1 INTRODUCTION TO COMPUTER DATA ACQUISITION

Two experiments in this laboratory (Flash Photolysis and I_2 Fluorescence) involve measuring signals which vary rapidly with time. This presents a considerable challenge to the experimentalist, and we find it most convenient to acquire data with the computer.

Computer data acquisition requires knowledge of both computer software and computer hardware. It is not magic; wires have to be run from the apparatus to the computer and code must be written to make sense of the voltages on the wires. This is a large, but not insurmountable, task, and the objective in this lab is to give you some idea how this is done with the hope of dispelling some of the mystique surrounding computer data acquisition.

Software

The software platform we have chosen for *acquiring* data is MATLAB because most of you have had some experience with MATLAB. Actual interfacing of instruments and computers is too complex for us to have you start from scratch in this lab, and we have hidden the nitty-gritty details of the interfacing in C-language MATLAB `.mex` files. These routines are .m files (`.m` files written in another language) which you call as any other `.m` file, but of course you need the correct hardware (only in 229 SPAC) and you need to make the appropriate calls and electronic connections in the lab.

The use of these routines is probably best described by using as example a simple routine called SCOPE which you will need for the flash photolysis experiment. SCOPE is a routine which you call from MATLAB as you would any one of the toolbox functions, with the important exception that it accesses hardware which is available only in 229 SPAC. This routine enables the computer to act like an oscilloscope which measures how a voltage varies with time. In typical MATLAB style, V and T are each stored as vectors, and you "read" the oscilloscope with the MATLAB call

```
[V,T] = SCOPE (a,b,c,d,e)
```

where "a,b,c,d,e" are experimental parameters which are discussed later. Note that execution of this one simple command stores *experimental* information in V and T!

In a similar vein, another routine called Tek2221 is used in the I_2 fluorescence experiment. This is considerably different because a digital storage oscilloscope actually acquires the data and the computer simply reads the oscilloscope. In this case you actually read the oscilloscope with the Matlab call

```
[V,T] = tek2221(a,b,c,d,e)
```

where "a,b,c,d,e" are experimental parameters which are discussed later

Once you can acquire data several other "problems" arise. You can generate a large amount of data (thousands of points) and you have to deal with this data once you've taken it. You will expend some considerable time taking the data, so you will wish to save it. This also facilitates the analysis, which can then be done perhaps at a later time.

We have tried to structure the MATLAB environment so that you can use your experience to best advantage in terms of dealing with the data, rather than taking it. We thus provide .m files to enable you to store and to retrieve your data which are discussed later. These are WFLASH.m and RFLASH.m, and WFLUOR.m and RFLUOR.m for the Flash Photolysis and I_2 Fluorescence experiments, respectively,

Before coming to the lab, you need to acquire some experience with these routines so that you are able to properly store your data so that you can retrieve it! For this purpose, we have written dummy routines called DSCOPE and DTEK2221 which will enable you to read data stored on the Web site to familiarize yourself with the experimental calls, with the storage and retrieval files, and to enable you to think about writing your own routines for data analysis. These files will not require the hardware in room 229 SPAC and should be used before coming to lab.

Getting Started

Flash Photolysis (this section is analogous to the I2 Fluorescence section below)

DSCOPE, WFLASH, RFLASH, and the data file for DSCOPE, dscope.mat are in the P.Chem Lab section of the Web site **python.rice.edu/~brooks**. You can copy these files and tailor them to your own satisfaction. These routines have been written for Macintosh because the lab computers are Macintosh. (The file I/O will not work with the Mac student edition.) RFLASHX is a modification of RFLASH in which the Mac dialog for file opening is commented out.

You should use DSCOPE to "acquire" waveforms under several different combinations of points and dwell time, and then plot them to see what they look like. You might want to use SUBPLOT to plot two waveforms so that you can really compare the effects of different calls[1].

