.

Appendix one of this report contains a detailed manual describing the use of the stopped flow interface. Also included within the manual are sample outputs produced by the interface programs.

# SECTION IV: Users Manual

UN82 LABY, D. M. THE APPLE II/STOPPED-FLOW, ETC.
L127a/1983 CHEMISTRY HRS. 5/83 SHT. 2 OF 2

END

This manual, in a step by step fashion, is designed to teach the reader how to use the STOPPED FLOW INTERFACE PROGRAM. There are two parts to this interface. Before taking data, the user must make the necessary external connections between the instrument and the computer. Once all of the external connections are made, the user makes use of the computer to set the necessary pre-conditions for the interface. Finally, the user has the option, to take data from the instrument or to analyze, with some of the other programs included in the interface package, data taken previously.

Initially, the hardware configuration of the interface must be set. This task involves five different components: the stopped flow instrument with oscilloscope, the Apple computer, the power supply, the stopped flow interface box, and the interface disk.

I. EXTERNAL CONNECTIONS

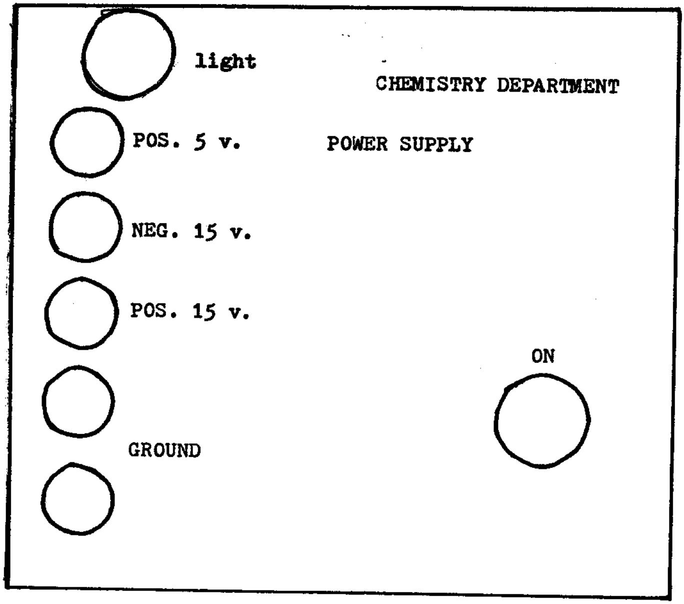
1) using the banana plugs provided connect the power supply to the interface box. For example, the connection port labeled pos. 15 V. on the power supply should be connected to the port on the interface box labeled +15 v. When all the connections are made one of the GROUND connections on the power supply will remain unused (see fig 1).

2) Before turning the power supply on, the user should connect the stopped flow instrument to the interface box. This is accomplished by connecting the long cords from the "T" bar connection on the oscilloscope to the labeled points on the interface box. For example, the trigger line on the oscilloscope should be connected to the trigger position on the interface box, etc.

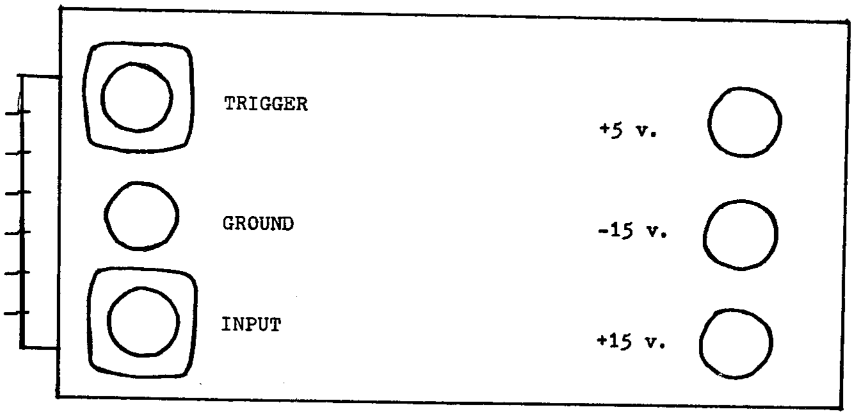
3) The final connection is between the interface box and the Apple computer's A/D board. This task is accomplished by connecting the ribbon cord located behind the computer to the side of the interface box. At this point all of the connections have been made which will enable the Apple to collect data from the stopped flow instrument.

