DATACP

INSTRUCTION MANUAL

Version 15, 1995

Paul B. Manis, Ph.D.
Department of Otolaryngology-Head and Neck Surgery
420 Ross Bldg.
720 Rutland Ave.
Johns Hopkins University School of Medicine
Baltimore, MD 21205

Based on the Manual for Datac, V3.2 by:
Daniel Bertrand, E. Favre
DÄpartement de Physiologie
Centre Médical Universitaire
9 avenue de Champel
1211 Genève 4
Switzerland

INTRODUCTION

OVERVIEW

DATAC (and its derivative program, DATACP) is a program designed for the analysis, manipulation and graphical representation of any kind of data. As you will see after having used DATAC for some times, it is quite difficult to define this program in a few words. Basically, you will find your own definition of DATAC because you are going to use it your way. Depending upon whether you are a mathematician, a biochemist, a student in any kind of field, an electrophysiologist or a morphologist, DATAC will not represent the same tool. The reason for that is that DATAC is not a program including an endless number of commands that will perform many tasks in a rigid way. Rather, DATAC contains few commands but is able to perform tasks it has not be programmed for! Moreover, you can ask DATAC these tasks in a language most people will know, regular algebra. The flexibility of DATAC is due to the fact that you can easily write equations that DATAC will compute immediately. In these equations entire arrays (of up to 512 values) of data are treated from your point of view just as if they were an individual number. In a way, you can think of DATAC as an elaborate pocket calculator in which nearly any kind of equations can be written and executed on groups of data and in which the result can immediately be displayed graphically. Indeed, you can visualize graphically and in a few seconds many mathematical functions without having to actually write a program. In that sense, DATAC turned out to be an unexpected self teaching tool in mathematics.

* * *

DATAC was originally developped for the acquisition and analysis of electrophysiological data and you will find some traces of it in this manual. The design of DATAC, however, allows it to handle any type of data and to perform all sorts of analysis. So just skip the electrophysiological parts if you do not use them.

The core of DATAC is made up of 4 major elements:
1) A set of operators (+-/*, trigonometric, ln, exp etc and some other more specialized operators which will be described in due time)
2) A particular structure, called a strip, which contains, among other things, an array of up to 512 data values
3) A mathematical interpreter able to understand regular algebra notation and to compute the equations you write.
4) A few specialized modules (curve fitting, differential equations, etc).

The operators can act on individual numbers and on strips. You can write on the terminal mathematical equations with various combinations of operators, numbers and strips. The mathematical interpreter will interprete the equation so that it can be computed by the program. Therefore, you can do any type of analysis you want, provided you can write the appropriate equation.

CONDITIONS UNDER WHICH DATAC WILL BE ABLE TO FUNCTION

DATAC is written in C-language and is designed to function on IBM PC or IBMPC-like machines running the MSDOS version 2.11 or up. The computer should be equiped with a mathematical co-processor 8087 or 80287 and with at least 380 kbyte of memory. The size of the present version of DATAC is only approximately 200 K but DATAC uses the memory between the top of the program and the system for data strorage and computations. Therefore, the bigger the memory, the better! The original version of DATAC was made for a color screen display but a monochrome screen can also be used (see below). All the data and text displayed on the screen can also be displayed with a higher quality on a plotter. The only fully supported plotter is the Hewlett Packard 7550A. However, other plotters which accept the HPGL commands can be used or implemented.

Terminals supported by the DATAC version:
- -Enhanced color display (IBM) with 256 K RAM on display adapter
- -Standard RGB monitor
- -Professional display (IBM)
- -Tektronix 4105
- -AST preview monochrome display (IBM)

CONDITIONS FOR DATACP:

DATACP is a protected mode version of DATAC, with several enhancements, which are described in this manual. DATACP requires at least 480K of DOS memory, plus extended memory, an 80386 processor or better, and a math coprocessor (80387 or better). DATACP requires a color monitor and graphics card capable of at least EGA graphics. DATACP also requires the PharLap 286 DOS Extender program be available. With these requirements, DATACP can address all of the memory that is generally available in a computer, permitting the user to load large amounts of data and many strips (thousands of the 2048 length strips). DATACP was derived from DATAC in 1993 by Paul Manis. DATACP also supports data acquisition, although it is not recommended for this. DATACP runs well in either a DOS environment, or as a protected mode program under Windows (3.1, 3.11, Windows95 have all been tested).

DATACP supports:
Access to networked drives in networked environments.
HPIII and HP4 laserJet Printers/plotters, EPSON 8-pin printers and raster graphics dumps.
HPGL-based plotters, such as the 7550 and BBC-GoertzMetraWatt 284.
EGA and better graphics monitors.

This manual specifically relates to DATACP, and to the modified version of DATAC that runs in the Manis laboratory. Refer to the V3.2 Manual (Bertrand and Favre) for the earlier DATAC descriptions.

The information in the manual is organized in chapters:

An index is available at the end of this manual. Note that in this manual you will often see words connected by a _ . This has been done to facilitate the indexing of this manual and has no special meaning.

FIGURE 1 : DATAC COMMANDS

QUIT	to quit the program. Confirm with y

COMPUTATIONS
DIFF	differential equation
FITTING	activates the curve fitting
PRINTVAR	lists the buffer of variables
ELECTROPHYSIOLOGY
AVERAGE	on-line averaging with storage of the result only CLEAR	deletes records during acquisition
CROSSHAIR	activates the crosshair for data readout
DATA	activates data acquisition
DIR	gives the directory of selected drive
DISPLAY	direct display of electrophysiological data
ERAIV	deletes the strips created by IV commands
ERASE	erase the graph
IVFIX	IV??? are a series of commands for current-voltage measurements IVMAX	IVFIX measures at a fixed time, IVMAX and IVMIN look for IVMIN
max and min current over a given interval. IVV gives a current IVV	at one time as a function of voltage at another time IVTRACE	retrace all
iv strips in memory
NUMBER gives the last record number in data acquisition OPEN	opens a file for data replay
PARAMETER	sets parameters for direct display
SPECTRUM	calculates the power spectrum
SUBTRACT	direct subtraction of electrophysiological data INPUT / OUTPUT
DIR	gives the directory of selected drive
OPEN	opens a file for data replay (electrophysiology)
READ	retrieves an ASCII file (MULTIPLAN or regular)
WRITE	saves on disk an ASCII file (for MULTIPLAN or other) RETRIEVE	recovers a strip from the disk
SAVE	saves a strip on the disk
CLOSE	closes a file that was open for input/output
FSAVE	saves functions in LIBRARY.FUN
FRETRIEVE	retrives functions from LIBRARY.FUN

GRAPHICS

BLANK	selective removal of some of the data in a strip
CROSSHAIR	activates the crosshair for data readout
DISPLAY	direct display of electrophysiological data
ERASE	erase the graph
ERADIAL	erase the dialog area
ERATEXT	deletes the strips of text
GRAPH	reactivates the graph mode
RETRACE	retraces the previous set of curves after any modification
PTEXT	prints text on terminal or plotter
TEXT	creates text for graphic display
TPARAM	sets the parameters for texts in graphics
TRACE	displays strips
UNBLANK	restores the strips cleaned with blank to their original state

MISCELLANEOUS

CALCUL	activates a pocket calculator
DIR	gives the directory of selected drive
EDIT	activates the selected text editor
MACRO	gives access to a macro command

PLOTTER

PAGE	ejects the page on the plotter
PLOT	activates the plotter
PLOTOFF	turns plotter off
PLOTOPTION	allows to select plotter features

STRIP MANIPULATIONS

BLANK	selective removal of part of the data in a strip
COMPUTE	permits to create strips and variables
DIR	gives the directory of selected drive
FFREE	flushes the function tree
FLIB	lists the function in LIBRARY.FUN
FLUSH	clears the heap memory of all strips
FROOT	displays the functions in the function tree
LISTSTRIP	displays the content of a strip
MEMORY	checks the available memory space
RECALCON	set flag for automatic recalculation of interdependent strips
RECALCOFF	cancel the above flag
ROOT	lists the strips in memory (interupt display with ctrl-s)
SMOOTH	apply a smoothing on a strip
SORTING	puts y-strip in growing order of the corresping x-strip
STRIP	sets the parameters of a strip
TRACE	displays strips
TROOT	lists the text strips in memory
UNBLANK	restores the strips cleaned with blank to their original state

I
INTERACTING WITH DATAC and DATACP

GETTING STARTED

To operate DATAC, commands have been implemented, which are listed in Fig. 1 under several headings. The headings are there to help you in your selection of a command for a particular purpose. Note that a command may give access to entire sets of interactive inputs which are not visible in this command list.

NOTE: DATAC has facilities for displaying and plotting curves that fall into two categories: the simple routine display which requires minimal input from the user and the more elaborate procedure which produces high quality figures for publications. The latter requires more input from the user and will be described in a separate section of this manual.

TO ACTIVATE DATAC type datac (cr), a logon text will be displayed for a few seconds, then the program will ring a bell, erase the screen and display a @, indicating that it is waiting for any command. TO SKIP THE LOGON, wait a few seconds after having activated DATAC and type Ctrl-C.

TO QUIT DATAC, simply type quit (cr).

Datac Initialization

Datac reads a file (if found) called datacp.ini assumed to be in the current directory. This file contains information about directory paths for data, strips, macros, and stimulus files, as well as information about the hardware configuration.

Command Interface

DATAC accepts commands from a command line prompt, much as a DOS program would. DATAC commands can be abbreviated. Note, however, that several commands begin with the same letter or couple of letters. If there is an ambiguity, the program will choose the first corresponding command it its list.
You can activate any command wherever you are in the program.
You can nest_several_commands on a line. For example, having set some parameters in strip you can enter:
erase trace x y ptext 1; 5 (cr)
This will cause the program to erase the screen, then to trace y as a function of x and finally to superimpose on the graph 5 lines of text.

NOTE that (cr) is equivalent to ENTER

To display the list of general headings while the program is running, type help (cr). To see the commands corresponding to a particular heading, type the first letter of this heading followed by "help(cr)".

The command line editor permits command line editing, a command history buffer, and provides enhancements to the macro interpreter.

Command Line editing

During the entry of a command line, or when one has been recalled from the history buffer, a command line can be edited. The left and right arrow keys move the cursor left and right one character at a time. CTRL-Left and CTRL-Right move the cursor to the previous or next word, using general punctuation to define word boundaries also (i.e., ();:,/, etc.). Insert and delete change the typeover mode of the following keystrokes. Home and end position the cursor at the beginning and end of the line respectively. CTRL-end deletes from the current position to the end of the line. The escape key clears the entire command line.

Command History buffer

As commands are entered, they are saved to a circular history buffer that currently stores the last 50 commands. Commands can be retrieved from the buffer using the up and down arrow keys to move through the buffer. The up-arrow key retrieves the most recently entered command. When the command line is retrieved, it is displayed on the screen at the command prompt. The line can then be edited, escaped, or executed with the enter key. The modified command will be updated and the buffer pointer remains at this new position. The F-10 key clears the command buffer. Occasionally a warning appears on the screen to indicate that the command buffer is empty (if you attempt to move through it and no commands have been entered). You may ignore this.

Command Entry with Parameters

Commands may be entered either singly or as a group. If they are entered singly, then when additional information is needed by the program it will prompt for the appropriate data (to the extent that the new routines have been implemented, this applies to most of the code. Report (for the JHU version of DATAC, to P. Manis) any locations where this does not occur so it can be fixed. If the commands are entered as a group, DATAC will execute them to the extent that it can; if there is an erroneous command an error will be issued and the command line aborted (e.g., remaining commands on that command line are not executed). The command can be edited and reissued using the history buffer recall and editing the line to produce an acceptable command.

Numerical arguments to commands

All commands (except those in macros specifying the arguments of a LOOP) accept a variable (e.g., n[4]) instead of an expected number. The number is replaced with the contents of that variable. This is useful for setting values in strips or blanking commands, as one example. This is also most useful in macros.

II
ACQUISITION OF ELECTROPHYSIOLOGICAL DATA

A few words of introduction are needed here to explain briefly the organization of a data_acquisition_file in DATAC. A more detailed description is given in a separate manual. Note that the data_organization could be different and that it simply reflects our choice in our own application. It could be changed by user familiar with programming (see Programmers Manual). A data file contains the stored data in the form of records of fixed dimensions. All records in a given file have the same dimension. The user is prompted to give the size of the record at the beginning of a data acquisition session. This record_size can vary (in powers of 2) between 256 and 2048 points per record. Which size should you choose? It depends upon the type of event your are recording and on your available storage space. Suppose you have a floppy disc with 1.2 megabyte strorage space and you choose 2048 points per record. You will find out below that each record will actually occupy 16384 bytes. Therefore the diskette will only hold approximately 75 recordings. On the other hand if you choose 256 pts/rec the same diskette will hold 580 recordings. If you want best possible time resolution combined with the longest possible recording time you will choose 2048. As an example, if your data is sampled every 40 microsec (25kHz sampling frequency), you will be able to store 80 msec of data. Under these circumstances a 1 msec action potential would consist of 25 data points. Clearly, if you are recording slow processes, you do not need that type of time resolution. There, you can skip points during the A/D conversion and you can choose a smaller number of points/rec. In any case, you have to make these decisions before you activate DATAC in the data acquisition mode because the program will ask you to enter the number of points per record you whish.
To acquire data with DATAC you need to activate the program in a special way:

```
DATAC dev:filename.ext 1

(ex: datac b:31-12-85.bbs 1)
```

If the file exists already, DATAC will not re-open it. You will have to start over again with a new file name. This was done on purpose; getting good data on disk is not always easy. Therefore we thought that a total overwrite protection might be a good idea. If the file opening was successful you will be asked to enter the number of points/record (see above). Enter that value (256, 512, 1024 or 2048). You will then be prompted to enter one line of comments (80 characters). If you do not feel like writing anything simply type (cr). Note that this line_of_comments can be helpful later in finding out what type of data are in the data file. Indeed, when you activate DATAC in the replay mode this line will always be displayed. The program enters then into the interpreter loop, where all commands can be accessed. The data_acquisition_commands per se exclusively consist of DATA and AVERAGE.
In the data_acquisition mode, the program remains in a loop and acquires data until (CTRL-C> is typed. The data are stored on the device on which the file was created and the current record number is automatically displayed on the screen.
For data_acquisition_activation type

D (cr)

For data_acquisition_interruption type

CTRL-C (cr)

DATA_CANCELLATION CLEAR_command

Sequential records, starting from a given record number and ending at the last record stored can be cancelled with the clear command. This can be performed exclusively during data acquisition. The command is:
clear (cr) (abbreviated: c (cr))

The program will ask for the number of the last record that should be kept on the disk. Type this number followed by (cr). All records up to and including this last number will still be safe on the disk.

RECORD_NUMBER NUMBER_command

The number of the last record during data acquisition can be displayed by typing
number (cr)

AVERAGING_ON_LINE AVERAGE_command

When you activate this command the program will ask you how many records you want to average. Enter the corresponding number and DATAC will perform the average and store exclusively the result on the disk. This may be useful to save disk space during an experiment. The averaging can be interrupted with CTRL_C.

OPENING_A_DATA_FILE

OPEN_command

To replay electrophysiological data stored in a file it is necessary to open the file. To open a file, type open (cr)

and then on the request of the program

dev:file name.ext (cr) (ex: b:31-12-85.bb)

Three new commands have been added to open, close and provide a default extension for data acquisition files. These obviate the need to exit and restart DATAC every time a file is to be closed, and permit one to start DATAC without opening a file. Data cannot be acquired without opening a file however.

aopen filename: opens the file filename with the optional extension set by aext (otherwise extension will be ".dat"). Checks for existence of file first; files cannot be overwritten. Note: aopen used within the pulse module without an argument on the command line will effect an automatic file name generation. See the pulse module description.
aclose: closes the currently open acquisition file; blocks data acquisition.
aext extension: sets a default extension to be used for aopen or open (reading). The extension can also be set by the datac.ini file.

Abstract

The pulse generator control module allows the DATAC user to control the delivery of voltage and current clamp command signals. Depending on the hardware configuration specified in datac.ini, pulses may be generated by the Geneva Pulse Generator board (two simple single pulse lines), a QuaTech WSB10 waveform synthesizer board, or by a CYCTM10 board. Patterns and amplitudes of pulse delivery may be defined by commands in a "macro" control file, or a built-in sequencer, for use in rapid, semi-automatic data acquistion, or may be controlled from the keyboard during running acquisition.

Activation of the pulse module is caused by the command "pulse" at any command level in DATAC. This command brings up a screen indicating the current settings of the various parameters controlled by the pulse generator module. These settings are divided into four main parts: the base timing, the voltage-current clamp protocol, the pulse command protocols, and the "shock" or waveform synthesizer protocol. Each protocol has a specific set of commands within the pulse module, as described below. Additional commands control the way data is acquired during the automatic acquisition "macro" mode. Thus, the description of commands is broken into appropriate sections below.

Function keys are active for the pulse generator, and these return command equivalents as described with each command. A listing of the function key mapping can be obtained with the command "function".

1. Basic Timing Control

The timing control consists of two commands.
Cycle controls the cycle time per stimulus epoch or total pattern. Cycle takes one argument, the cycle time in msec. Cycle times are constrained to be between 1.0 and 65,000. msec, but should always be longer than the sum of (dh+d1+d2+d3) in the clamp protocol (see Section 2) or delay+duration of pulse protocol (see Section 3), for reliable pulse outputs.
Clock controls the basic clock timing. The default value is 1.0 msec, and it is recommended that the user not modify this value. The clock can be set to any value between 0.1 and 10 msec in special cases; all other delays and durations are automatically recomputed to fit this new clock value.