You need to test WFLASH and RFLASH to assure yourself that you know how to read and write data to a file. The call to WFLASH is

```
z = WFLASH(V,T);
```

where z is an irrelevant variable (nothing is returned to it) and V, T are the names of the vectors containing the voltage and time. (They can have other names, which you insert in the call.) This routine asks for a name (anything will do--your name, your group letter, initials, etc. I suggest you use the same for each file you save) WFLASH then concatenates your name with the word "flash" and a 6 digit number representing the time (hour, minute, seconds). Group "Q" could thus produce a file entitled "Qflash.164312". Automatic appending of the time eliminates remembering a new name and ensures that you

[1] (DSCOPE reads data from the same file every time it's used, and the stored values are used to produce [V,T] for different numbers of points and different dwell times. "Data" obtained for different dwell times will be very slightly different because the data are averaged differently.)

don't overwrite a file. After a filename is generated you have two lines for comments to place in the file, which we recommend you use to record the acquisition conditions (all the calls to SCOPE, for instance, as well as the conditions of the sample you're studying.) Following those two lines, V and T are written into the file. (The semicolon in the call suppresses intermediate output; if you forget, command-period will terminate the output by terminating execution of the .m file, so be careful when and how you terminate, because your file i/o might not be complete when you terminate.

After writing your data to a disc file, try reading it with the call

```
[V1, T1] = RFLASH;
```

This opens a Macintosh dialog box enabling you to open the file you want; RFLASHX requires you to enter the entire name from the keyboard which might be necessary for non-Mac systems. The two lines of explanation are read out to the screen, and then V1 and T1 are read. (By using different symbols here you can plot the data and determine if it has correctly been written and read.)

I_2 Fluorescence (this section is analogous to the Flash Photolysis section above)

DSCOPE, WFLASH, RFLASH, and the data file for DSCOPE, dscope.mat are in the P.Chem Lab section of the Web site **python.rice.edu/~brooks**. You can copy these files and tailor them to your own satisfaction. These routines have been written for Macintosh because the lab computers are Macintosh. (The file I/O will not work with the Mac student edition.) RFLASHX is a modification of RFLASH in which the Mac dialog for file opening is commented out.

DTEK2221, WFLUOR, RFLUOR, and the data file for DTEK2221, DTEK2221.mat are in the P.Chem Lab section of the Web site **python.rice.edu/~brooks**. You can copy these files and tailor them to your own satisfaction. These routines have been written for Macintosh because the lab computers are Macintosh. (The file I/O will not work with the Mac student edition.) RFLASHX is a modification of RFLASH in which the Mac dialog for file opening is commented out.

You should use DTEK2221 to "acquire" waveforms under several different combinations of points and dwell time, and then plot them to see what they look like. You might want to use SUBPLOT to plot two waveforms so that you can really compare the effects of different calls[2].

You need to test WFLUOR and RFLUOR to assure yourself that you know how to read and write data to a file. The call to WFLUOR is

```
z = WFLUOR (V,T);
```

where z is an irrelevant variable (nothing is returned to it) and V, T are the names of the vectors containing the voltage and time. (They can have other names, which you insert in the call.) This routine asks for a name (anything will do--your name, your group letter, initials, etc. I suggest you use the same for each file you save) WFLUOR then

[2] (DTEK2221 reads data from the same file every time it's used, and the stored values are used to produce [V,T] for different numbers of points and different dwell times. "Data" obtained for different dwell times will be very slightly different because the data are averaged differently.)

concatenates your name with the word "fluor" and a 6 digit number representing the time (hour, minute, seconds). Group "Q" could thus produce a file entitled "Qfluor.164312". Automatic appending of the time eliminates remembering a new name and ensures that you don't overwrite a file. After a filename is generated you have two lines for comments to place in the file, which we recommend you use to record the acquisition conditions (all the calls to DTEK2221 , for instance, as well as the conditions of the sample you're studying.) Following those two lines, V and T are written into the file. (The semicolon in the call suppresses intermediate output; if you forget, command-period will terminate the output by terminating execution of the .m file, so be careful when and how you terminate, because your file i/o might not be complete when you terminate.

After writing your data to a disc file, try reading it with the call

```
[V1, T1] = RFLUOR;
```

This opens a Macintosh dialog box enabling you to open the file you want; RFLUORX requires you to enter the entire name from the keyboard which might be necessary for non-Mac systems. The two lines of explanation are read out to the screen, and then V1 and T1 are read. (By using different symbols here you can plot the data and determine if it has correctly been written and read.)