4) The final task in initialization involves turning all

POWER SUPPLY




INTERFACE BOX (CIRCUIT)



power switches to the on position. This includes: the power supply for the stopped flow, the oscilloscope, the power supply for the interface box, the video monitor on the computer and finally the computer itself (this should only be done after the user has placed the interface disk into the disk drive). At this point the user should see a menu on the video monitor of the computer (see fig.2)

MANIPULATION OF DATA PROGRAMS

1.STORE NEW DATA

2.PLOT RAW DATA

3.CALCULATE FIRST ORDER RATE CONSTANT

4.GO BACK TO MENU 1

5.QUIT

ENTER A NUMBER AND PRESS RETURN. 1

## II. SET UP

The user is now in position to 'SET UP' the conditions for the interface. This consists of two tasks; firstly, the user must be certain that the stopped flow instrument and the computer are grounded at the same point, and, secondly, the user must set a ten volt range on the computer using the power supply for the photomultiplier. Once this is accomplished on the computer, the user can then adjust the oscilloscope if desired to the proper settings.

1) Press option one on the menu program (labeled SET UP). The user should now see a new picture on the video monitor(see figure 3). By turning the fine offset knob on the photomultiplier power supply, the user should adjust the voltage reading on the screen to be between zero and .1 volts (optimally this value should be flashing between 0 and .03 volts). This procedure has the effect of grounding the PMT and the computer at the same potential.

2) The interface box is designed to accept a potential between ground and ten volts. In the previous step we have set the ground; we therefore must now set the ten voltage maximum output of the PMT. By first turning the

SET-UP PROCEDURE

PROCEDURE TO COUPLE COMPUTER TO PMT.

INCREASE OFFSET ON PHOTO-
METER UNTIL VOLTAGE ON COMPUTER
IS 0 TO .1 VOLTS (FLASHING).
VOLTAGE : 8.2

PRESS SPACE BAR TO RETURN TO MENU

course adjust and then the fine adjust, set the voltage on the computer to as close to a constant 10 volts as possible. Once again there is an optimal value here; the optimum is achieved when the monitor has between a 10 and a 9.9 volt display flashing on and off.

## III. A TYPICAL RUN

At this point we are ready to begin taking data. This manual assumes the reader is already familiar with the use of the stopped flow instrument. The most efficient way to demonstrate the use of the interface software is to lead the user through a typical run.

### 1) DATA ACQUISITION

Enter the number 2 on the menu for data acquisition. At this point the computer will ask two questions (see figure 4). The first question determines the length of time the computer will take data. This value can range from .250 seconds to 249 seconds. If the user enters a value of less than .250 the computer will re-ask the question. The second question deals with the amount of time the computer should delay before taking an infinity reading. The user may enter a value between 1 and 999 as the number of milliseconds the computer will automatically wait, or, if so desired, the user may chose a manual infinity value; in this case, a value of 0 should be entered. A manual infinity value causes the program to prompt the user as to the point in time to take a value. After the run is over, a prompt line will

DATA ACQUISITION PROGRAM

```
ENTER LENGTH OF RUN IN SECONDS : .4
INFINITY VALUE OPTION


    1.ENTER 0 FOR MANUAL INFINITY VALUE

    2.ENTER DELAY TIME (1-999 MILLISEC.)


INPUT DESIRED TIME (0-999): 200
START'RUN'WHEN'RED'LIGHT'GOES'OFF
```

be printed on the monitor telling the user how to record the infinity value. After the run conditions are set, the computer states when it is ready to begin the run; anytime after this, the user may begin the run by depressing the plunger button on the stopped flow.

2) SAVE DATA ON DISK

Once the run has ended, the disk drive will spin for a few seconds, and another menu (see figure 5) will be printed on the screen. The first task that must be performed with the data set is to save it permanently on the disk (WARNING: failure to save data at this time may result in loss of data, see figure 6). This is accomplished by entering the number one for saving data and pressing the return key. The program will ask the user to enter a name under which the data for the run will be stored. This name can consist of either numbers or letters and may include any key on the keyboard (except a comma and the first character must be a letter). One must keep in mind that using a name already used to save data will result in the loss of the old data, with the new data stored in its place. The computer will require about 45 seconds to save all of the data after which it will return the user to the menu. At this point, the user is free to examine the data in several different ways or shut the system down and return at a