The cycle source is set by the software according to the board configuration in use. The preferred configuration is to use the CYCTM10 as the cycle source (requires hardware modification to the geneva board), but the Geneva board or a DMAX-54 board can also be used.

These values are displayed on the second line of the screen, underneath the file name.

2. Basic Acquisition Control

The acquisition control parameters are displayed on the next line of the screen. These include the number of sampled channels (set in datac.ini), the sample rate, the calculated epoch, the amplifier gain and low pass filter settings.

Rate controls the sample rate. The default timing (in fast mode) is in microseconds, expressed per channel (thus, 50 µs with 2 channels results in each channel being sampled every 100 µs). In the slow mode (see below), the rate is expressed in milliseconds per point, extending the sampling duration to minutes per trace (watch out for Nyquist limits, however!).

The epoch is the duration of the trace that will be collected with the selected sample rate, based on the number of channels in effect and the number of points per trace. This value is valid only when an acquisition file is open; otherwise it assumes a default value of 1024 points.

Gain is the amplifier gain, as Vout/Vin. If the gain is 0, then no value is assumed. The gain factor is used in calculating the current/voltage measured between samples, and is listed in the note file for each record if it is not set to 0. If the amplfier is an Axopatch200, and the gain is not 0, then the program will force you to set the amplifier gain to the desired value, and reads the gain telegraph from the amplifier to insure that the gain switch is set correctly.

LPF is the amplifier low pass filter setting, in kHz. This operates the same way as the gain command above.

3. Pulse Outputs

The pulse output controls are listed over the next section of the display, and the number of channels will depend on the hardware configuration. The boards are "installed" in a specific order. Thus, if they are present, the Geneva board will appear first (as channels 0 and 1), the CYCTM10 board will appear next (e.g., channels 2 through 9), and the WSB board will appear last (display depends on mode of use of this board, see below).

Each board is capable of specific types of output, as displayed in the table. Across the top of the table are the parameters available generally. For each channel, only those parameters that can be rationally changed are displayed. The boards may be divided into 3 categories:

TTL one shots (CYCTM10, DMAX54): These can generate only 5V output pulses with a specified delay and duration. They have only delay and duration column parameters, with the exception of the last 2 channels of the CYCTM10 board, which are wired as train generators. These can generate independent trains with specified interpulse intervals, delays, pulse duration, and numbers of pulses.

Analog one shots (GENEVA "Bert"). This board generates one shot pulses, however the off voltage and the pulse voltage levels are adjustable (level and offset parameters). In addition, there is a software scaling parameter so that the output can expressed in units other than the output voltage of the board.

Analog waveform generator (WSB10). This board can generate a triggered 2048 point analog waveform. Software allows it to run in 4 modes:

(1) "Shock" mode generates 100 µsec pulses of specified amplitude and duration; multiple pulses may be generated also, as specified by the number of pulses and the interpulse interval. A limit is that the pulse train must be shorter than 205 msec in duration from the triggering command. The output is calibrated in terms of the 100V output of a DAGAN isolation unit.

(2) Voltage clamp command mode generates 4 levels (a holding level and 3 steps) of adjustable delay (for the holding level) and duration (for each of the steps). A scale factor relates the board output to command levels at an amplifier.

(3) Current clamp command mode is similar to the voltage clamp mode.

(4) Ramp mode is a modified version of the voltage clamp mode. The voltage ramps between the s1 and s2 levels over the duration set by the s2 value. This generates a linear voltage ramp.

3.1 General control information.

Each channel of the pulse can be set with a command line of the form:

    parameter channel value

where:

    parameter is the control parameter (delay, duration, level, offset, scale, ipi, np).
    channel is the channel number or the channel alias name.
    value is an appropriate numerical value for the parameter (limits are usually checked to prevent out of bounds operation).

Each channel may (and should) be given an alias so that it can be accessed in a ready fashion. To set an alias, enter:

    alias channel# aliasname

3.2 Single line sequencing

The command "sequence" permits the rapid collection of parameteric data once basic timing has been set. The use of this command can reduce the number of macros that one might maintain on disk as well as permits some flexibility during the experiment as to how the data will be collected. The syntax for "sequence" is as follows:

sequence variable valuelist acquisition

where:

    variable can be any of the adjustable paramters that pulse controls (such as dur 1, lev 4, s1, d1, etc.).
    valuelist is a sequence of the form a;b/xc (see Macro Extensions, above) that lists the values that the variable will take. Floating point values are permitted. Random presentation (always the same random sequence, however) can also be used (use the /nr switch; see Macro Extensions).
    acquisition is one of the following:

take n: sample and store n individual records for each entry in valuelist.
average n: average n records and store result for each entry in valuelist.

A typical command might be (note abbreviations):

```
seq l1 -1000;1000/100 t 1
```

Sequences may be edited (using the command line editor) using the set command (no arguments). This displays the current sequence on the command line and lets you edit it. Enter a <CR> to update the sequence with the edited version.

Hitting Control-C during a sequence will terminate data colletion.
Sequences are stored with the stimulus files, so that a common protocol can be easily executed.

Sequence is also available at the main command level of DATAC and leaves you in the pulse generator when it is done.

4. Analog Outputs/Shocks (WSB Board specifics)

4A. Voltage/Current clamp protocols

The voltage/current clamp protocol controls a single 12-bit DAC which is remote to the board (i.e., it can be located near the amplifiers). The DAC is programmed to put out four different voltage levels in sequence. These are referred to as the holding value, and step1, step2, step3. At the start of a pulse generator cycle, the voltage output is set to the value in step1 (s1), for the time set by the duration 1 (d1), followed by the value in step2 (s2) for the duration 2 (d2), followed by the value in step 3 (s3), for the duration 3 (d3). The voltage is then stepped to the value set as the holding level (ho) for the remainder of the cycle until the start of the next cycle.

The clamp protocol has two control variables in addition to the timing and voltage values indicated above. The commands on and off turn the protocol on and off respectively. In the OFF state, the output value in the DAC is always set to the holding value. In the ON state, the specified sequence of voltage steps is sent to the DAC.

The commands vclamp and cclamp change the status indicator, and change the scaling factor that is used to scale the outputs appropriately. VCLAMP causes the vscale factor to be used in computing the output voltages. Thus, the command values can be tailored to correspond to (for example) mV command steps at the electrode. The selection of the vscale value will depend upon the command voltage gain of the amplifier being used (i.e., input command signal to mV electrode command voltage). The default value of 1.0 corresponds to a full-scale output voltage of 5.12 V, where the voltages in ho, s1, s2, and s3 read directly in mV output. A similar but separate scale factor is used in the CCLAMP mode, cscale. Appropriate setting of the cscale factor allows one to specify the command levels in terms of pA or nA for the current commands.

Use of the clamp module requires that the remote DAC board be built and connected to the pulse generator board.

4B. Shock protocol

The shock protocol is implemented using a Quatech WSB10 waveform synthesizer board. The protocol sends out a series (or one) pulse of 100 usec duration. Voltages are scaled to correspond to the output of a DAGAN optically isolated stimulus unit, set on the 100 V output scale. The maximum duration of a train or pulse interval is limited to 204.8 msec, which corresponds to clocking the full WSB10 buffer out at 100 usec intervals. Longer durations could be acheived at the expense of some external hardware. Commands are as follows:

vo: (output voltage, volts). Takes one argument, the output voltage in volts. Minimum voltage step size is 0.067 V at output of DAGAN unit. A voltage of 0 turns the shocks off.

vdel: (delay to first pulse, msec): Takes one argument, the delay to the first pulse, in msec. Valid values are 0.0 to 204.7 msec.

vpi: (interval between voltage pulses, msec): Takes on argument, the interval between pulses, in msec. Valid values are 1.0 to 200.0 msec.

vn: (number of pulses output): Takes one argument, the number of pulses to be output per cycle, in integer format. Valid values are 1 to 64. If the value (vdel + vpi * vn) is greater than the 204.8 msec maximum, the pulses are clipped at that time. Therefore, some intelligence on the part of the user is necessary when setting this value.

4C. Arbitrary Waveforms

Arbitrary waveforms may be generated on the WSB10 board by specifying a file using the wget (waveform get) command. This command accepts one parameter, a file name with a default extension of ".wav". The file is assumed to be one created by the program wavgen. The waveform is read from disk and loaded into the WSB10 board. While the waveform is active, the filename will continue to be displayed on the screen. However, if the user specifies some shock parameters, the waveform will be overwritten, and the filename will
disappear from the screen. Use of this command requires the WSB10 board.

Arbitrary waveforms can now also be computed within DATAC and loaded directly to the board. Ask me how to do it.

5. Control Functions

The remaining commands are used to control data acquisition in macro files, to save or retrieve a set of parameters as a group, or to initiate data acquisition.

da, data: Initiate data acquisition in "running" mode, exactly as if command was executed from any other level in DATAC. Acquisition is terminated with a control-c (^C).

ave: Take acquisition as an average on N samples. Takes only one argument, N, which must be between 2 and 512. In a macro file, ave N takes the average, and cancels the normal single record taken at the end of a line.

wait: Takes one argument, the number of seconds to wait through before taking the next record.

take: Takes one argument, the number of single traces to be sampled.

until: takes one argument, the time (in seconds) for a data acquistion to continue until. Call: until 4800, will take records at the current settings until 4800 seconds have elapsed (on timer 0).

put: saves the current "environment" or all of parameters controlling the pulse board in a file. (Changed from "save" which caused conflict with strip save/retrieve at main DATAC command level). Takes one argument, the name of a file. Default extension is ".stm". If the file exists, asks if ok to overwrite. Usage in auto mode not recommended.

get: retrieves an "environment" or all of the parameters controlling the pulse board from a file. (Changed from "retrieve", for same reason as put). Takes one argument, the name of a file. Default extension is ".stm". Usage in auto mode IS recommended.

do: saves the current stimulus environment in a temporary file (tempexec.stm), retrieves the specified file, executes it's sequence, and then returns control to the tempexec.stm file.

edit: invokes the PS screen editor. Consult the PS manual or the on-line help for the editor. Will not work if insufficient memory due to the number of strips in memory.

6. Digital output/valve control

The commands v0,v1,v2,v3 select fluid control solenoid valves connected to bits 0 and 1 of the output port (formerly the dac port) of the pulse generator. v0 turns the valves off. The other commands select valves 1,2 or 3 (using a 3-way normally closed fluid control solenoid, such as manufactured by General Valve).

The command "digword" sets the remaining bits in the digital port: these are reserved for future use (controlling input data sources etc.).

III
DISPLAY OF ELECTROPHYSIOLOGICAL DATA

This section describes the functioning of the ordinary display procedure for electrophysiological data stored on disks. The display procedure is described in the section display and the parameters involved in the display are described in the section parameters.

DISPLAY_OF_DATA   DISPLAY_command

The command DISPLAY displays one or several recordings stored in a data file, either during data acquisition or during off-line replay. Display is not used for displaying users created strips (see Ch. III). Several data curves can be displayed simultaneously. To display voltage and/or current, check that vsel and /or isel are set to 1 (see info in parameters, below).
Typical examples of display are:

disp 3 (cr):   record 3 will be displayed

disp 1;5,7 (cr):   records 1 to 5 and 7 are displayed. The semicolon means that all records between two boundaries will be taken.
NOTE: data on disk can be directly subtracted (see below, for info). All parameters for the direct_display are controlled by parameters.
IMPORTANT: when you are displaying_data_simultaneously, you must make sure that the data where actually recorded in the same conditions (same frequency_of_acquisition). If not, this may be the cause of some funny displays. If in doubt check the number of points that are being displayed in each data with ROOT (see this term). Note that DATAC will allow you to compare data acquired at different times and in quite different conditions of acquisition but to do it safely you will have to use the strips and the operator INTERPOLATION (see Ch. IV).

It is also important to understand that EVEN for the DIRECT DISPLAY command described here, strips are actually created. After a display command, the data are read from the disc, stored in temporary_strips (each has a number as label), which eventually are displayed. In successive display commands, DATAC will overwrite these temporary strips. But if you give one command with many curves to display there will be many temporary strips created. You should be aware of that when you require a display of a large number of data curves. The three consequences of this request are: (1) it will take time to read the data from the disk, (2) you will load up the memory with many temporary strips and (3) there will be a general confusion on the screen (there are only 7 colors available).

PARAMETERS CONTROLLING DIRECT_DATA_DISPLAY PARAM_command

The parameter_command PARAM (cr) gives access to the parameters controlling the display of electrophysiological data stored in a file. Since in electrophysiological experiments the abscissa is often the time and the ordinates the current and/or the voltage, parameter controls the time displayed and the I and V axes (including tick marks on the 3 axes). In addition, it controls the color and markers on I and V curves. To set any of the variables in parameter, you must first type parameter (cr). The program will display a table of the present values of all variables (Fig. 2). Type the variable you want to modify followed by the new value. For example, imin -180. After (cr) the new value will appear in the table.
REMEMBER THAT YOU CAN NEST YOUR COMMANDS!

For example you can perfectly write:

imin -150 imax 300 istep 150 icol -1 erase dis 3

on the same line. All parameters indicated will be changed and eventually the screen will be erased and then record 3 will be displayed.
Note also that in strips, colors and markers are controlled independently of parameter (see the info on strips in chapter III).

FIGURE 2
DISPLAY RESULTING FROM ACTIVATING THE PARAMETER COMMAND

```
Parameters>


tbeg=      0.00 tend=    100.00 tstep=      0.00 tname ms tfmt %6.1f

vmin=   -100.00  vmax=      0.00   vstep=      0.00  vname mV
vsel=  0     vsmo=    0  vcol=  1   vgain=20.010 refvg = 1.000
vmark= 0     vmsi= 1.00  vskip=    0   vfmt=%6.1f
vazob=      0.00  vazoe=      0.00  vchg= 2  vchan= 0

imin=     -1.00  imax=      1.00  istep=      0.50  iname mV
isel= 1      ismo=    0  icol=  1   igain=1000.000 refig=1.000
imark= 0     imsi= 1.00  iskip=    0   ifmt=%6.1f
iazob=      0.00  iazoe=      0.00  ichg= 2  ichan= 1

wmin=   -100.00  wmax=      0.00  wstep=     20.00  wname mV
wsel= 0      wsmo=    0  wcol=  1  wgain=20.010 refwg=1.000
wmark= 0     wmsi= 1.00  wskip=    0 wfmt=%6.1f
wazob=      0.00  wazoe=      0.00   wchg= 2  wchan= 2

data file ext: .pbm
data: ..velop\datacp\data\ stm: stm\                 str: str\
mac:  mac\                 msb:                      txt:
```

The various commands of param will be described now.

Display_duration

TBEG, TEND, TSTEP parameters

The time_axes begins at tbeg and ends at tend. Both values can be set by the user. tstep determines the interval_between_tickmarks. By default (tstep=0) the axes is automatically divided in 3 intervals. IMPORTANT: the direct display of a data curve is always between tbeg and tend! This is not necessarily so with strips.

Selection of current and voltage tracing

ISEL, VSEL, WSEL parameter

The parameters isel and vsel permit to select current_display and/or voltage_display when they are set to 1. For displaying the current only, set isel to 1 and vsel to 0.

Smoothing_data **ISMO**_parameter **VSMO**_parameter

The smoothing is a moving_average on a number of points set by the user. To smooth the current on 10 points, set ismo to 10. Vsmo controls voltage smoothing. When both ismo and vsmo are set to 0, no smoothing takes place. Note that smoothing is executed just before the display on screen. This avoids the possibility of unwanted cumulative smoothing. Note also that a smoothing is analogous to a filtering_of_the_data. Consequently data will be shifted just as with any filtering procedure.

Axes_control
IMIN_parameter IMAX_parameter ISTEP_parameter
VMIN_parameter VMAX_parameter VSTEP_parameter

The time axes has been described above (see display duration). Current and voltage axes are defined by a minimum value imin(pA) and vmin(mV), and by a maximum value imax and vmax. The parameters istep and vstep control the interval between tick marks on the corresponding axes. If istep or vstep are 0, the intervals are set by default (3 intervals on one axes).

Color_code VCOL_parameter ICOL_parameter

The parameter color controls the color of the curve to be displayed. Voltage and current curves can be controlled independently by vcol and icol, respectively. vcol and icol function according to the same rules. When icol (or vcol) is > 0, all the current (or voltage) curves displayed in a graph will have the same color. The color is set by a number between 1 and 8 according to the following code: green(1), red(2), blue(3), yellow(4), magenta(5), cyan(6), white(7) and black(8).
Example: vcol 2 (cr) sets voltage color to red.

When icol (or vcol) is < 0, there is an AUTOMATIC_COLOR_CHANGE starting at the color defined by the number. For example, if vcol is set to -4 the first voltage curve will be yellow and the next magenta etc.

Symbols_on_curves
IMARK_parameter IMSI_parameter ISKIP_parameter
VMARK_parameter VMSI_parameter VSKIP_parameter

The data can be traced as curves without symbols_or_markers, as sequence of symbols alone or as curves with superimposed symbols. Current markers are controlled by imark, imsi and iskip, which set the marker type, size and interval, respectively. The corresponding controls for voltage markers are vmark, vmsi and vskip.
Marker_type
The parameter vmark (or imark) selects the type of marker for the voltage (or the current) according to the DISSPLA code: square(0), octogone(1), triangle(2), +(3), x(4), diamond(5). To set a marker, triangle for example in the voltage type: vmark 2. The default value will be 0 (square).
Marker_size The parameter vmsi (or imsi) controls the markers size. To set it type (for a voltage curve) vmsi followed by a value. Values of 5-6 are usually adequate.
Marker_interval
The parameter vskip (or iskip) controls the interval between markers. In addition, the sign of the parameter determines whether the symbol will be traced alone(-) or superimposed on a curve. To set the interval for the voltage, type vskip followed by a value. For example: vskip -10 (cr) induces the tracing routine to skip 10 points between voltage markers and to trace the markers exclusively.