Hardware

The Chem 313 Macintosh is equipped with two National Instruments plug-in cards which communicate with laboratory instruments:

NB-GPIB ("General Purpose Interface Bus")
: requires a compatible interface *on the instrument itself*, and is used to acquire data with the Tektronix 2221 digital storage oscilloscope in the I_2 fluorescence experiment. This board is relatively specialized and we shall not go into detail about it here.

NB-MIO16 ("Multiple Input/Output")
: is a general purpose multiple input/output board capable of measuring signals from a wide variety of general instruments. National Instruments describes its board in the following manner:

> The NB-MIO has a 12-bit ADC with 16 analog inputs, two 12-bit DAC's with voltage outputs, eight lines of TTL-compatible digital I/O, and three 16-bit counter/timer channels for timing I/O.

What does this mean, and what can it do for you?

ADC stands for analog-to-digital converter. It converts an analog voltage from some laboratory circuit into a digital signal that can be read by the computer. (In the Flash Photolysis experiment the voltage from the photomultiplier is digitized as a function of time.) These instruments span a pre-determined range, typically -10 V to + 10 V, although this can be changed (for instance to -5 V to + 5V) by opening the computer and changing the wiring on the circuit board. We will not do this. The converted voltage is expressed as a 12-bit

binary result, and the maximum resolution will be one part in 2^{12} or 1/4096 = .000244. Since the maximum voltage span is 20 V, the resolution is 4.88 mV. If greater voltage resolution is required, more bits (say 16) would be necessary and this is more expensive.

These measurements can be made very quickly. The dwell time is not specified in the description above, but the actual data specification[3] quotes typical times of 20μs. (There are some complications which we will ignore here.)

This board can digitize 16 input signals, but there is a catch. As the computer scientists are fond of saying, "garbage in = garbage out". A problem with laboratory measurements is noise, and the computer will quite happily digitize the noise as well as the signal. Steps must therefore be taken to reduce the noise before it gets to the computer, and a very effective way to do this is to make *differential* measurements. Voltage is undefined in an absolute sense; only voltage differences are significant. Voltage must thus always be measured between two points (as a battery has two terminals) although frequently one of these points is ground (arbitrarily assigned to be zero). If the wires connected to the instrument are a twisted wire pair (or a shielded pair), the noise impressed on each wire is about the same. Measuring the *difference* in voltage causes the noise to be subtracted out and is known as common mode rejection. This is the method used in this laboratory. In this differential mode two inputs are required for each signal and the board can thus digitize only 8 different signals. (The inputs are numbered 0-15; in differential mode 0 and 8 are paired, 1 and 9, etc.)

DAC stands for digital-to-analog converter. It converts a digital signal specified by the computer to an analog signal which could be used to control some aspect of an experiment, such as a heater, but see below. Two voltages can be controlled. Again the output voltage resolution is 1 in 12 bits. The range is ± 10 V so the resolution is 4.88 mV. The current is limited to 2 mA, so the heater in our example needs to be very small [20 mW] or the voltage must be used somehow to *control*, not power, the heater. Response times are ≈ V/μs. This is not currently used.

TTL-compatible **I/O**. TTL stands for "transistor-transistor-logic" which specifies various voltage and current levels[4]. Two voltage levels are recognized: 0 and 5 V, which can correspond, for example to "on" and "off". I/O is input/output, and this section of the board is capable of either sensing whether a component is "on" or "off" or it can be used to output a signal which will control the "on/off" state of some component. The eight separate lines are grouped into two sections of four lines; the four lines in a given section can be configured to be either all input or all output. The two sections are independent. Time response should be ≈ μs. This is not currently used.

The **counter/timer** circuits are useful in various timing applications. Output pulses can be sent at predetermined times to turn equipment on/off, and A to D measurements can be triggered by external timing pulses. External pulses can be counted to give either total

[3]National Instruments NB-MIO User Manual (1988)
[4] Horowitz & Hill, *The Art of Electronics*

number of counts, frequency, pulse width, or time intervals. Frequency response ≈ MHz. This is not currently used.