MENU FOR STOPPED FLOW INTERFACE

1.SET UP

2.DATA COLLECTION

3.MANIPULATE DATA

4.QUIT

ENTER A NUMBER AND PRESS RETURN. 3

SAVER PROGRAM

```
INPUT FILE NAME TO SAVE DATA : TEST1
SAVING:          TEST1
```

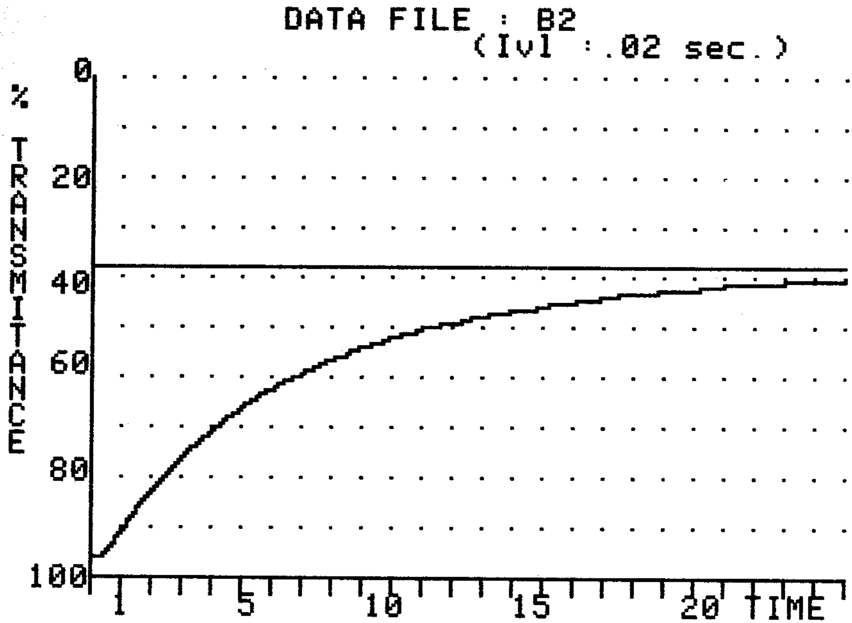
later date to examine and manipulate the data.

3) EXAMINE DATA

The most logical item to execute after storing the data set on the disk is to look at it in a plot of transmitance versus time. This is accomplished by pressing choice 2 on the menu and answering the subsequent questions(see figure 7). The program will ask the user for a file name. Care should be taken to enter the correct name, since the plotting program takes a few minutes to load completely. After the computer has made a plot of the data, the user has the option of making a hard copy print-out of the picture. The user simply presses any key to be prompted by the question concerning a permanent record of the picture. If the user desires a hard copy, the computer must be hooked up to the Epson printer and the printer must be turned on. By answering no to the question concerning a hard copy the user can return to the menu.

4) CALCULATE RATE CONSTANT

This part of the interface assumes that the user is working under conditions which are first order with respect to a given reaction (see figure 8). From the menu, enter the number 3 to calculate the first order


DATA FILE : B2
(Ivl : .02 sec.)
% TRANSMITANCE
0
20
40
60
80
100
1
5
10
15
20
TIME

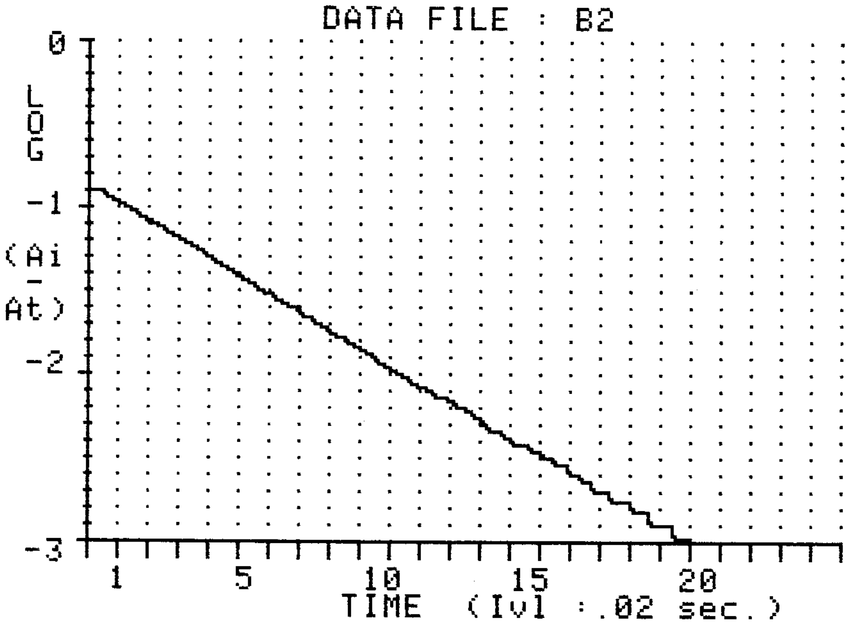
```
ENTER DESIRED DATA FILE : B2
LOADING DATA FILE : B2


              CALCULATING


THE S' OPE IS :-5.28
THE INTERCEPT IS :-.902
THE CORRELATION COEF. IS :-.999


PRESS ANY KEY TO SEE PLOT
OPEN SCRAP
```

rate constant (if by chance you find yourself in the first menu , also enter 3 to move to the data manipulation programs which are located on the second menu). This program will ask the user to enter a file name for the data he wishes to use; once entered, the computer will require about a minute to complete the proper calculation and print the desired rate information. Once again the computer will make a first order plot of the data (see figure 9); this is accomplished by pressing any key after the computer has printed out the information as to the best fit line for the data. After the plot has been made on the screen, the user has the option of producing a hard copy of this picture. The questions are identical to those used above in the rough plot program. In any case, after the user has ended this program he will be returned to the menu.