SUBTRACTION OF ELECTROPHYSIOLOGICAL DATA STORED IN A FILE

The SUBTRACT_command allows you to directly subtract experimental curves. The procedure is as for display. Example: To subtract curve 3 from curve 1, type:

```
subtract 1,3 (cr)
```

The status of isel and vsel will decide whether the current, voltage or both are being subtracted. Series of subtractions can be performed: subtract 1,3,7,10 causes 3 to be subtracted from 1 and 10 from 7. Obviously you need and even number of data. Subtraction of data can also be performed on strips (see chapter IV).

DATA_READOUT, CROSSHAIR_TO_READ_DATA CROSS_command

The crosshair_command cross (cr) activates the crosshair. This permits for example to read a current or voltage value on a graph. After activation of the command, select the v or the i for voltage or current reading and the crosshair will appear on screen. Move the crosshair to the desired place with the joystick (on the IBM PC-AT, use the arrow keys). For slow crosshair displacements press simultaneously on the shift key. When you wish to make a reading type any key. The program will display the x and y position of the crosshair at the upper right corner of the screen. Press q to quit the crosshair.
NOTE that the reading of the crosshair can be saved in two strips named xcur and ycur if you selectively type on the key S for your readings. For further information on this facility, see Ch. III, under CROSSHAIR WITH STRIPS.

TEXT_CREATION_ON_GRAPH

This topic is described in Ch III. The same procedure applies whether a graph is obtained through direct display or through tracing strips. So, whatever is said there is directly applicable here.

CLEAR_GRAPH_AREA ERASE_command

The command ERASE (cr) or more simply E (cr) clears the graphic area on the terminal. This command clears the axes and the curves and also the text entered with the text command.

CLEAR_DIALOG_AREA   ERADIAL_command

The command ERADIAL (cr) selectively erases the dialog without erasing the graphic area. It can be used when you dont want to retrace a graph that was traced before (remember that you can use graph to display again a graph previously traced, see below, quick redisplay).

CLEAR_TEXT_IN_MEMORY ERATEXT_command

The command ERATEXT (cr) clears the strips of text created with the command text (see this term).

QUICK_REDISPLAY OF A GRAPH GRAPH_command

The command GRAPH (cr) permits to restore a graphic display that was previously on the screen but was blanked to allow an alphanumeric_report (example after a root command). NOTE: this command will not restore a graphic that has been erased with the command erase!

IV
STRIPS AND THEIR USE

A strip is a stucture made up of a series of values (theoretical or experimental),of the equation that was used to create it and of several parameters that will control its display. A strip is created with the command compute and its parameters are controlled by the command strip. The various commands described below are important for the creation, display and manipulation of strips. As you will see in Ch VI, strips can also be created in a different way (READ and RETRIEVE commands).

STRIP_CREATION, STRIP_MODIFICATION COMPUTE_command

The command COMPUTE (cr) permits to create_strips. It is important to understand that you are basically writing equations that are completely interpreted by the program. Equations can be corrected or modifyed in video_mode. A strip has a name. The STRIP_NAME has 8 characters and must begin with an x, y or z. The remaining characters can be numbers or letters. A strip can be loaded with theoretical data, with experimental data by hand or directly from data stored on disk. Strips are useful when you want to manipulate the data (subtraction of constants, multiplication by a scaling factor, shift of the time base, etc). Another useful application of strips is comparison between data and theoretical curves. Note that operations between theoretical strips and data are possible. Parameters of the strips are set with the command strip (see below). Each strip can have its own set of parameters.
To create a strip, type
compute (cr)
the program will display a * and is now waiting for you to write an equation of the form

y=10*(1-exp(-x/20)) (cr)

at the (cr), DATAC will calculate y, provided you created the x- strip before. Otherwise y will simply be created and you will have to calculate it later (once x exists) by typing y (cr). DATAC will display the equation above and calculate it if you type a second (cr). You will see below how to create a theoretical x-strip if you do not have any data. See also DEMO manual.
NOTE: 1) any arithmetical operation performed on a strip is actually executed on every single value present in the data array of the strip. Suppose that a strip x contains the data 1,3,5,7,9. If you write: x1=3*x
x1 will contain the data 3,9,15,21,27.

2) the command compute is also used to initialize variables (see Ch. IV).

STRIP_AND_ELECTROPHYSIOLOGY COMPUTE_command

To load a strip labelled y with voltage values of record 23, activate compute and then type y=v23 (note that v determines that the strip handles the voltage part). For loading current values of the same record, type y1=i23. NOTE: the voltage (and current) values will be loaded over the time interval defined by tbeg and tend (command PARAMETERS, see ch. II). To create a strip with the time basis for tracing the data, type x=t23. In the strip x the values of time of record 23 are stored between tbeg and tend. THIS IS AN IMPORTANT FEATURE: suppose the interesting part of record 25 is between 5.5 and 20 msec, all you need to do is set tbeg and tend to these values before you say x=t25.
currents are in pA, voltages in mV

Here are a few examples of operations on data strips:

Shifting_the_timebase

Suppose you recorded over 3 sec a current (say record 142), which first increased over 100 msec and then decayed. Your goal is to trace exclusively the decaying current and you want the to place the beginning of the decay at 0 msec. The way to proceed is as follows:

1)set tbeg at 100 and tend at 3000. The first 100 msec of the data will not be loaded in the strip. So you will see only the decay.
2)type compute and write a strip x=t142-100. Now you have a time base for your data that effectively starts at 0 msec (remember that the decay in the real data began at 100 msec)
3)write a strip y=i142. This is the current strip loaded from 100 msec on. But the first point in the strip will be plotted as a function of the first point in the time base x, i.e. 0 msec
4)set tbeg to 0 in param and trace x,y . This is due to the fact that the ctype of y here is i (see ctype below); for strips of the i or v ctype the command trace uses the parameter in param for the axes.

Subtract_a_constant current (say 124 pA) from rec 1:

y=i1-124

Multiply_by_a_constant:

y18=i18*3.25

Operations_between_strips:

y4=(i1-i2)/i3

Operators such as sin, log, ln, etc can be applied to data strips: ln(y4-y18). NOTE: the equations you wrote can be displayed at any time by root. Mistakes are easily corrected. Type compute and type the name of the strip you want to correct (say y4), the equation will be displayed and you can correct it (you can delete, insert and move the cursor (see in the INDEX under cursor position)). Whenever strips are compared or operations between strips are performed it is important to check whether the data where acquired in the same way. Remember that depending upon the type of recording set up you use the frequency_of_acquisition can be changed within one data (to improve resolution during a spike) or between data. Displaying a strip with the time base of another strip is perfectly possible but just make sure that it is also legitimate. If there is a timebase_missmatch you can still compare the two data but you will have to use a trick (see INTERPOLATION in Ch. IV).

THEORETICAL_STRIPS COMPUTE_command

It is easy to create theoretical strips. A first step is often to create a time base. Example : x=time*20 (or x=20*time) will allow you to display your curve on 20 time units (for additional details TIME see below). Then any y-type strip can be created. For example: y5=10*sin(3*x). A more elaborate example:
x=3000*time and y7=100*exp(-x/120)+50*exp(-x/3600).

To check your equations at any time, type root. If the display takes more than one screen, type ctrl-s to interupt the display and ctrl-s again to restart it. To correct the strips see next section below (STRIP CORRECTION) NOTE: if you want to perform operation between equations and strips of data, make sure that they have the same curve type (see info on c-type). If not you can set it. It is YOUR RESPONSIBILITY to make meaningful operations.

Theoretical_abscissa   TIME_command

The operator TIME allows you to create a strip of 512 values comprized between 0 and 1. The number 512 is the number of points we assigned to a strip. time can be used conveniently to create_a_timebase. For optimal display, multiply time by the desired duration of the display. Example, to display 50 msec: x=time*50. The result is a strip x made of 512 values with an time interval between successive values of 50/512 (0,09765 msec). Although it is called time, this operator can also be used to create_any_abscissa strip (concentrations, apples, etc).
If you wish to create a theoretical_voltage_axes from say -100 mV to +50 mV, write: x=-100+150*time

Strip_with_a_constant CTE_operator

If you want to set all the values contained in a strip to a constant, say 0 or 350, you CANNOT simply write in compute
y=0
or
y2=350

If you do that the program will tell you WRONG EQUATION because you are trying to equate two quantities which are not alike, a strip on the left with a scalar value on the right. The way to proceed is to write in compute

y=cte0 or y=0cte
or
y2=cte350 or y=350cte

The letters cte indicate to the interpreter that you wish to put a constant. In our examples strip y will be filled with 512 0 and strip y2 with 512 350.
using preset_FUNCTIONS_in strips

You may want to read this section when you get a bit more familiar with DATAC, since the description of this useful tool requires frequent references to features that have not yet been described. In compute, you can define_your_functions which can be saved and used at any time to create strips. This is a way to accelerate the writing of equations that are often used. It reduces the mistakes and allows you to actually enter more than 80 characters in an equation. In compute a strip can only be writen one one line (i.e. 80 characters) but by using one or more functions_in_a_strip you can indirectly increase the total number of characters.
Functions_are_saved in special file called library.fun, from which they can be retrieved. The interpreter will recognize that you are

creating or using a function whenever it encounters a capital_F. The syntax is as follows: F???????(*,*,*,*,*,*,*)=mathematical expression
The ? (up to 7) are optional and constitute the functions'name (Fgauss, F23, Ftest69 would be legitimate names). The * (up to 7) represents the function_arguments that are defined in the mathematical expression (a single argument cannot exceed 20 characters). Please note that nothing is assumed about the nature of the argument in a function. For example an argument called x is not necessarily a strip. It can be a strip or a scalar or even another function depending upon the nature of the argument you use when you introduce the function in a strip. Note also that the mathematical expression can contain more than one function. When a strip is created with one or more functions, these are expanded and the calculation is performed with the appropriate arguments. The result is then put in the strip. Here are a few examples:

F(x,y)=(1-exp(-x/y))

This function can be used in this way

ytest=20*F(x,25)

Here you see that x is used as a strip and y as a scalar (25). The interpreter will expand the expression as

ytest=20*(1-exp(-x/25))

Let us take a more elaborate example

FG(a,b,c)=(.39/a)*exp-(.5*(((b-c)/a)^2))

This is the expression for a gaussian distribution, with a the standard deviation, b the variable and c the mean. You could use this function to fit a set of data observed at a series of values of x contained in a strip x. For this you would create a strip yth as
yth=F(n[0],x,n[1])
where n[0] and n[1] are variables that will be used in the fit (see below). At the end of the fit variable n[0] will contain the value of the standard deviation and n[1] that of the mean. Note that functions can be nested. You can write
F(x)=F1(a,F2(a,b),c)
Simply, be aware of what you do. The program is protected against recursivity (which could produce an infinite loop) in that it will automatically stop the calculations after 10 turns. The functions are stored temporarily (like the strips) in a binary tree. You can display them by typing FROOT. If you wish you can save a function definitively by using FSAVE. This will cause the storage of one specific function or of all the functions present in the tree to be saved (same basic procedure as that described Ch. VI for save) in a file called LIBRARY.FUN (described in more details below). This file is organized in such a way that you can add comments with EDIT (see this term) to describe the function. The functions present in this file can be displayed in DATAC with FLIB. They can be retrieved individually or collectively with FRETRIEVE (same basic procedure as that described in Ch. VI, for retrieve). The function in the tree can be released with FFREE, which flushes this part of the memory.

library_of_functions library.fun

You can store and thereby save functions into an ASCII file called LIBRARY.FUN. This library can be updated at any time and you can retrieve function from it while running DATAC. The current library management is done through the FSAVE_command and the FRETRIEVE_command. If you modify a function (correction or updating) and whish to save it under the same name the new function will replace the old one and will be stored at exactly the same position in the library. Every function in the library is allocated 80 characters (fixed length) and is delimited by two markers: a ":" at the end of the function per se and a"!" which indicates the end of the paddable spaces required to reach 80 characters. This was done to allow you (and us) to write comments describing the functions using the editor EDIT. Without these comments the library would be nearly useless. So add comments and blank lines wherever you want but try to avoid editing in the function line otherwise the program may not recognize the function anymore. If this happened to you, make sure to place a ":" at the end of the function and to align the "!" on that of another function. Modification of the function itself is best achieved in COMPUTE where you can directly test it before you save it. CHECK YOUR FUNCTIONS CAREFULLY. IDEALLY YOU SHOULD HAVE A TEST AND THE RESULT OF THE TEST SHOULD BE WRITTEN IN THE LIBRARY COMMENTS. Experience shows that it is easy to make mistakes even (or may be especially) when you copy a function from a book!

recalcon_flag and recalcoff_flag
It will happen that strips are interdependent. For example, suppose you have x=100*time and y=exp(-x/30). In this case if you changed something in x you will have to recalculate y. This procedure will be done AUTOMATICALLY if the recalcon flag is on. This is the default situation in the program and you can see it as the status of the flag is given on screen in compute. IMPORTANT NOTE There are situations where you dont want the flag to be on. A typical example is y1=y1+y. Here with the flag on there will be an infinite loop. Since this will undoubtedly happen to you, remember that you can interrupt the loop with CTRL-C. When you can anticipate such a problem, simply turn the flag off by typing recalcoff.

STRIP_CORRECTION, CORRECTING_A_STRIP, EQUATION_CORRECTION

COMPUTE_command CURSOR_position

To correct the equation of a strip you must first type compute (cr) wait for the * and then you enter the name of the strip you want to correct. Text edition for correcting the equation of a strip works essentially like Wordstar. Ctrl-s, moves the cursor 1 character left, ctrl-d, 1 character right, ctrl-qs and ctrl-qd move the cursor to the beginning and end of line, ctrl-g deletes one character, ctrl-y deletes the entire line. Insertion can be done everywhere at the cursor position. On IBM-PC and compatibles, running MS-DOS version 3.0 or above, the cursor can be moved with the arrows of the numerical keypad.

NOTE: you cannot correct directly the data in a strip in compute. However, if the data are first transferred in an ASCII file you can obviously act in the data file using the editor present in DATAC (see EDIT). If you just transferred data in an ASCII file and want to edit it, do not forget to close (type CLOSE) the file before accessing it with the editor.

LIST_THE_DATA_IN_A_STRIP, STRIP_LISTING LIST_command

The command LIST strip-name(cr) allows you to display the values contained in a strip of specified name. This applies to datastrips as well as to strips created by equations. 25 values will be displayed on a screen; to see the 25 next values type a (cr). To interrupt the display and return to the main module, press q.
Example: to list strip y4 type list y4 (cr)

STRIP_PARAMETERS STRIP_command

The command strip (cr) permits to set the various parameters in a strip created by the command compute. When you activate strip, the program will ask you the name of the strip you wish to work on; enter the name and (cr); the program will then display your strip and the entire set of parameters that you can modify (see Fig. 3). Stripparam permits to select the color of a strip to be displayed on a graph. It also permits to use markers of different types, sizes at selected intervals. When axes different from the default time current and voltage axes are used parameters for these axes can be set, and adequate names can be given to the abscissa or to the ordinate. NOTE: a two dimensional graph is always made up of at least two strips. The dimensions on the abscissa and on the ordinate are selected by the parameters set in the corresponding strips. The abscissa and ordinate strips need not to be labelled x or y respectively. It would be perfectly legitimate to display a strip x as a function of the strip y. It is simply more easy to keep track of the various strips in memory if a meaningful label is given. GRAPH_and_ORDINATE_STRIP: the line type, color and markers of a graph are determined by the parameters of the ordinate_strip.

FIGURE 3
DISPLAY RESULTING FROM ACTIVATING THE COMMAND STRIP

```
Strip Command:


Strip: x    Points:     2048     [vers 3.0]
Equation :
Name :
Orientation:  0   Label origin  0
| xpos   :          0 | ypos   :          0 | axecol :      green |
| min    :          0 | max    :          0 | step   :          0 |
| marker :          0 | skip   :          0 | size   :          0 |
| ctype  :            | aflag  :          0 | log    :          0 |
| linet  :          0 | bar    :          0 | nonumb :          0 |
| noaxe  :          0 | fig    :          1 | intax  :          0 |
| format :                                |                     |
| color  :      green |                     |                     |
........... User figure definition .........
| scale  :          1 | user   :          2 | pos    :          0 |
| umin   :          0 | umax   :          0 | ustep  :          0 |
| fxpos  :          4 | fypos  :          4 |
| tick   :      2.200 | dir:   :
| bwidth :      0.000 | depend :          0 |                     |
| owner:                    none |
No template strip
```

IMPORTANT All the commands in strip can be nested on a line. Therefore, you can change several parameters at once. For example the following line of command after having typed strip y (cr) is perfectly legitimate:
min -500 max 100 step 200 color red intax 1 era tr x y (cr)
It will cause the modification of min, max, step, color and intax then the screen will be erased and y will be plotted as a function of x.
  You must also understand the following: you have the possibility to access DATAC commands_from_anywhere in the program. But this applies only to the commands listed in figure 1. For example, if you are creating a strip y3 with compute you cannot type color red and expect DATAC to know what to do. Indeed the word color can be applied to a text as well as to a strip. Therefore, if you want the strip y3 to be traced in red you must first give the command strip y3 (cr) and then indicate the color.

We shall now describe the various strip parameters

Strip_color   COLOR_parameter

To set the color to red for example type color red. Available colors are green, red, blue, yellow, magenta, cyan, white and black. Note that you need only to set the color of the strip determining the ordinate.

Curve_type   CTYPE_parameter

The type of a strip is an important characteristic. Since strips can be created at will from equations and from data, it is crucial to put some barrier to prevent inadequate operations between strips. Strips recovered from data are either of the type t(time) or i(current) or v(voltage). Theoretical strips are of undefined_curve_type. To allow operation between a strip of data of the i-type and a theoretical strip, simply set the curve type to 0. This is done by typing ctype 0 for the data strip. Remember that if for some reason you re-load a data in this same strip the curve type i will be set again.

Strip_marker_types MARKER_parameter

The marker types and their code (according to DISSPLA) are: square(0), octogone(1), triangle(2), +(3), x(4), diamond(5). To set the type enter marker (if you are in the strip module) followed by the appropriate number. Example, for triangle type marker 2.

Strip_marker_interval SKIP_parameter

This parameter (skip) determines the number of points skipped between each marker (remember that a continuous line is made up of 512 points). If a MINUS SIGN precedes the value as for example in the command skip -10, markers will be displayed alone 10 points apart. In the absence of a minus sign markers will be superimposed on a line. The markers color is that defined for this particular equation.

Strip_marker_size   SIZE_parameter

Size determines the size of the markers. On screen adequate sizes are usually between 5-6.

Strip_axes
MIN_parameter MAX_parameter STEP_parameter AXECOL_parameter AFLAG_parameter XPOS_parameter YPOS_parameter NAME_parameter ORIENT_parameter

You can set the dimensions of the axes, their color, position on the graph and give a label. REMEMBER, one strip is either of the x-type, the y-type or the z-type as determined by the user in compute. So in a two-D graph, one strip can only define either the abscissa or the ordinate. The label of the axes is given by typing name followed by a text of at most 8 characters. The position of the text on the graph is set by xpos and ypos followed by adequate values in cm. Positions are with respect to the margin of the A4 format on the plotter (see Ch. V). The orientation of the text on the screen can be either horizontal (orient 0) or vertical (orient 1). Note that this is not true for the HP plotter, where more degrees of freedom are available (see Ch. V). The color of the axes is set by axecol followed by a color name (see COLOR_CODE for the available colors). The minimum and maximum values of the axes is set by min and max followed by a value. Tickmarks_interval is set by step (by default it is 3 intervals; to set the default, set step to 0). An AUTOMATIC_SCALING is possible by setting aflag 1. In this case the program determines the min and max automatically. This works only for strips that are not of the i or v type. For the latter, the default parameters of the axes are those existing in ordinary display (parameter). When two curevs are displayed with auto_scaling, an autoscale message will be displayed to draw your attention on the fact that the scale may actually not be the same for both curves.
NOTE that you can read_min_and_max_values automatically set by DATAC by simply displaying the strip parameter after a display attempt with the AFLAG on. If a strip has a constant value, the min and max will obviously be the same. An attempt to display such a strip with the AFLAG on will cause an error. If you want to display a strip with a constant value, you must set the min and max values yourselves (the AFLAG must be off i.e. 0).

Bars_for_histograms   BAR_parameter

When plotting a histogram (see Ch. IV for the procedure creating_histograms), you can chose to have the ordinate plotted as bars by setting BAR 1 (reset with bar 0). The thickness of the bar is determined by the bin's width when creating the histogram.

Log_plots   LOG_parameter

To have the axes plotted logarithmically instead of linearly set LOG 1 (reset with log 0). The correct number of tick marks will be set automatically depending upon the boundary you did set. NOTE: in a log plot you cannot have a minimum value at 0. Therefore, if the minimum is left to 0, the program will either signal an error or automatically adjust the lower value three decades lower than the upper boundary. If, in the case where you use a log the program fails to display the curve it means that something is wrong with a strip (for example negative values!). Check your strip again.

Line_type   LINET_parameter (for PLOTTER)

You can chose the type_of_line for a given curve by setting LINET to the values 0 to 3. The LINE_CODE is:
0: continuous_line _____________

1: line_dot   ___ . ___ . ___

2: dashed_line   _ _ _ _ _ _ _ _

3: line_dash   _____ _ _ _____ _ _ _____

Lettering_control (Decimal_suppression) INTAX_parameter

The INTAX_parameter allows you to suppress decimal values in the display of numbers on the axes when INTAX is set to 1.

Lettering_suppression NONUM_parameter

Some journals require graphs_without_lettering. To trace the axes and the tickmarks without the corresponding numbers, set NONUM to 1 (reset the numbers by setting nonum to 0).

SPECIAL_AXES, USERs_DEFINED_AXES

FIGURE_parameter

You may want to position you axes yourself especially to construct high_quality_graphics (see Ch. VI and the DEMO manual). The access to users defined axes depends on the value of the parameter figure in stripparameter which can take the values 0 or 1. When figure is 0, default axes are used.
When figure is set to 1, you have access to customized axes and the display of strip changes: a few more words appear, most of them beginning with a u (see Fig. 3, under the heading user figure definition). We shall now explain the meaning of these command words and you will see an example of application in the demonstrations (see DEMO manual).

USER_parameter

This parameter can take the values 0, 1 or 2. If your strip is an ordinate strip and you plan to display only one ordinate axes, set user to 1. If, however there should be more than 1 ordinate axes (see DEMO manual for an example of what we mean by that), make sure you set user to 2 otherwise a curve will be traced but without the corresponding ordinate axes. When user is set to 0 no axes will be displayed for this strip.

UMIN_parameter, UMAX_parameter, USTEP_paremeter

Umin defines the actual value of the beginning of the axes, umax the value of the end of the axes and ustep the increment (number of tickmarks on the axes). For example, if umin is 0, umax 100 and ustep 25, the axes will begin at 0, end at 100 and tickmarks will be displayed at 0,25,50,75 and 100.
Note: as you will see in the demonstrations (DEMOFIG2 in the demo manual is a good example), you can displace the graph on the screen (and on the plotter) by playing with the min and max and the umin and umax. You can visualize that in the following way; the min and max set a fix but invisible frame for a given axes, and by choosing umax and umin you decide where on this frame your axes is actually going to be displayed. That way you can shift an ordinate axes up or down and an abscissa left or right. THE DATA CURVE WILL ALWAYS BE POSITIONNED CORRECTLY, you dont have to worry about that!

POS_parameter

POS defines the horizontal position of an ordinate axes or the vertical position of an abscissa axes (sorry for the headache!). The value is given in cm with respect to a plotter frame (see frame in Fig. 5). pos allows you to position an ordinate either at the left or at the right of the curves you want to display. Note that depending on which side you put the axes the tick mark will not be oriented correctly. If you put the y axes on the left, TICK must be set to 0. If you put it to the right set TICK to 1.

SCALE_parameter

Scale allows you to simply scale down your display. If you set scale to .5, the display of the corresponding axes will occupy 50 % of the space occupied by the normal scale (note that scale=0 is the default. i.e. 100% or 1).

SETTING_DEFAULT_PARAMETERS

TEMPLATE_STRIP, DEFAULT_STRIP

When many of the parameters of a text or data strip are alike from one strip to the next, the template facility allows you to set all these parameters with a single command, default. The way to proceed is first to set the parameters of one of your strips (data or text) with either strip or tparam. Then, while you are in strip (or in tparam) type template followed by the name or the number (for a text) of the corresponding strip. When you call another strip in strip or tparam (for text) the label template x will be displayed indicating which strip is the template. To set the parameters of another strip with the template, simply type default (cr) in strip or tparam.
Note that you can change the template strip at any time. The strip you chose, however, must obviously be an existing strip. If not an error message will be displayed (no template selected).
If you flush the data strips or activate the eratext command you will have to recreate a template for the corresponding module (provided you need a template, of course!).

IMPORTANT -you cannot select the template for a text when you are in strip
-THERE IS ONLY ONE TEMPLATE FOR DATA STRIPS AND ONE FOR TEXT STRIPS

shx, shy: shift the x and y position of the strips (cm units). (Prefered is to use the panel structure to build sets of strips and manipulate the panels).

dupe: (short for duplicate): copies the template strip information EXCEPT for the color and marker information; also sets the user axis flag to 1. Useful when multiple plots are created on the same axes but with different characteristics.

setfigure: if the strip ctype is v,V,i,I, or t, then the parameters from the param block are copied into the strip and the ctype set to 0.

Useful for turning strips of data pulled from a file into strips useful for figures.

rightaxe: sets the strip up so that umin = min, umax = max, and ustep = step, with the axes plotted on the right hand side.

leftaxe: sets the strip up as in rightaxe, but for left hand side axes.

TRACING_GRAPHS_FROM_STRIPS TRACE_command

With the command TRACE (cr) you can make a graphic_from_strips previously created with compute. Let us assume you created two strips, one for the abscissa (x3) and one for the ordinate (y4). The command tr x3 y4 will plot y4 as a function of x3. Note that unless you have current or voltage curves recovered from data, it is your responsability to set the parameters of the axes correctly (see strip axes).

retrace_command

It often happens that after tracing a series of curves one would like to change a setting (for example the time displayed) or recalculate a strip after modifying a parameter. In order to trace the graph correctly you would then have to erase and enter a trace command. This can be tedious when tracing_several_curves. In this case use the command retrace. It wil cause all the curve that were previously on screen to be erased and retraced appropriately. Note that this command can be particularly useful when combined with the recalcon_flag and also when adjusting parametesrs for a figure.
smoothing_strips
It is possible to smooth a strip directly by applying the operator smooth to a particular strip. The result will be to apply a mooving_average on 3 points to the strip. The operation can be repeated on the same strip several time. An example is: y1 = smooth(y)

CROSSHAIR_WITH_STRIPS   CROSS_command

Having activated the crosshair (CROSS (cr)), move the crosshair to the desired place with the joystick (or with the arrowkeys on the PC-AT) (for slow motion, press simultaneously on the shift key) and finally type any key. The program will display the x and y position of the crosshair in the upper right corner of the screen. The crosshair will also work adequately in the LOG_MODE. Press q to quit the crosshair. NOTE: you can save the reading you make with the crosshair in two strips named xcur and ycur. To do that, each time you want to save the coordinate of a point, type s. The two strips, xcur and ycur, can be used later. Be aware of the fact that each time you activate the crosshair command the strips xcur and ycur are cleared. To save the data contained in these strips transfer them in compute (xd=xcur, and yd=ycur, for example)
An application of this facility is described in Ch. IV, under the heading REDUCTION OF DATA (see index for the page; see also DEMO manual).

TEXT_CREATION ON A GRAPH   TEXT_command

The command TEXT (cr) allows you to display a text on a graphic where you want and in the color of your choice. If you type text, the crosshair will be activated. Position the crosshair where you want the text to begin or where you want the text_to_be_centered (see Label Orientation parameter and Fig. 6) Now enter the color of your text (code: green (1), red(2), blue(3), yellow(4), magenta(5), cyan(6), white(7), black(8) and type (cr). The cursor will come back home. You can now type your text (maximal number of characters allowed: 80). After (cr) the text will be displayed. Each time you activate text, a new text strip is created, which automatically receives a label (a number). Text strips are special structures containing the text and all the information concerning this text (color, character size, orientation slant, type; presence of arrows; centering etc). Text strips are stored in the heap memory; therefore, the only limitation on the number of text strips that can be stored is the size of the available memory. All strips will remain in existence and can be recalled and introduced selectively in other graphics. In order to clear the text strips, use the command eratext (see info on this command for further details). To set the parameters of a text strip use the TPARAM_command. You can set these parameters by default if there exists a template_text_strip (see DEFAULT STRIP above).

PARAMETERS_OF_A_TEXT   TPARAM_command

The command TPARAM (cr) allows you to set or update the parameters for a given text strip created with text. When you type tparam, the program will prompt you to enter the text number. Parameters will then be displayed (see Fig. 4). To set or change a parameter in a text strip, simply enter the appropriate parameter name followed by the new value. An all purpose character size is 1.9. The slant determines the inclination of the character with respect to the vertical; suitable values are in general .1 to .25. The angle determines the orientation_of_the_text on the plotter (on the terminal you will only have horizontal and vertical display). A 90o angle will plot the text vertically and upward. 270o will plot the text vertically and downward. A nice character_type is 10. You can choose the color of your text (green, red, blue, yellow, magenta, cyan, white and black).
NOTE (i) you can correct the text or add something to it using edit in tparam (see below). (ii) you can use ROOT_in_TPARAM If you do not remember the number of a text strip, you can activate root. When you are in tparam, root will display all the existing text strips.

The text_position is given in cm with respect to the boundaries set by the plotter. In this way you can modify the position that was originally set when creating the text with the crosshair. To center_a_text near a particular location or to justify_a_text with respect to the crosshair location, use the Label_Orientation_instruction (see Fig. 6).

TO MOVE_FROM_A_TEXT_TO_ANOTHER SIMPLY TYPE tparam (cr) and then x (cr) where x is the number of the text in which you want to set the parameters.

error bars and on bar-graphs

Error bars
When error bars need to be traced on a graph additional information must be given because the graph becomes more than just a tracing of strip y as a function of a strip x. This problem was solved by associating a strip containing the calculated standard deviation to a corresponding strip of data. We call the associated strip a dependent strip. Let's give an example.
Say that strip y contains a series of mean values that are a function of a series of variables contained in strip x. Say also that yd contains the standard deviations corresponding to each data in y. Proceed as follows: (1) set the parameters in strip x (as usual with stripparameter) (2) set the parameters in strip y
(3) in strip y, set depend to 1 and write sdep yd
to indicate that there exists a dependent strip and to give the name of this dependent strip (4) type depfun sem to select the type of function of the dependent strip. Here sem indicates that the standard deviation is the function. This is an indication for the program as to what type of graphics will be involved. NOTE that it does not mean that the standard deviation will be calculated automatically. Now if you write tr x y you will see that error bars are added to each point on the graph. Note that the color used for the error bar is the same as that of the main strip (here y), unless you select another color in the dependent strip (yd). You could have error bars in x-axis as well either selectively or simultaneously by proceeding as above with a strip xd. Should you decide to display y on a logarithmic scale the dimension of the bar in the dependent strip will automatically be set correctly. The "esthetical" (not the real length!) dimensions of the error bar can be adjusted so that it matches that of the symbol used in the y-strip. This is done by setting the symbol size in the dependent strip. Note that if you put error bars in both x and y you will have to set the size in both dependent strips. Finally, be aware of the fact that with most available terminals the on-screen resolution is worse in the vertical than in the horizontal axis and that it is considerably worse than on the plotter. Consequently, the fine tuning will have to be done on the basis of the plotter output.
Bar-graphs
Bar graphs can be represented by setting the flag bar to 1 in the corresponding strip (with the stripparameter command). The width of a bar can be adjusted with bwidth. If bwidth is -1 the bar will have a size corresponding to the size of the first increment in strip x. This is applicable when all increments in x are of the same size! The bars will be centered on the value in x. This is sometimes bothersome as in amplitude histograms, for example. The problem can be solved by shifting the x strip by half a width. Positive values of bwidth set the bar width to a user-defined value.

FIGURE 4
DISPLAY RESULTING FROM ACTIVATING THE TPARAM COMMAND

```
Text Strip Command:


------ Text Parameter -------
Label # = 1
xpos (cm)   7.33129ypos (cm)  11.85820
size = 4.000 slant = 0.000 angle = 0.000
Text font     1  Label Origin = 0 (LO)
| color  :      green |                      |                      |
Text: This is a text line

------------ text pointer definition ------------
marker   0  line    0  pcolor   0  psize 0.000
x1 0.000 x2 0.000 y1 0.000 y2 0.000
arrow 0

Current Numeric format: %8.3f

No template strip selected
```

MODIFYING_A_TEXT_IN_TPARAM EDIT_in_tparam

You can correct a text in a text strip or add something to it (remember the max number of characters is 80). For that purpose activate tparam, enter the number corresponding to the text strip you want to modify and type edit. This will cause the text to be displayed at the bottom of the screen with the cursor pointing to the last character in the strip. You can now edit using the same types of command as in Wordstar (ctrl-s,ctrl- d, to move the cursor left or right, ctrl-g to delete a character, etc). Terminate the edition by typing a (cr). On IBM-PC and compatible running MS-DOS 3.0 and above the arrows on the keypad can be used to move the cursor.

Text Strip manipulation

shx, shy: shift the position of a text strip.
move: invoke the mouse/keyboard to move a text strip to a new location. Moves both text and any associated pointer/marker.

ARROWS_OR_MARKERS_(GRAPH)   TPARAM_command

It is possible to emphasize a particular curve or zone of a graph by using an arrow or a symbol. An arrow is a vector, which is defined by two sets of coordinates (x1,y1,x2,y2) that are created by positioning the crosshair at two different locations on the graph. When the two sites are selected to be the same, the program infers that the user wants to place a symbol (and not an arrow!) at that location. The color, dimension of the arrowhead, the type of symbol used as marker can be selected.