DATA FILE : B2
0
LOG
-1
(Ai
-
At)
-2
-3
1
5
10
15
20
TIME (Ivl : .02 sec.)

## IV. GENERAL GUIDLINES

There are a few general guidelines that are helpful in using this interface package. The user should keep in mind that all menu items return the user to the menu after execution. In fact, there are two different menus in this package. The first menu allows the user to set the initial conditions for the interface and to collect data, while the second menu includes all of the data manipulation programs. When the system is started the first menu will be automatically displayed, after which the user may chose the desired task. If the user finds he has accidently left the program for any reason he may recover by typing RUN SF-INTFCE from the cursor mode. If all else fails the system can be re-started by turning the computer off then on with the interface disk in place. This will bring the user back to the first menu. There is one drawback to this method: by turning the computer off the user will lose any data which has not been stored on the disk (data not saved with choice one on menu two).

INSTRUCTIONS FOR USE OF THE
STOPPED FLOW INTERFACE

I. Hardware (Electronic Connections)

1.Parts :

a)Stopped Flow Instrument with Oscilloscope
b)Apple Computer
c)Power Supply
d)Stopped Flow Interface box
e)Interface disk

2.Connections :

a)Using banana plugs connect all labeled points between interface box and power supply.

b)Connect trigger and input leads from oscilloscope to interface box.

c)Connect A/D plug from Apple Computer to side of interface box.

3.Initialization :

a)Place program disk into disk drive
b)Turn on Video Monitor
c)Turn on Apple Computer(switch on back)
d) Follow set-up routine.

NOTE : At this point a menu should appear on the screen

II. Software (Computer programs)

A TYPICAL RUN

NOTE :Knowledge of the use of the Stopped flow Instrument is assumed.

1.Data Acquisition

a)Enter number 2 (Data Collection) from Menu.
b)Enter length of time computer should take data(less then 250 seconds)

c)Enter delay time to take "infinity" reading

i)If zero is entered computer will wait for user to press any key to take infinity reading (manual)

ii)If 1-999 is entered computer will wait the specified number of milliseconds.

d)When red light on disk drive goes off, start reaction.

2.Save Data on Disk :

a)Enter number 1 (store new data) from menu.

b)Enter name under which you want data to be stored.

NOTE: Entering a name already used will cause replacement of old data by the new.

3.Examine Data :

a)Enter 2 (plot raw data) from menu

b)Enter name of desired data file (used in 2 above)

c)Disk drive will spin for about a minute before plot is displayed on screen.

d)After viewing plot, press any key to move on.

e)Answer <u>YES</u> to hardcopy question if desired.

NOTE: Apple must be connected to printer and printer must be turned on.

f)If answer is <u>NO</u>, computer will return to menu.

4.Calculate Rate Constant :

a)Enter 3 (Calculate first order rate constant) from menu.

b)Enter name of desired data file (used in 2 above)

c)Computer will take about 30 seconds to calculate.

d)Follow directions on screen to see plot of data.

5.General :

a)All menu items return user to menu at end of procedure.

b)There are two menus - They can be mutually accessed.

c)If you are in Apple Cursor mode (a ] followed by a flashing star) type <u>RUN</u> <u>SF-INTFCE</u> and press return.

d)If all else fails... turn computer off then on again.

NOTE: This will result in a loss of any data <u>not</u> already stored on disk (with store new data option).

# REFERENCES

(1) A/D + D/A Operating Manual, Mountain Computer Inc., 1980

(2) Apple Clock Operating Manual, Mountain Computer Inc., 1980

(3) Aminco-Morrow Stopped-flow Apparayus, No.4-8409 Instruction No.939, 1969

(4) Kepco Voltage Regulated DC Power Supply; Instruction manual, Kepco Inc., 1967

(5) 6502 Software Design, Leo J. Scanlon; Howard Sams & Co., Inc, 1980

(6) Apple II Monitors Peeled, Apple Computer Inc., 1981

(7) Apple 6502 Assembler/Editor, Apple Computer Inc., 1980

(8) The DOS Manual, Apple Computer Inc., 1980,1981

(9) Apple II Reference Manual, Apple Computer Inc., 1979

(10) Basic Programming Reference Manual, Apple Computer Inc., 1978

(11) a. Howlett, K.E.; Sarfield, S. J. Chem. Soc. (A), 1968,683 and b. Linge, H.G.; Jones, A.L. Austral. J. Chem., 1968, 21, 2189 and c. Haight, G.P.; Richardson, D.C.; Coburn, N.H. Inorg. Chem., 1964,3, 1777 as referred to in Chemistry 150 laboratory experiment: The Kinetics of Formation of Blue Peroxychromic Acid in Aqueous Solution, Union College, 1982.

ÉND