Drawing_an_arrow or drawing_a_line (vector)

The goal is usually to connect an area of a graph to a text with an arrow. The first step, then, is to create a text strip (see above). Note that if you wish to have exclusively an arrow drawn without any text, you just have to write a (cr) in your text. You need a text strip, however, because an arrow or line must be related to a text strip. After having created a text strip, activate tparam and enter the appropriate text strip number. Now type read; the screen will be returned into the graph mode and the crosshair will be turned on automatically. Position the crosshair where you want the origin of the arrow (x1,y1) and type any character. Move the crosshair to that particular location where you want the arrowhead to be (x2,y2) and type a second character. The screen will then return automatically to the tparam module. You can now select the color and the various parameters of your vector. The arrow is drawn according to the value of a four digit code number (we follow the DISSPLA nomenclature). The code is: arrow wxyz (cr)

where w,x,y, and z are numbers with the following characteristics and significations:
w: 0<w<7 defines the width of the arrowhead
x: 0<w<7 defines the length of the arrowhead

y: 0   means filled arrowhead
  1   means unfilled arrowhead
  2   means ------
  3   means ------

z: 0   means no arrow (simple line)
  1   arrow points to x2,y2
  2   draw a second arrow <--->

3 draw a second arrow >--->

Example of entry: arrow 2301 (cr)

Note that these vector parameters can be modified in tparam at any time. Note also that some of the features of the plotter, such as automatic centering are not reproduced on the screen. When you use these features some fiddling and diddling may be required to find the right combination on the plotter output. To display the marker you just need to print the updated strip.

Placing_a_symbol(graph)

After having entered a text (or a (cr)) activate tparam and enter the appropriate text strip number. Type read, the graph mode will be activated with the crosshair on. Position the crosshair where you want the symbol to be on your graph and type any key TWICE. The program will return to the tparam module where you can select the marker type, size, color (for the color code, see this term in the index).

DYSPLAY_A_TEXT ALREADY IN MEMORY PTEXT_command

The command PTEXT displays one or more selected texts already stored in memory on the graphic terminal or on the plotter (if it is set, see PLOTTER section). ptext works like display. Typical command to display the texts 1 to 5 and 7 and 8: ptext 1;5,7,8 (cr)

Up to 50 text strips can be displayed with a single ptext command.

PRINTING_SERIES_OF_TEXTS
READTEX_command
When preparing posters or slides it is often convenient to prepare a text and have it written by the plotter. This could be done line by line with the command TEXT, but is quite cumbersome. The goal of the READTEXT command is to allow you to set conveniently the various parameters of your text and also the general orientation (analogous to landscape and portrait for the Laserjet).
You can edit your text from within DATAC with edit. Write the text, each line containing the text you want in your final version. Each time a particular line should have a special setting (color, size of characters, etc) a dot_command_instruction must precede that line. A given setting will apply for the rest of the text unless a new correpsonding command is given.
The dot commands are
.xpos sets the position of the text in the x-axes
.ypos sets the position of the text in the y-axes
.size sets the character size (2.5 / 3 are ok for retro) .slant sets the slant of the text
.color sets the color (1=green, etc, see COLOR CODE)
.rot sets the orientation of the text.
EX:.rot=1 set the portrait configuration i.e. A4 in vertical .rot=0 (default) set the landscape configuration i.e. A4 horizontal .LO (see Fig. 6) default is 1. For centering, use .lo 15. This command is important with respect to ypos in the portrait configuration and with xpos in the landscape config. In A4 vertical .ypos 10.5 and .lo 15 will center the text. .type sets the character type. Nice characters are type 10 but they unfortunately are of the proportional type which sometimes may complicate the preparation of the text to be plotted. Indeed on the screen you will not see the final disposition of the text so that misalignement may occur. You can obviously correct afterward. Another possibility is to use ordinary character type 1) when alignement is crucial. Note that the default is type 10. An example being worth approximatively 1000 words, here is one edited as it would be from within DATAC

```
.rot 1
.size 3.5
.color 2
.ypos 10.5
.lo 15
.xpos 5
POTASSIUM CURRENTS


.size 3
.color 3
.slant .25
.ypos 2.5
.lo 1
1. VOLTAGE - GATED

2. MODULATED INTRACELLULARLY

3. MODULATED EXTRACELLULARLY
```

If you activated readtex and give the name of the file where you wrote this example, each line (including the blank lines) will be stored in a text-strip with the appropriate parameters. To send the text to the plotter use ptext and enter the number of strips (be generous, remember the blank lines count too, and an excess cant hurt). The text will appear in the A4 vertical (portrait) configuration, with potassium currents in red and centered. The rest of the text will be in blue at 2.5 cm from the left margin and with a smaller letter size together with a nice little slant.
If you are unhappy with the output, you can either correct a text strip with tparam (see this term) or change your file with the editor and use readtex again. Note that READTEX does not destroy existing text-strips and simply appends new ones. If you dont use the

old strips operate an eratext command to flush all the text strips before you initiate readtex again.

BLANKING_PARTIALLY_A_DATA BLANK_command, UNBLANK_command

It is sometimes necessary to erase_selectively a part of a curve for cosmethic_purposes. This selective_blanking_procedure, which works exclusively on strips, is activated as follows; type BLANK (cr). The program will ask you to trace the data you want to correct (one data at a time will be corrected). To trace, you need an abscissa and an ordinate strip, but the blanking takes place exlusively on the ordinate strip. Move the crosshair to the beginning of the section you want to blank and type a letter. Move it to the end of the section to blank and type a letter. The program will retrace the data with the appropriate section blanked. Note that you can also blank_backward. You can blank several sections on a same curve by repeating the procedure. The blanking session is terminated when you depress the key q. To restore the original curve without blank simply type UNBLANK (cr) and enter the name of the strip to unblank.

STRIP_LISTING, LISTING_THE_EQUATIONS   ROOT_command

The command ROOT (cr) will display on the screen all the equations you wrote with compute. In tparam, root displays all the text strips. NOTE: if the display takes more than one screen, use ctrl-s to interupt the display and visualize the first strips; use ctrl-s again to restart the display.

MEMORY_AVAILABLE   MEMORY_command

Strips are stored in the computer_memory called heap. Depending upon the memory initially installed in the computer and on the number of strips you (and the program) have created, you may run_out_of_memory causing the error message no_more_allocation_space to be displayed. The program is protected so that nothing will go wrong. Note that you can run out of memory with regular strips, with text strips and with function strips. The way to proceed is to save the important strips or the functions (see Ch. VI) and then to flush the appropriate memory section with either the flush command for strips
the eratext command for text strips
the ffree command for functions
You can make a memory_check with the command MEMORY (cr). If the available_memory is larger than 65 kbytes the answer will be 65000. If it is smaller, the corresponding number will be displayed. NOTE that the STRIP_SIZE_IN_MEMORY is approximately 3K bytes.

CLEAR_THE_STRIPS, MEMORY_FLUSH FLUSH_command

To avoid running out of memory it is wise to flush_the_memory whenever you can (for example when you are through with an analysis). This is done with the command FLUSH (cr) and answering Y to the program prompt. If you have strips that you dont feel like loosing, save them (see Ch. VI) before the flushing procedure.

PANELS

Panels are a new structure in DATAC. They permit a group of up to 16 strip pairs and associated text strips to be treated together, for placement, scaling and plotting. Panels can be linked together to make more complex figures or to put more strip pairs into a group.

Panels are defined by a 32 character name; the program automatically prefixes a "P" to the name but the user only addresses each panel by the rest of the name. Automatically linked panels have an extension of "$n", so short base panel names are necessary to fit into the name limit. Panels have an x and y anchor point that is used to locate the crosshair when moves are made. There is also a scale value that, once all the strips are specified with a scale of 1.0 and have the correct relationship to each other, can be set to scale all the strips the same at once. The x and y scales may be changed simultaneously; or individually. Text scaling is done with a separate command. Strip pairs can be added or deleted from the panel with the add or del commands. Added strips are always put at the end of the list, but strips can be deleted from any point. The text strips associated with the panel are specified in a command line just as you would specify the ptext command for the strips you wanted. The overall character size is selectable.

Panel creation

Panels are created by typing the command "panel name" at the main DATAC prompt. The name should not have a "p" prefixed to it unless you explicitly wish it. The panel screen will then appear. The following commands perform operations on panels.

Housekeeping

quit   : return to main DATAC level.

help or ? : list the current valid command list.

panel name: select/create a new panel.
add xstrip ystrip: add strip pair to trace to the current list. If the current list is full, linked panels will be created as necessary to extend the list.
clear: clear entirety of current list.
delete n: delete selected item from list.
swap n m: swap order of a pair of elements in the list.
text textlist: add a ptext command of text strips to display with the panel.
proot: display panel root.
pchk: check to see if all strips for panel are in memory.
ptrace: trace just the panels (erases screen first).
x pos: set x anchor location (cm).
y pos: set y anchor location (cm).
scale f: set scale for data strips (scaling is absolute relative to default full screen).
depend [1,0]: enable/disable link flag to another panel.
link panelname: name of linked panel.
shift: shift the panel as a unit using the mouse/keyboard.
move: alias for shift.
rshift relx rely: relative location shift (in cm).
ashift x y: absolute location set for anchor (in cm) and panel.
charsize f: set text size for axeslabels: absolute.
relchar f: set text size relative (adjusts all).
gettrace: put contents of retrace list into a panel.
next : moves to next panel in a linked list.
fixscale : adjusts the scaling of all of the strips in the list (including linked panels) to the min and max of all of the x and y values, then sets the first strip pair in the base list for the axes (user 2 and setfig) and all of the other strips are effectively treated with the "dupe" command (see strips, above). Useful when you need to plot a number of somewhat disparate records on a single scale.

Using Panels

The panel is a convenient form to maintain an associated group of traces in a figure. First, you should get your data into strips, and set the min/max/umin/umax etc to your liking (or you may wait until you create the panel and use the fixscale command). It is best to work with a full screen display (i.e., don't scale the data down until later). Text strips can then be created to label the axes, point out features of the data, etc., and positioned at the normal scale. Remember that when the panel is rescaled that the text strips and labels will be reduced in size, so make the text strips large enough to be readable later. The axis number label size is controlled by the panel.

There are two ways to load the panel strip list. You may use the "add" command (as well as the delete, swap, etc.) to put the strips into the panel. Since the strips will be plotted in the order of their appearance in the panel strip list, set the user axes accordingly. The second way to load the list is to manually trace the data on the screen, then go to the appropriate panel and use the gettrace command. This command will load all of the strips (both data and text) into the panel lists. Subsequently, a ptrace command will trace the panel.

To change the panel size and move it around the screen, first make sure you have everything set the way you want it with the panel at full scale. Scaling and moving are not commutative, so you should first scale the data (for example, to put 4 plots on a page, use a scale of 0.5 for each panel). When the panel is replotted, it will appear in the lower left corner of the screen. Then you may move the panel with one of the shift commands to the appropriate location. Note that if you want to rescale the panel after moving it, I recommend that you first move the panel back to the origin (this depends on the scale; if the scale is 0.5, the origin is now 2,2), and then change the scale, then move it back to where you want it. Just changing the scale will result in the panel also moving (since the scale affects the anchor points and the assumed plot origin), and this can produce confusing results.

If you have more than 16 strip pairs that will be plotted in a panel, you can link the panels (add and gettrace do this automatically) with another that contains the rest of the list. You should do this before the final placement of the panel if the data are to be plotted on the same axes.

The final step in a figure might be to link a group of panels (a,b,c,d, for example) together so that only one ptrace command to the first panel is necessary to invoke the entire figure. However, if you then want to move components of the figure individually, you will have to break the links (use depend 0).

Panels provide a convenient mechanism for grouping data objects together and manipulating them on the screen; their use is recommended over creating macros that plot lots of individual strips. Panels are saved and restored with the strip files.

Calibration bars for axes: You may now change the axes to be a corner (two lines joined at right angles) with labels, similar to that used in many papers to indicate the calibration of a recording. This is a function in the panel module. First, create a panel with the strips you wish to plot. This will normally have the usual graphic axes. Before scaling and shifting this panel to its final destination, use the "calbar" command (in panel):

calbar xcalsize ycalsize xlocation (cm) ylocation (cm)

When you replot the data, the calibration bar should appear. If you wish to change its location, you will need to emit the entire command again. The strips in the panel are assigned the "c" curve type to indicate the use of a cal bar for axes display. Setting the curve type to "c" is insufficient to initiate a calibration bar, since its location requires calculation by the command in panel. Once a calibration bar has been established, the panel may be scaled and shifted, but the individual strips should not be directly modified, or the command may need to be reissued. I recommend using this command only when the strips are plotted on the scale that you wish for a final plot.

zciv: stands for "zero crossing iv axes". This command, located in the panel module, permits the graphic axes to be aligned along each others zero position, as is customary for the plotting of current-voltage relationships from voltage clamp or current clamp. The numbering of the axes at the zero position is suppressed. This can be done at any time to a panel; the strips in the panel are assigned the type "z" to indicate their unusual status. Again, this command should be executed once the strips are correctly plotted; modifications to the strips directly will not result in a correct axes location because zciv calculates the necessary positions on the fly. The strips may be scaled and shifted within the panel module however.

Ownership of strips and panels: Strips which are comprised of concatenated strips have an "owner", which consists of the base strip (the first one, which is now "owned" in the sense that it is not a dependent of any other strip). In the strip module, the command "owner" will change the current strip to the base strip of that concatenated group (i.e., yv of yv, yv&0, yv&1, yv&2, yv&3 if you are in any of the concatenated strips). "Previous" will move you back to the strip preceding the current one in the list.

The panel module has a similar function.

V
OPERATORS AND DATA ANALYSIS

As mentioned in the introduction of this manual, one of the strength of DATAC comes from the ability of the program to recognize and solve various combinations of operators and strips. We shall call operator an operation which returns its result in a strip (be it user defined or temporary).
The operators are

+,-,*,/

sin,cos,tan,asin,acos,atan

exp,log,ln,sqrt,pow (or ^)

cte, time

mean, variance, spec, fft

integr, rnd, reglin, interpol, smooth, diff, sort, iv???

trigger, ctime, otime, flatency, histo, pgen

Any of these operators can be used when writing an equation with COMPUTE (see Ch. III). Note that some of these operators are special and will be described in more details in this section:
- variables

- linear regression (REGLIN)

- a series of routines for I/V curve determinations (IVFIX, IVMIN, IVMAX, IVV)
- curve fitting (FIT)

- differential equations (DIFF)

- a routine for putting values of a pair of strips (x,y) in growing order of x (SORT)
- a routine for emulating single channel opening and closing, PGEN

The use of arithmetical operators is straightforward. The precedence is */, +-. If in doubt, use parenthesis.
The trigonometric_operators require parenthesis:
x1= sin(x+3)   y3= sin(x1) + 25* cos(x2)

The log, ln, sqrt and exp operators all need parenthesis

The pow_n_operator (or ^n) raises a number or a strip to the power n.
x3 is written x pow3 or x^3

NOTE: you can put a strip in the exponent. It may be useful sometimes (too long to explain here). The operator pow will accept parenthesis but does not require them. The space between the strip name and pow is required (remember that xpow3 would be a perfectly legitimate strip name!). If you use the "^", however, you do not need the space since this character is not allowed in a strip name.

MEAN_operator

This operator takes the mean of several data. To create the strip, proceed as described in compute (see STRIPS section).
Examples: (i) take the mean of current records 1 to 8 and 13 to 15. Write the equation (labelled arbitrarily y3 here):
y3=mean(i1;8,10;15).

(ii) take the mean of strips x1,x2,x3,x4,x5:

y4=mean(x1,x2,x3,x4,x5)

The results are in the strips that we named y3 and y4. For the display you will use tracing (see this term in Ch. III).

VARIANCE_operator

The variance_operator will calculate the variance of a series of strips and put the result in a strip. Suppose we have several recordings of the current as a function of time (i1,i2,...i10) during steps in voltage clamp to the same voltage. The variance of the current at this voltage is: y5 = variance(i1;10)
the result is in y5 and can be plotted as a function of time with one of the time (t1, for example) as abscissa. Note that it would be perfectly legitimate to write

```
   y7 = variance(i1;10)/mean(i1;10)
```

LEVEL_DETECTOR TRIGGER_operator

This operator works as a level detector and can convert data (on a 512 points basis) or strips into zeros or one's depending upon the threshold level. This operator was originally developped for single channel analysis but can be used in other applications. Example:
y=trigger(x,4)

will generate a strip y whose values are 0 when the values of x are below or equal to 4, and 1 when the value of x exceeds 4.
NOTE: - the operator works on one record at a time. To apply trigger to several records use a macro command with a LOOP (see section on macros). - the operator can be used several time in a given equation the following way:
 yd=trigger(y,4) + trigger(y,7)

where 4 and 7 are two different thresholds

-trigger can be used to generate a rectangular_step_strip. As such it can be used indirectly to perform a pseudo (or real) logical_operation or act as a mask. Example: suppose you have a strip y over the interval 0 and 100. You would like this strip to take the value 0 between 0 and 20, to keep its original value between 21 and 80 and to be 0 between 81 and 100. All you need is to multiply the strip y by a rectangular step strip which will take the value of 1 between 21 and 80. This can be done as follows, using trigger:

```
x=100*time
yt=trigger(x,20)*(1-trigger(x,80))
yc=y*yt
```

OPEN_TIME AND CLOSE_TIME MEASUREMENTS

OTIME_operator, CTIME_operator

These two operators calculate respectively the open_time_histogram and the close_time_histogram of strips that have been previously processed with the trigger operator.
 Example: yo= otime(y1,y2,y3,y4,y5,y6,y7)

The open and close time histograms can be plotted as a function of the time base of a record. The basic sequence of operations would be as follows:
1) set tbeg and tend in param (time window)
2) x=tm where m is one of the records
3) ym=trigger(im,level) where m is a record number
4) yo=otime(y1,....ym)
5) yc=ctime(y1,....ym)
6) tr x yo
7) tr x yc

NOTE that the procedure can be considerably simplified by using a macro command.
NOTE: The function otime() and ctime() were extended for the direct use on data stored on disk. This gives a better resolution (up to 2048 points instead of 512 in a strip) and a faster execution of macro loops. The typival call is: y = otime (sn,trigger,min,max,step) where s is the data type (i,v) n is the number, which can be replaced by % in a loop. Trigger is the trigger level determined by the user and where the channel is assumed to be in the open state. The minimum and maximum value of the histogram are given as min and max, and the bin value of the histogram is given as the step. Embedded blank are automatically discarded and a temporary strip xtemp is automatically created which can be used to trace the histogram. IMPORTANT: if you want to create a macro loop to establish an open time histogram do it as follows: ytot= otime(i1,9999,min,max,step)

```
LOOP 1;50
y=otime(i%,trigger,min,max,step)
ytot=ytot+y
END
```

The first line is very important; it initializes the strip used for the summation (ytot) with the correct number of points in it. If you were to ommit it ytot in the loop would automatically be created like a normal strip with 512 points. Since this would not match the number of points in xtemp, you would have a slight problem.

FIRST_LATENCY_HISTOGRAM   FLATENCY_operator

This operator calculates the time between the beginning of a strip and the first transition. It operates on a series of strip and calculates an histogram. Example: yf=flatency(y1,y2,y......,ym)

SINGLE_CHANNEL_SIMULATION PGEN_operator

This operator was designed for the simulation of single channel behaviour with a Monte_Carlo method. Each time pgen is activated it will generate a strip equivalent to an oscilloscope sweep during single channel recording. To get the equivalent of several sweeps you will have to activate pgen repeatedly. The simulation is done on the basis of a probability matrix which expresses all probabilities of transitions between the various channel states defined by the user and a random number generator. Up to nine states can be defined for a given channel type. The probability of transition from state i to state j is defined as pij = kij * dt

where kij is the rate constant for that particular transition and dt is a small time increment.
The time increment, dt, is automatically set by the program as 0.1 * (1/fastest rate constant). One can visualize dt as the time between two sampling during real data acquisition. Note that you can set the value dt yourself by saying Y at the appropriate time. YOUR dt, however, must be smaller than that set by the program.

The various rate constants must be edited in a file (use EDIT followed by a filename of your choice while running DATAC) which will subsequently be read by the program. There is a required format in which rate constant from one state to the other must be written as:
1,2=8.6
2,1=33.69
2,3=185.52
3,2=47.39

where 1,2 would be the rate constant for the transition from state 1 to state 2. Note that the order in which rate constants are given in the file does not matter. However, spaces and other signs must be respected since they are used as delimiters.
On the first run through pgen the program will read this file, determine dt (here 1/185.52, i.e .000539) and calculate a probability matrix on the basis of the rate constants you gave. For a linear 3-state channel the matrix may look like that: p11 = 1-(1,2)*dt
p12 = (1,2)*dt
p13 = 0
p21 = (2,1)*dt
p22 = 1-(2,1)*dt-(2,3)*dt
p23 = (2,3)*dt
p31 = 0
p32 = k3,2*dt
p33 = 1-k3,2*dt

This matrix can be visualized after calculation. For an explanation of the display see the manual on DEMO which will give a practical example. Pgen can use up to three different matrices in order to simulate the behaviour of voltage-dependent channels during steps to various voltages. It is obviously the responsability of the user to calculate sensible rate constants on the basis of some theoretical model (with energy barriers and wells for example). Pgen knows when it should switch to another matrix because an argument is given to it as a strip in parenthesis. Example: y = pgen(x)

The strip x contains values between 0 and 3. When the value changes, a new matrix is selected. An x-strip with abruptly changing values can be created with a trigger_operator. Here are two examples: 1) x1=time

x=trigger(x1,0.2) * (1-trigger(x1,0.8))

here the x strip is a step function which is equal to 1 between 0.2 and 0.8 and equal to 0 otherwise. This pattern can be used to select two matrices.
2) x1=time

x=trigger(x1,0.2)*(1-trigger(x1,0.8))+2*trigger(x1,0.81)

here the strip x is 0 between 0 and 0.2, it is 1 between 0.2 and 0.8 and it is 2 between 0.81 and 1. This pattern can be used to select 3 matrices.

how to give pgen() a new_set_of_matrices

Once a set of matrices was given to pgen(), it happily calculates probabilities of opening and closing as long as you ask it to do it. Should you wish to use another set probability matrix, you will have to reset pgen(). This is done as follows. As mentionned, data are read by pgen(x) when the value in the strip argument changes with respect to the preceding value. Therefore in order to read a new matrix it is necessary to modify the content of x. From within compute, proceed this way: x= cte-1 (this loads the strip x with -1's)

then
y=pgen(x)
on request simply type a (cr)

Since x in pgen() should only take values between 0 and 3 it will not be able to do anything with it. This is OK and now you can force pgen to read a new strip that will allow it to access new matrices.
x1=cte1 (load x with 1) (on anything appropriate)

then
y=pgen(x)

enter the name of the file containing the matrix when asked to do so.
The result of one run through pgen(x) will be a set of 512 values contained in y. This is equivalent to a single sweep of data.
In general, data simulation requires the generation of many sweeps to obtain open and close time histograms or summations (equivalent to macroscopic currents). To run repeatedly through pgen() it is convenient to use a macro command. Here is an example of a macro command called OTIME.MAC, that generates an open time histogram on 50 sweeps:

```
recalcoff
compute
LOOP 1;50
y=pgen(x)
y1=trigger(y,1.2)
yo=yo+otime(y1)
END
```

You can have 4 repetitions of this macro by evoking
mac otime 1;4
(in this case the overall result will correspond to 200 sweeps). Note that we use the recalcoff_instruction here because of the yo=yo+... instruction, which would otherwise induce an infinite loop.

AMPLITUDE_HISTOGRAM   HISTO_operator

In some cases you may want to display your data as amplitude_histograms. This can be done in compute. Assume that you have a ordinate strip y; you can obtain an amplitude_distribution of this strip by writing the equation: y1 = histo (y, min, max ,step)

where y is your strip, min is the lower and max the upper value of the amplitude and step is the width of one bin in the histogram. The program will create an abscissa strip xtemp, which you can use to display the histogram. Remember that you must first set the parameters of the strip xtemp and y1 and in particular you must set the parameter BAR in y1 to 1 (see Ch. III) in order to have a bar display.

FAST_FOURRIER_TRANSFORM   FFT_operator

The operator fft will calculate the fourrier transform of a strip y=fft(x,z) where x represent the real part and z the imaginary part of the data. If there is no imaginary strip (i.e. if you write y=fft(x)), DATAC will create a strip xtempi which is initially set at 0 an contains eventually the imaginary result of the FFT. The real part of the transform will be in the strip you chose (yf in the example below)
Example: x=6.28*time
 y=sin(20*x)
 yf=fft(y)

The data can be represented from -f to +f where f is equivalent to a frequency. The "frequencies" corresponding to the real part are in xtemp those corresponding to the imaginary part in xtempi. IMPORTANT: To give your abscissa the actual dimension of a frequency and to scale your frequencies according to the original data, you must create an appropriate strip (here for the real part): xr = 512 * xtemp / (xmax-xmin)
xmin and xmax are respectively the lower and upper time boundaries of the original data (in our example 0 and 6.28). Of course you have to keep track of whether you deal with sec, msec etc.

POWER SPECTRUM   SPEC_command

The SINGLE_SIDED_power_spectrum of an electrophysiological recording can be calculated by calling SPEC (cr). The program will ask whether any continuous value should be cancelled (Y/N) and will then ask for the record number. Enter it with the letter indicating whether you want the calculation on the current or on the voltage. If you want the spectrum on the current of record 26, you must enter i26. The result of the calculation will be in two strips xts and ys0. You can trace the result after having set the parameters of the two strips (for getting an idea, go with an automatic scaling). If you wish to perform other power spectra and to keep these data, you must transfer them in strips before activating spec again. Spec will overwrite in the same strips (xts and ys0). However, you can perform several spectrum at once if you give the order in a same command: if you enter i26,30 the power spectrum of current record 26 to 30 will be calculated and loaded in strips ysn with n=0,1,2,3.
NOTE: spec can be used as a operator acting on strips. You could very well write:
y3=spec(y1)-spec(y2)

In this type of operations on strips, the spec_operator works only on x,y or z strips, not on i or v-strips. Therefore, to work with strips but on data, load the data in strips first (y2=i18).
DISPLAY OF THE POWER SPECTRUM:

If the original data is a current as a function of time, y3, the result of the power spectrum operation is "current to the power 2". Classically a power spectrum is expressed with A2s on the ordinate. In addition there must be some normalization with respect to the time over which the spectrum was measured. Thus y3 must be normalized by multiplying it by a time value. This is accomplished by writing: y3s = y3 * (xmax-xmin)
where xmax and xmin are the upper and lower time boundaries of the original data. Similarly the abscissa must be given the dimension of a frequency and must be related to the original data. Since we consider only the real part of the spectrum the abscissa is calculated as follows: xf = 256 * xtemp / (xmax-xmin)
where xtemp is the result of the result of the fft calculation and xmax and xmin are the upper and lower boundaries of the original data. See the DEMO manual for an application.

RANDOM_NUMBER_GENERATOR   rnd_operator

The operator rnd allows you to generate a strip of 512 points with random numbers. Thes random numbers are values between 0 and 1. Therefore, to obtain values between -5 and + 5 for example you type:
y= -5 + 10 * rnd

You can obviously generate noisy curves of any type. Here is an example of a function that you could include in the LIBRARY.FUN that will give you symetrical random numbers: FRND(x) = x * (rnd - 0.5)

NUMERICAL_INTEGRATION   INTEGR_operator

This operator calculates the surface under a curve defined by a x-strip and a y-strip between two boundaries. The integration step is equal to the interval between two successive values of the x-strip. The result of the integration is returned in a variable that you define initially. Example: n[3]=integr(x1,y3,min,max)

where min and max are two values of the x-strip that set the lower and upper boundaries, respectively.

INTERPOLATION_PROCEDURE   INTERPOL_operator

This operator performs a linear_interpolation between successive data points. It is very important to realize that you must sort your data before the interpolation (see SORTING_command, below). This operator is used in compute. Its original goal is to allow you to perform operations between two strips which, for example, do not have the same number of points. Another application of the interpolation is when you want to compare curves which have not been recorded on exactly the same time base. It is an extremely useful tool because it can be used for sample determination using a calibration_curve (a typical application: protein measurements on a BSA_calibration_curve). Finally, the interpolation operator can be used whenever you want to decrease the number of data point in a strip; this may be necessary in the initial steps of a curve fitting procedure to electrophysiological data. Indeed a curve fitting on 512 points may take some time on a small computer, depending upon the operations involved (see DEMO manual for an example). The interpolation operator works in the following way:(we assume compute was activated)

y2=interpol(xref,x1,y1)

where xref is the reference strip, x1 and y1 are the data on which you want to perform the linear interpolation. The result of the interpolation will be in strip y2.
It may happen that xref covers a range outside x1. In this case a linear_extrapolation is performed on the basis of the two preceding values of x1. Of course you will have to appreciate yourself whether this is meaningful or not!

Calibration_curve

There are several ways to proceed. Let's start with the simplest one. Assume you have a series of absorption values (y) for known BSA concentrations (x) and a few samples where you know the absorption (y1) and would like to determine the protein content. We shall simply use the linear interpolation on the calibration values. Your sample's values are in strip y1. Our reference will be the y1 strip and our result will be in a strip that we call x1:   x1=interpol(y1,x,y)

All you have to do now is to list x1 and y1 (see LIST in Ch III) or to trace them on the plotter, together with the calibration curve. If you want to be more fancy why not use a curve fitting on you calibration values? For this all you will need is to find a reasonable equation and fit the parameters with fitting (of course if a linear regression does the job, use reglin). Once you have a theoretical curve (an xth and an yth strip) you can proceed exactly as above: x1=interpol(y1,yth,xth)

Clearly, if the calibration curve is linear, or if you work in the linear part of a curve, there is little advantage in going through the fitting procedure. If the calibration function is more complicated, however, your precision will improve.
Reduction_of_data, fewer_data_points (see also DEMO manual)
Let's assume that you want to fit a sum of two exponentials to a decaying curve. our data strip contains 512 points. It is rather long to fit that amount of points. On the other hand you want to be able to determine where you want to take your sample points for the fitting. Easy enough! This is an excellent example on how to use the crosshair to create a strip with a few selected times stored in it. First, trace the graph (trace x y) with the decaying curve. Then activate the crosshair, position it to the 0 time and move the crosshair along the x-axes, TYPING S each time you want store an abscissa value (dont worry about ordinates, this will be the business of interpol). When you are done quit the crosshair and activate compute. Remember that the x-coordinates of the points you saved are in a strip named xcur. So, you write xd=xcur   (this saves the data in xd)

yd=interpol(xd,x,y)   (here is your data reduction: interpol selects for you the y's that corresponds to the xd's)
You are ready for the fit. Once you have reasonable parameters, and if you feel nervous you can always fit the entire set of 512 data points.
Note also that the precision of the crosshair is quite good. It is limited by the screen resolution. See DEMO manual, for an illustration of this procedure.

VARIABLES_creation

Variables can be created with the command compute. A variable has always the letter n and is followed by a number in brackets: for example, n[3]. The number is between 1 and 26 (size of the variable's buffer) A variable can be assigned a value in compute or may be equated to an operation, which itself involves variables that can be altered by other operators of the program. Examples: n[2]=600 assigns 600 to this variable

n[3]=log(n[5]/3300)

Also, it is legitimate to transfer_variables:

n[1] = n[24] or n[4] = 3*n[2]

In the calculator (see Ch. VIII) you can also use variables or print them.
The buffer_of_variables can be displayed with the **var** command.
REGLIN, INTEGR and FITTING are examples of applications that return one or more variables (see DEMO manual for applications).

LISTING_VARIABLES, PRINTING_THE_VARIABLES, VARIABLES_BUFFER
VAR_command

You have seen that variables can be assigned values in compute. These variables can later be changed by other operators of the program (see VARIABLES, REGLIN, FITTING). In order to display the buffer of variables, type PRINTVAR (cr) or print (cr).

LINEAR_REGRESSION   reglin_operator

The reglin facility allows you to perform a linear regression on a set of data defined by an x-strip and a y- strip. The activation of reglin is performed in compute as follows: n[i]=reglin(x-strip,y-strip)

where i is a number smaller than 24, and x-strip and y-strip are the names of the strip to test. The coefficients of the regression_line   Ax+B

are returned in the following variables, which you can visualize with printvar:

A corresponds to var[25]

B corresponds to var[24]

the correlation_coefficient r is in var[i] with i defined by the user. NOTE: now that you have the coefficients, it is easy to trace the regression line through the data. First you create a theoretical abscissa xtheor over the appropriate range (see Ch. III, theoretical_absissa) and then you calculate the ordinate with the strip (in compute):

ytheor = n[25] * xtheor + n[24]

Set the parameters of the two theoretical strips for the display and you are done.
If you use linear regression often, you may want to write a macro command to speed up the operations.

recalcoff and recalcon now are handled within the compute module as well as in the main DATAC list, so you can change the recalculation mode without leaving compute. The default has been changed to recalcoff.

In calls to highfunc.c functions that take parenthetical arguments (see Appendix), any numeric value can be replaced with a variable (i.e., base(n[0],n[1],x,y).

You may activate compute by typing an equation at the main command level with an "=" sign; the sign will trigger a switch to the compute module and pass the command line to compute. The equation is evaluated and you are left in the compute module when the processing is done. This does not work then the left hand side of the equation is a variable (n[#]=).

base(x,y,min,max) computes the average value of y between x = min and x = max. Returns scalar.

wmin(x,y,min,max) finds the minimum of y between x = min and x = max. Returns scalar.

wmax(x,y,min,max) finds the maximum of y between x = min and x = max. Returns scalar.

store(x,v,n) puts the value v into strip x at location n. Lengthens result strip if necessary. If n is negative, puts it in the next available location. Returns a strip. Example syntax: x = store(x,n[10],-1), puts n[10] into the next available location of strip x and returns the result in x. The length of strip x is increased by 1.

cumul(x) returns a strip that is the cumulative sum of x.

concat(x,y) returns a strip that is the concatenation of x and y (if they fit).
x=x<y is a more compact form of concat(x,y).

val(x,n) returns the value of strip x at index n (i.e., x[n]). n can be a variable.

y[n]=n[10] stores n[10] in the indexed n point of strip y (a more compact form of val(y,n)).

n[10]=y[n] returns the indexed n point of y into n[10].

initstrip x : main datac command to clear the contents of a strip, as may be necessary before a concat, cumul, or store function.

ttest(x,y) returns t value for difference between two strips.

isi(x,y,min,max), movave: special spike processing routines (ask).

find(x,y,t0,yp, dir): finds the point in the y strip that is closest in amplitude to yp, starting at time t0 (in x) and going either to the left or right in time, according to dir (dir = 0 is left, dir = 1 is right). This function returns the index into the x,y pair that corresponds to the point. You must use the following construct to get the actual data value:

```
n[0]=find(x,y,2.0,5,1)    ;find the index starting at 2.0
n[1]=y[n[0]]              ;get the y value into n[1]
n[2]=x[n[0]]              ;get the x value into n[2]
```

Strip indices always run from 0 to the maximum number of points in the strip. Max is 511 (512 points). Examine the root list of strips to see the specific number of points in a given strip.

CURRENT_VOLTAGE_CURVES, IV_CURVES

IVFIX_command IVMIN_command IVMAX_command IVV_command
IVTRACE_command ERAIV_command

DATAC allows you to calculate directly current-voltage curves from electrophysiological recordings stored on a disk. For each I/V curve a pair of strips is created by the program. These strips are called xivn and yivn, with n being 0,1,2,3... as more i/v curves are generated. You can treat these strips with compute exactly as any other strip. For example it may be perfectly legitimate to write the following equation (provided that the two yiv strips were determined at the same voltages!!): y=yiv3 - yiv4

Tracing y as a function of xiv3 or xiv4 will give you the difference between these two i/v curves.
The command IVTRACE (cr) will trace all the I/V curves that are currently in the memory. On the other hand, the command ERAIV (cr) will clear_the_IV_strips named xiv and yiv present in the memory.   You have several options for the I/V curve determination:

IV_at_fixed_time IVFIX_command

IVFIX (cr) will determine the I/V relationship at a given time (note that the program will ask for two time values: in this case you should enter twice the value at which you wish to make your measurement). IV_at_minimum_current IVMIN_command

IVMIN (cr) will determine the I/V relationship for a minimum current value within two time boundaries that you will be asked to enter. An application is the determination of peak inward currents such as INa.
IV_at_maximum_current IVMAX_command

IVMAX (cr) will do the same but for a maximum current value (application: outward currents as IK)
IV_at_different_times IVV

IVV (cr) determines the I/V relationship at a given time for the voltage and at a different time for the current (application: inactivation as a function of previous holding voltages).
Typical command: ivmin (cr) then you type whatever is required.
NOTE: operations between I/V curves measured at different voltages can be done (provided they are in the same voltage range) by using a linear interpolation (see INTERPOLATION in this chapter). Be careful to have the data sorted before doing an interpolation.

CURVE_FITTING, FITTING_PROCEDURE, BEST_FIT FIT_command

The FITTING_command allows you to adjust the parameters (up to 9) of an equation that you write in compute in order to fit your data. The fitting_algorithm performs a least_squares_minimisation of the parameters exploring the possibilities with a structure called a SIMPLEX which is contracted, expanded and reflected at each iteration (REFERENCE: M.S. Caceci, W.P Cacheris (1984) Fitting curves to data: The simplex algorithm is the answer. Byte may 1984, 340-362). The fitting will require an initial guess on the parameters. It will also be your responsability to set the magnitude of the steps used to modify the various parameters. When you activate FIT, a display typical for this module will appear on screen. The way to handle the display is exactly the same as that already described for PARAM and STRIP. You simply have to type the name of the variable you want to adjust and give your entry. Several entries can be nested. The meaning of the various entries are:

xdata label of the x-strip to be fitted
ydata label of the y-strip to be fitted
yth label of the theoretical strip to be adjusted

(after your entry the corresponding equation will be displayed) should there be an error in yth, correct it in COMPUTE and come back in FIT, nothing will be changed in the other variables, so that you can continue rightaway. niter number of iterations you wish (ex:100)
ndisp number. You will have a report every ndisp (5 if ndisp=5) nvar number of variables to be fitted. Once entered a new display will appear wich you may also fill
v0 (1,2,3..) initial value of the first (second etc) variable fitted
i0 (1,2,3..) value of the step on the first (second etc) variable fitted. This value can be kept small if you have a good feeling about the range it should have
e0 (1,2,3..) minimal error where you tolerate interruption of the fit. (does not need to be given, but can be typically 1e-5). The fit will stop if the minimal error between 2 run falls below your level for all you variables. maxerrs does not need to be given

When done, type run. You will have reports on the evolution of the various parameters on the first time through and then every ndisp. If something goes wrong, interrupt with CTRL-C, and check your theoretical equation and your initial parameters and steps. Otherwise fit will stop when a minimum is reached or when niter is reached. You will have a report on the error of the overall fit in terms of a standard deviation. NOTE at the end of the fit you will see variables displayed. Should you wish to proceed the fitting starting with these variables type CONTINUE.
For an illustration of FIT, see DEMO manual.

Graph_of_a_fit

Should you wish to display the result of your fit as a continuous line, even if you actually used discrete points proceed as follows: first create a theoretical abscissa (say xth) on the appropriate range for the data (see Ch. III, theoretical abscissa). Then change in your theoretical strip yth all x's to xth's and calculate it (in compute). Set the parameters of your strips for display and trace the data and the theoretical curve.

SORTING_DATA SORT_command

The command SORT (cr) allows you to sort data present in two strips (x and y, with y, a function of x). The sorting procedure will simply rearrange_the_data in x and y but in growing_order_of_x. This is useful when you want to display the data connected by a line.
Example: sort xd2 yd2 (cr)

will put the data in yd2 in growing order of xd2.

DIFFERENTIAL_EQUATIONS DIFF_command

DATAC will solve multiple differential equations (7) using the Runge-Kutta integration method. Also, up to 7 variables can be set independently. These variables can be constants or can themselves depend on the result of a differential equation. When you activate DIFF, a specific display appears on screen. Here again the same rules apply as in PARAM or STRIP (write the name of the thing you want to enter or modify and type your equation or number or variable). The meanings are: limit is a number representing the upper limit on which to calculate step value of the integration increment (the min is limit/512) time label of the strip in which the integration variable will be loaded. This is the strip you will often use to plot the result of the integration. i1 (2,3...) initial condition of equation 1,2,3 etc
equ1 (2,3..) statement of the equation (see below)
n1 (2,3..) statement of the variables see below.

NOTE entries can be nested

In the Demo manual, an example is given of an application of calculation on a 3-compartment system (Close-Open-Inactivated) with the initial condition that only the close compartment is occupied (i1=1). The 4 rate constants are given as n[i]=number. Note that the classical nomenclature
dc/dt = -n[1]*C + n[2]*O
has been replaced by
xc = -n[1]*xc + n[2]*xo
To activate the calculation type run.
In the Demo, the compartments C, O and I were plotted as a function of the integration variable here xt.

VI
PLOTING

A plotter can be used in combination with DATAC. The plotter interface was designed for the Hewlett Packard 7550A plotter (see also in the INTRODUCTION) The PLOTTER_command, PLOT (cr),activates the plotter The PLOTOPTION_command sets the parameters of the plotter.

The PAGE_command ejects_the_paper.

The PLOTOFF_command disconnects_the_plotter.

The physical representation of a plotter_tracing_zone is shown in Fig. 5. The xpos and ypos parameters of the text strips are given with respect to the bottom and left parts of the frame. The 0,0 point is at 4 cm of both axes. Two text are traced. Note that the xpos of both text was the same (12.98 cm with respect to the frame) but in one case the text was centered with respect to this value using LO 14 in tpar.

SETTING_THE_PLOTTER_PARAMETERS OPT_command

The command OPT (cr) gives access to the various parameters of the plotter. The parameters that are implemented are the character set, the size and orientation of the characters. The pen speed, acceleration and force can be set and the pen color can be selected. In order to use the parameters set in plotop, THE FLAG P_SEL MUST BE ON. To change a parameter in plotop simply type the parameter name followed by the new value.
Plotter_selection_flag P_SEL_flag

This is a flag which determines whether DATAC uses or not the parameters of plotop for the HP plotter. Parameters are used when the flag is set to 1. If the flag is 0 the default parameters of the plotter are used.

Plotter_characters

An appropriate SIZE is around 2.0. A nice little SLANT is 0.1 (just a touch). The nicest CHARacter set is 10. CHORD determines in how many segments a circle is traced. The smaller it is, the nicer the circle but the slower the plotter. For ordinary plot a value of 5 is fine (this is actually the default value). The parameter CV should smooth the curve; I have not been to impressed so far.
Pen_control

The speed, acceleration and the force of each pen can be selected. The PEN_SPEED is in cm/sec and the range is between 1 and 80. The PEN_ACCELERATION is in g's and the range between 1 and 6. The PEN_FORCE is in grams. The code is 1=15gr, 2=24gr, 3=30gr, 4=36gr, 5=45gr, 6=51gr, 7=57gr, 8=66gr. To set the speed of pen 3 to 60 type pen 3 60 0 0 (cr). To change the speed_of_all_pens at once, type pen 0 60 0 0(cr).

PAGE_EJECT PAGE_command

This command allows you to eject the sheet of paper present on the HP 7550A plotter. A new page will be automatically loaded. Obviously, this command will only work if the plotter is enabled (see PLOT). Also verify that the plotter is in the auto_load mode (* displayed in the LCD on the plotter). If not, activate the autoload command.

PLOTTER_ENABLE PLOT_command

The command PLOT (cr) initializes and activates the HP 7550A plotter. The plotter will trace whatever graphic you trace on the screen as long as you do not send the command plotoff (cr).

PLOTTER_OFF   PLOTOFF_command

This command disenables the HP7550A plotter. You should send it whenever the plotter is enabled (by plot) and you wish to trace a graphic on the terminal exclusively.
SENDING_TEXT_TO_PLOTTER   PTEXT_command

Remember that you can control the text strip you decide to plot (see TEXT command). You can also control the parameters of these texts (TPARAM command). Finally, you can plot the texts with ptext. To plot the texts 1 to 5 and 7, type: ptext 1;5,7 (cr)

VII
SAVING, RETRIEVING AND TRANSFERING INFORMATION

OVERVIEW

It is often necessary to transfer into DATAC information that was previously stored on disk. This information may represent data resulting from a previous run of DATAC itself. Alternatively, the information may represent data manually edited in a file or data that were obtained while running a program like MULTIPLAN. These various data are not stored in the same format and, in order to be able to handle them, DATAC must have them in its own format. On the other hand, if information generated by DATAC is to be used in MULTIPLAN, DATAC must be able to return files with the appropriate format. To address this format_problem, two modes of storage and retrieval of information were implemented: -The first mode deals exclusively with information which is directly DATAC-compatible, and is accessed by the save_command and retrieve_command described in the section save and retrieve in DATAC-compatible format.
-The second mode deals with ASCII files, whether in the symbolic mode (like MULTIPLAN files) or simply written by a text editor. This mode is accessed through the read_command and write_command and will be described in the section transfer of information with ASCII files.
Data are stored on disks in files. In order to save data, a file must be first openened (if it exists) or a file must be created. Similarly retrieval of data will require the opening of an existing file before the data can be read. You have to aware of this in order to understand some of the inputs the program will expect from you when you wish to save or retrieve data.

SAVE AND RETRIEVE IN DATAC COMPATIBLE FORMAT

SAVE_command, RETRIEVE_command

DATAC uses binary information stored in strips, either data strips or text strips. Both kind of strips can be saved or retrieved.

SAVE_command, close_command

If you wish to save a strip type save (cr); the program will ask for the name of a strip. If it is a data strip, enter the name (ex:x3); if it is a text strip, enter the number of the text strip. If it is the first saving operation in a particular run of DATAC, the program will ask for the name of a file. You will be asked to type:
- the device (ex a:)
- the file name

ex: A:test

NOTE: the extension .STR is automatically set by the program. If the file does not exist, the program will create a file with this name and save the strip (the message save operation will be displayed). If the file exists, the program will ask whether you wish to overwrite over the existing data in the file or whether you wish to add the new data at the end of the existing one (append). Make you choice and type the corresponding letter. The save operation will be executed. If anything goes wrong in the save operation an error message will be displayed. IMPORTANT: once a file has been opened, subsequent save commands will cause the program to save strips IN THIS PARTICULAR OPENED FILE. This has been done on purpose, because it is convenient to save entire sets of strip together (when preparing figures for example). Should you wish to save a strip in another file, you must cause the program to close this file and to open another one. This is done by activating the command CLOSE. This command will cause the closure of the opened save file. Therefore if you now type save again you will be asked for a new file name (proceed as explained above).
You can save_all the data strips and text strips present in memory at once by typing SAVE ALL (cr). The program will ask you for a file name and will save all strips automatically. RETRIEVE_command

This command allows you to recover strips of data or strips of text that were previously saved while running DATAC. When you call retrieve the program will ask you to enter the name of a file. You do not have to give an extension since the extension .str will be assumed (see above). The program will then ask for a strip name. If you want to retrieve_all the strips contained in a particular file, give initially the command retrieve all (cr) (the program will ask for a file name). NOTE: Before you do a retrieve all operation, make sure there is enough memory available in the heap (see MEMORY command).

IMPORTANT NOTE: retrieved strips having the same name as already existing strips in the memory will be overwritten on the existing strips (strip_overwrite). If you want to be sure not to loose some strips you just created better save them or give them another name and set the parameters of these new strip equal to the old ones (compute, strip, template, default will do that in litterally seconds) Once you have opened a file for retrieval, this file will remain opened (the philosophy is the same as that developped in the save section). Therefore, in order to retrieve strips stored in another file, you will have to close the first file; this is done by simply calling close. This will automatically close the existing retrieve file and allow you to open another file by calling retrieve again.

TRANSFER_OF_INFORMATION WITH ASCII_FILES

READ_command, WRITE_command, SYMBOLIC_format, EDITOR_format

Data contained in strips can be read from or written into an ASCII file. DATAC can handle two types of ASCII files. One format is MULTIPLAN compatible; this format is called symbolic. The other format allows the user to edit data files that can then be transformed into strips by datac; this format is called editor. The basic operations are similar to that seen in the save and retrieve operations. Here again data will be stored in files which will have to be opened for data transfer. The difference is that the read/write operations have to be done in one of the two formats just mentioned. The program determines automatically the type of format on the basis of the read/write instructions you give. When necessary, the program will ask you to enter a file name in order to be able to store or retrieve the data.

Operations_in_symbolic_mode

In this mode the read or write instruction must be:
read (or write)   strip-name   column number

examples:
(i) read x2 3
this instruction reads data from column 3 of a symbolic ASCII file and transfer them in strip x2
(ii) write y 1
this instruction writes the data contained in strip y into a symbolic ASCII file with the label of column 1

Operation_in_editor_mode

In this mode the read or write instruction must be:
read (or write) strip-name   identifier (a letter)

examples:
(i) read z9 a
this instruction will transfer all data having the identifier a from an ASCII file into strip z9
(ii) write x5 c
this instruction transfers the data of strip x5 into an ASCII file with the identifier c

As an example of application, imagine you have a series of binding values of a ligand to some substrate for several concentrations of the ligand. You can create a file with the editor which may look like that: c.01   b1

```
c.1   b1.5
c1    b1.3
c5    b2.1
c10   b2.3
```

In this file the identifier for the ligand concentration is c and that for the binding is b. These two sets of data can be transferred into two strips by two successive READ commands such as:

```
read x c

read y b
```

The data are now contained into two strips, x and y and can be treated by DATAC (curve fitting, Scatchard plot etc)

VIII
MACRO COMMANDS

DATAC includes a macro interpreter which allows you to define a series of DATAC commands that will be executed when you activate a single macro command. MACRO commands are useful for speeding up operations that you perform repeatedly and which require the same sequence of commands. All the commands that can be typed by the user can be executed in a macro with a few exceptions (see below, macro rules). Evoking a macro actually consists in redirectioning the input stream from the standard input/output device to an ASCII file which will be read in the same way as if the commands were typed sequentially on the keyboard.

MACRO_COMMAND_EDITION
Editing_a_macro_command is simple. With any editor (use EDIT if you are running DATAC) create an ASCII file. The only requirement is that the extension of the file must be .mac (example of a macro name: TRACE.MAC). The file name cannot be temp or temp1 to temp8, since these names are used by the program for the macro expension. Having created the file, you write the commands exactly as you would give them in DATAC. When you are through, get out of the file. EXAMPLE: suppose you always use some well defined parameters when you display your electrophysiological curves. Then it is worth creating an INITIALIZATION_MACRO_command. You activate EDIT and create INIT.MAC:

```
param
tbeg 5 tend 35 imin -1500 imax 0 vsel 0 ismo 10
```

With this macro, after activating DATAC you can set your parameter by simply typing MAC and then init.

MACRO_EVOCATION MAC_command

A macro command can be used at any time by typing the command MACRO (cr) followed by the name of the macro you want to activate (you do not have to give the extension, since .mac is a requirement). Example: mac init (cr)

MACRO_RULES
Almost all commands can be used in a macro_body. You can even evoke macro command inside a macro command. Macros_can_be_nested up to 8 levels. The only commands that cannot be executed automatically from a macro file are those involving the crosshair reading and text edition. The macro offers you the possibility to perform loops. This can be very useful when a given operation must be executed on different electrophysiological records, for example. The loop begins with LOOP and end with END. Within a loop each time a % is encountered it will be substituted with a number. As an example, suppose you want to make an amplitude histogram on single channel currents recorded in traces 1 to 50. You could write the macro AMPLIT.MAC: recalcoff

```
rem         careful, see y=y+... below (this is ideal for an infinite loop) compute
y=cte0
rem         to initialize y at 0
LOOP 1;50
y=y+(histo(i%,0,20,4)
rem         i% will become i1......i50
END
```

NOTE

(i) the lines starting with the 3 letters rem will not be used in the macro and are simply comment lines.
(ii) LOOP and END must be written in CAPITALS

(iii) within a LOOP the maximum is 50.

(iv) you can use LOOP with non-consecutive numbers. It would be ok to write
LOOP 1,7;12,25;30

INTERRUPTING_A_MACRO, MACRO_COMMAND_INTERRUPTION

You can interupt a running macrocommand at any time by typing simultaneously the keys ctrl and c (ctrl-c interrupt). This will return you to the main part of DATAC.

PAUSE_IN_A_MACRO_EXECUTION PAUSE_command

Pause is a command we use in the demonstration macros. A text can be displayed during a pause. The pause is interrupted automatically when you type any key on the keyboard. A pause command must always be terminated with an end (which must be in lower case!), whether or not text is being displayed.
Example:

```
tr x y
pause
THIS IS A NICE GRAPH
end
pause
THIS IS THE END OF DEMO3
end
```

NOTE that we carefully avoided to say END OF DEMO3, because the line would begin with an END, and the macro interpreter would desperately look for a return point from a LOOP which does not exist!

MACROS

Macros may be used in the new version just as in the old version. Nested macros are permitted. Only two major differences are evident, one being the specification of LOOP style arguments (called "sequences"), and the other being the addition and specification of parameters passed to macros.

Loop arguments (sequences)
The **specification of records or LOOP arguments** has been extended. The syntax is:
a;b/xc (also a,b,d,f,...).
where **a** is the starting number for the sequence
b is the final number of the sequence
if only a and b are specified, then the sequence a;b is equivalent to a,a+1, a+2, ... b.
x is a step control variable. The meaning of x depends on the character **c**.
c is a control argument (single character). c can be:
nothing: then x is the increment for the steps between a and b.
'**n**': then x is the number of steps between a and b.
'**l**': then a to b is taken in logarithmic steps, with x steps.
'**r**': then a to b is taken with x steps between a and b, but the returned numbers are randomized.
'**s**': same as 'r', but the steps are made logarithmically.
'**a**': causes alternation between the two values given in the list.

The ';' separating a and b sets these as a range. Individual values may be returned using ',' as a separator (this, "a,b,c,d" will successively return a,b,c and d. A comma may also be used to separate multiple sequences that are to be concatenated. Thus, "1;10/2,20;100/5" will go from 1 to 10 by twos, then from 20 to 100 by fives.

A space terminates the sequence so it is important that the sequence by typed with exactly no spaces between any of the arguments.

For most commands dealing with getting records from disk, the arguments are assumed to be integers (if not, they are adjusted to be integer values). However, for some commands (i.e., sequence command in **pulse**, or index function in compute), the arguments may take on floating values. The numbers may be negative and sequences can run in either increasing or decreasing order.

Thus, the command LOOP 1;18/3 will cause the program to replace the %'s with 1,4,7,10,13,16 in successive cycles through the loop. The /3 means "by three's" for example. In addition, negative loop values are permissable in certain circumstances. Note that loops are no longer limited to 100 or 200 cycles, as in version 3.0 of DATAC. For non-randomized sequences, there is no limit to the number of elements in a loop sequence. For randomized sequences, since they must be precomputed, a limit is imposed by the amount of memory available to the program at the time the sequence is generated. In practice, this will almost always be larger than the user will need (probably about 1000).

Loops may still be nested (each nest maintains its own copy of the sequencer control block on the stack; thus the depth of nesting will depend on available stack space).

Macro Substition codes

%: a "%" is substituted with successive elements of the loop sequence argument(s).

@: "@" means indirection and expects an argument. Thus, x@+1 will replace the @+1 with the value of %+1, so that if % would have been 2, the result will be x3. x@-2 will be replaced with, for example if % is 2, "x0".

#filename: use only on calling command line. This reads a file of substituition definitions with a default .msb extenstion, and substitutes arguments in the macro. Examples to follow.

$n: Sequential arguments on the calling macro command line are stored into $0..$9, and can be substituted in the macro. An argument starting with # will be ignored. Any string, number, or sequence is a valid substitution parameter. There is a limit of 10 command line arguments.

Special Macro

When DATAC is started, a special macro, called "**init.mac**" in the current directory is opened and executed. This macro may set up the user's custom parameters for data acquisition, display, etc. Frequently, this macro will call the macro that sets some default display parameters in the **param** module, and might set the default filename extension for data acquisition or for reading data files.

Macro Help

The command "**hmacro** filename" (help macro or just "hm filename") from the main command line of DATAC lists all of the lines in a macro that start with "REM" (but not those starting with "rem"). This is useful for jogging one's memory as to how the arguments to a macro should be entered (since the macro cannot prompt for input). When writing a macro, use REM lines to describe the name of the macro, the purpose of the macro, and the arguments expected with their order, as well as any unusual conditions that may be necessary for the macro to function.

IX
ACCESSORY TOOLS

TEXT_EDITION   EDIT_command

When you activate EDIT (cr), you actually activate a simple editor. This editor will allow you to write data file while running DATAC or to write macrocommands (see Ch IX). When activating EDIT followed by a filename and extension (for example edit test.001) the file will be created or reopened if existed and the cursor will be on the top of the screen. The text of the file will be displayed. The cursor will be on the top line. To begin editing type a cr which will bring the cursor down one line. Type your text. Successive cr bring you down. The arrows on the notepad will move the cursor If they do not work use the Wordstar commands (ctrl-s=LEFT, ctrl-D=RIGHT, ctrl-E=UP, ctrl-X=DOWN. To get back to the top line type the key "home" on the notepad (or ctrl-T). To exit type "exit esc esc" on the command line (esc is the escape key). To abort type "quit esc esc". To insert text in a line type the "ins" key on the notepad (or ctrl-N). This key work as a toggle (either overwrite or insert). You can kill a line with ctrl-y (like in Wordstar). To insert several lines in a text: "esc ni", where n is a number. To delete n lines: "esc nk". To scroll the display on page down, ctrl-P, and up ctrl-O. To read for example 100 lines from another file write the following text on the command line: 1000r'filename.ext' and then type the escape key twice
The same applies to a write command with 1000w'name'

POCKET_CALCULATOR   CALC_command

The user can activate the pocket calculator with CALC (cr)and then return to the data acquisition or replay program for further processing. The calculator can perform calculations on equations that you enter on the terminal. When you activate the calculator, the program will display: ENTER YOUR COMMAND OR QUIT. To perform an operation, write it Example to divide 500 by 35 type 500/35 (cr)

Parenthesis must be used when operation_precedence must be controlled (for example when an addition should precede a multiplication). Operators must always be followed by a parenthesis (ex: sin(3.14*26)). Available operators are: sin, cos, tan, asin, acos, atan (for trigonometric operators angles must be given in rad); sqrt (square root), exp, ln, log, pow.
examples:
58*log(5.4/150)

2 to the power 64 is 2pow64 (cr)

HOW_DO_I_GET_OUT_OF_HERE, EXIT

QUIT_command BYE_command

BYE or QUIT terminate the DATAC program and returns control to MS-DOS. The program is protected against accidental termination and will ask you whether or not you really want to quit.

HIGH_QUALITY_GRAPHIC

With the 7550A HP plotter, it is possible to get high quality graphic, such as is required for publication. This type of graphic is more complicated to realize because the user has many options and must therefore have a good representation of what he wants and of what is actually going on in the display procedure, things he normally does not worry about in simpler display. The best way to get a feeling for how to proceed is to run a demonstration. The various macros DEMOFIGx.MAC (x being a number see DEMO manual) illustrate the progressive shift from a simple display to a ready-to-publish figure. Run these demonstrations and list them to see the commands that were used. Remarks included in the macros will also help you.

X
Matrix_Computations

Datac version 3.0 or above includes a matrix interpreter, which was incorporated to extend the computation abilities of the program to vectors and matrices. This module was originally designed for matrix computations involved in single-channel activity analysis and modeling. Therefore, computations are restricted to real vectors and matrices. The maximal size of a matrix is 20 by 20 (standard floating points). Operations on matrices can only be executed by a special module of DATAC. This was done to minimize possible confusions with other DATAC functions that operate on strips. Results, however, can be transfered by ways of variables between the various parts of the program. The matrix commands can be executed once the special module has been activated with the command matcomp or simply mat. The text matrix computation is displayed in the top part of the screen to remind the user that he is in a particular module of the program.
commands in matcomp

Matrices are stored in a binary tree as strips are. Therefore, many of the commands and structures are very similar those used for strips. For example, matrices can be created, saved or retrieved in a similar way as strips.
The commands are:
entry to create a new matrix
display to display a matrix
mroot to display all matrices currently in memory
mfree to erase all matrices currently in memory
msave to save all matrices in a file on disk
mretrieve to retrieve all matrices from a file on disk
NOTE that these commands can be executed exclusively from within the MATCOMP module. The user can leave the MATCOMP module and return to the main modules of DATAC by simply typing any legitimate command of the main program, such as ERASE, COMPUTE, TRACE, etc.
creating_a_matrix, ENTRY_command
When the command entry is typed the program will ask the user to enter the name of the matrix. A matrix_name can have up to 7 characters but the first character must be a capital letter. A lower case first letter defines a vector_name. All the explanations that follow applies to vectors as well. Once a valid name is entered the program asks for the matrix_size (max 20 by 20 (standard floating points); for a vector it is 20 by 1 or 1 by 20). The user must enter two numbers separated by a space, the first number represents the number of rows, the second the number of columns. The size is checked by the program. An entry may be rejected if the wrong dimensions are entered (for example, 3 3 for a vector, which should be either 3 1 or 1 3) or if there is not enough space left in the computer memory. If the entry can be accepted the program wil ask the user to type the matrix elements row by row (elements must be separated by a space). NOTE All the information can be typed on a single command line as usual in DATAC. Example of an entry:
entry Atest 3 3 1 2 3 4 5 6 7 8 9
The latter command creates a matrix with the name Atest and having a size of 3 by 3. The elements of the first row are 1,2 and 3. The element of the third row and third column is 9. NOTE that there are reserved_names: I (the identity matrix), det (for determinant), eigen (when calculating eigenvectors and eigenvalues) and tr (for transpositions). displaying_a_matrix, DISPLAY_command
A matrix that exists in the computer memory can be displayed with the command display followed by the name of the matrix.
Example: display Atest
listing_the_matrices, MROOT_command
The list of all matrices stored in the computer memory and of their content can be displayed using the command mroot.
NOTE: you can temporarily stop the display by typing ctrl-s and restart it in the same way. This can be repeated several times.
flushing_the_matrices, Mfree_command
The command MFREE erases all the matrices that are currently in the computer memory. If some precious material is there, make sure to save it before activating this command! saving_matrices, Msave_command
You can save at once all matrices currently in the memory with the command MSAVE. The program will ask you to give the name of the file in which to save. The extension of the file is .mat by default. If the file already exists an overwriting warning message will be displayed; at that point it is possible to cancel the command by answering N. NOTE: in the current version of DATAC the selective saving procedure of individual matrices is not implemented. retrieving_matrices, Mretrieve_command
The command MRETRIEVE allows the user to recover at once all the matrices previously saved in a file on disk. Selective retrieval of individual matrices is not implemented in the current version of DATAC. When you activate MRETRIEVE, a warning message will be issued indicating that the retrieved matrices will be overwritten on matrices which are currently in memory and which have the same name.
Operations_on_matrices
Operations on matrices are activated by writing an equation in the matcomp module instead of any of the commands listed above. The program detects that it should perform an operation and the computation is automaticaly initiated. For example, if you are in MATCOMP and write
B = A * C

the program will compute matrix B as result of the matrix product of A times C.
NOTE: It is important to remember that the interpreter is designed to deal only with real matrices; therefore, it will not function with a

complex matrix. Although the interpreter allows matrix computation which resemble in many ways strips computations you cannot mix strips and matrices.
List of the built-in commands and legitimity of their application In the following list the symbol M represents the matrix type, the symbol v represents the vector type and s the scalar type.
addition (+) applies to M, v and S

subtraction (-) applies to M, v and S

multiplication (*) applies to M, v and S

division (/) applies exclusively to S

parentheses() apply to M, v and S

eigen(M) computes the eigen vectors and values of M

I(n,n) builds an identity matrix of size n by n

tr(M) or ~M performs row column transposition (v or M)
M^-1 matrix inversion

det(M) calculate the matrix determinant (result is S)

M(r,c) read or write a matrix location (result is S)

v(n) read or write a vector location (result is S)
n[m] read or write into a variable (result is S)

M(r,0) or M(0,c) extracts a row or a column from M (result is v)

Examples of computations

The few examples below are not meant to be exhaustive, but should help the user in figuring out the basic principles. NOTE: all the user entries are written in italic. For simplicity we ommited the prompt character of the matcomp module (*) which normally precedes the user's entry lines example of matrix_entry
entry A
enter the number of row and column ...
3 3
enter 3 data in row 1
3 1 6
enter 3 data in row 2
0 2 5
enter 3 data in row 3
1 1 1
inversion_of_A_matrix
B=A^-1
This is it, and now you can display the result with:
display B
NOTE: do not hesitate to use parentheses when in doubt on the result of an operation. Example: C1=A*A^-1 does not give the same result as C2=(A^-1)*A
In the first case the product will be executed before the inversion. creating identity_matrices
B=I(4,4) creates a 4 by 4 matrix with one's on the diagonal. It is perfectly legitimate to write: C=I(3,3)*A NOTE: we are aware that there is an inhomogeneity in the writing in the sense that an identity matrix is created with the essentially the same notation (except for the use of I) used for extracting a scalar from a given matrix position. extracting vectors or scalars from a matrix
a=A(0,3) generates vector a of dimensions 3 by 1 (3x1) which is made up of the elements of the 3rd column of matrix A. Therefore, this operation corresponds to the extraction of a column. b=A(1,0) generates vector b of dimension 1 x 3 which is made up of the elements of the first row of matrix A. Therefore, this operation corresponds to the extraction of a row. n[2]=C(1,2) transfers the element of row1, column2 of matrix C in a variable n[2], which is a scalar. NOTE that you can put a scalar in an existing matrix in the following way: A(1,1)=3 will load the value 3 in location 1,1 of A
multiplications
It is assumed that the user is familiar with the results of vector or matrix product. For example the row-column product of two vectors is a scalar. This scalar could become an element of a matrix or be stored in a variable, depending upon the notation. Consider a 3 by 3 square matrix, A. You could write: n[0]=A(2,0)*A(0,3)
B(1,1)=A(2,0)*A(0,3)

In the first case the result is in the variable n[0] and in the second case the same results becomes the element 1,1 of a matrix B. The following multiplications are legitimate:

```
B=3*A
C=(6/25)*A
Apow3=A*A*A
```

matrix_determinant

The determinant_of_a_matrix is a scalar and can be taken as: n[4]=det(A)

NOTE that the parentheses are required. In the current version, it is not possible to take in one operation the determinant of a transposed matrix (det(tr(A))) matrix_transposition

To transpose_a_matrix, write:

B=~A or B=tr(A)

NOTE that if you use the notation tr the parentheses are required. The transposition has a higher priority than the product. Therefore, it is equivalent to write B=~A*A or B=(~A)*A

In general, however, we recommand to put parentheses when in doubt about the priority of operations. eigenvectors_and_eigenvalues

The procedure for extracting eigenvectors and eigenvalues functions exclusively for square matrices that are real, asymetric and singular. The computation program is a transcription of a program written in ALGOL by J.H. Wilkinson and C. Reinsch in the Handbook for Automatic Computation, Volume II Linear Algebra, chief ed. F.L. Bauer Springer-Verlag, Berlin Heidelberg New York (1971). This program includes several modules: BALANCE rearranges a matrix according to the machine's precision ELMES and ELMTRANS transform the matrix to a computable format HQR2 performs the computation

BALBAK rearranges the resulting matrix (reverses balance) To compute the eigenvectors and eigenvalues of a matrix, write:

C=eigen(A)

eigenvectors will be in C and eigenvalues will be in a matrix created by the program and named MEV. NOTE you can save matrix MEV in another matrix (A=MEV).

Which shall give an example of how to compute the eigenvectors and values of a 3x3 matrix. Consider the following kinetic scheme:

```
     50      50
C____ 0 _____ I
     .5     .05
1        3        2
```

where C is the closed state, which we shall label as state 1, O is the open state (labelled 3 for reasons that are explained in DEMOPG, see Demo manual) and I the inactivated state also named state 2

The matrix Q1 reads like that:

$$
\begin{array}{cccc}
 & 1 & 2 & 3 \\
1 & -50 & 0 & 50 \\
2 & 0 & -0.05 & 0.05 \\
3 & 0.5 & 50 & -50.5
\end{array}
$$

Enter this matrix in DATAC (see procedure above). Then write the following macro (call it mat1.mac, for example) which will allow you to quickly determine the parameters. recalcoff

```
mat
entry R 3 3 0 0 0 0 0 0 0 0 1
entry w 3 1 1 1 1
entry p0 1 3 1 0 0
g=R*w
EV=eigen(Q1)
EIV=EV^-1
n[0]=p0*EV(0,1)*EIV(1,0)*g
n[1]=p0*EV(0,2)*EIV(2,0)*g
n[3]=p0*EV(0,3)*EIV(3,0)*g
MS=.001*MEV
display MS
```

Note that the multiplication by 0.001 is done to return the value in msec. On display you will get the eigenvectors

```
n[0]=-4.990427
n[1]=4.989428
n[2]=0.000999
```

and the eigenvalues (only the real part is written here)

```
l0=-0.055280
l1=-0.045270
l3=0.000000
```

You can now write an equation:

yth=-4.990427*exp(-.05528*x)+4.989428*exp(-.04527*x)+.000999 where x is a timebase strip (make it over 90 msec for example). Plot it and this will give you the probability of opening over time of the 3-state system described above. Note that the sum of the n[x] must be equal to 0. Note also that you could arrange the matrix Q differently, the only catch is that you have to define the R matrix and the p0 vector accordingly. In writing the equation you must also make sure to match the n[x] with the lx.