Time-Resolved EPR Spectroscopy in a Unix Environment

NATHAN M. LACOFF, JAMES E. FRANKE, and JOSEPH T. WARDEN*

Department of Chemistry, Rensselaer Polytechnic Institute, Troy, New York 12180-3590

Received August 28, 1989

A computer-aided time-resolved electron paramagnetic resonance (EPR) spectrometer implemented under version 2.9 BSD Unix was developed by interfacing a Varian E-9 EPR spectrometer and a Biomation 805 waveform recorder to a PDP-11/23A minicomputer having MINC A/D and D/A capabilities. Special problems with real-time data acquisition in a multiuser, multitasking Unix environment, addressing of computer main memory for the control of hardware devices, and limitation of computer main memory were resolved, and their solutions are presented. The time-resolved EPR system and the data acquisition and analysis programs, written entirely in C, are described. Furthermore, the benefits of utilizing the Unix operating system and the C language are discussed, and system performance is illustrated with time-resolved EPR spectra of the reaction center cation in photosystem 1 of green plant photosynthesis.

INTRODUCTION

In the state-of-the-art instrumental chemistry laboratory, the computer has become a prerequisite for data acquisition and analysis because of the large quantity of data that can be produced in a short time. Under such circumstances it is desirable to utilize the computing resources in the most effective manner. Therefore, when such instrumentation is interfaced to a computer, the choice of not only the computer but also the operating system becomes critical.

In our laboratory, it was desirable to create a computer-controlled time-resolved electron paramagnetic resonance (EPR) spectrometer.[1] However, computer implementations developed previously for either conventional[2-11] or time-resolved[12-15] EPR spectrometers have relied upon single-user, single-task operating systems. This approach, however, limits the productivity of both the user and the computing facility.[16,17] Moreover, in these previous applications most of the data aquisition or interface programs were coded in Assembler or machine language, which makes development, maintenance, and modification of the software very difficult. Finally, the operating systems utilized previously are in general not adequate for accommodating a planned computer upgrading because of incompatibility with a new computer's hardware, either in code portability or networking capabilities.

A solution to the limitations discussed above is provided by the Unix operating system with its integral C programming language. As a multiuser, multitasking operating system, for example, Unix facilitates data processing.[17] Just as importantly for data acquisition applications, however, Unix may be modified to support real-time processes.[18,19] Furthermore, the ability to code all of the application programs in C, which is structured, versatile, and readable, provides obvious programming advantages. Moreover, C programs are portable to any system running Unix,[17] and hierarchical networking[20] of several computers is well supported by Unix as Yang et al. have demonstrated.[16] Finally, on the basis of current trends toward Unix usage, computers purchased in the future as part of a planned upgrading can be expected to support Unix. For this reason, in addition to the immediate advantages outlined above, it is advantageous to implement Unix on existing computers in the laboratory.

In this paper, we will describe a computer-controlled, time-resolved EPR spectrometer that is unique in that it operates in a 2.9 Berkeley Software Distribution (BSD) Unix[21] environment that is switchable between real-time and multiuser modes on demand. Furthermore, we will outline the methods that were necessary to create a real-time spectrometer/computer interface within the multiuser, multitasking Unix superstructure. In addition, the software, which is written exclusively in C, will be described in sufficient detail to illustrate how its compact, yet versatile and readable, nature makes C an attractive language for programming not only data analysis but also data acquisition algorithms. Lastly, the capabilities of the system will be illustrated with time-resolved EPR spectra of the reaction center cation, P700$^+$, in photosystem 1 of green plant photosynthesis.

* Address correspondence to this author (BITNET: jtwarden@rpitsmts or Internet:jtwarden@charis.chem.rpi.edu).

HARDWARE

Computer Facilities. Our laboratory is equipped with a Digital Equipment Corp. (DEC) PDP-11/23A minicomputer configured with 256 kB of main memory, three cartridge disk drives (two 5-MB RL01s and one 10-MB RL02), two 0.5-MB RX02 floppy disk drives, and a MINC (Modular INstrument Computer) interface. The MINC interface boards, which are used for instrument control and data acquisition through either front-panel BNC connectors or internal control blocks, provide a real-time programmable clock and Schmitt trigger, 16 12-bit analog-to-digital (A/D) converters, 4 12-bit digital-to-analog (D/A) converters, 16 digital-in (DI) lines, 16 digital-out (DO) lines, and an IEEE-488 controller. Four ASCII serial lines are available via a DLV11-J communications board in the computer; presently a VT105 terminal (acting as the console device) and a Macintosh II (emulating a VT100 or Tektronix 4010 using VersaTerm) occupy independent lines, while a DEC Letterwriter printer, Tektronix 4662 plotter, two Tektronix 4010 terminals, and an AT clone (emulating a VT100 using PC-VT) share the remaining two lines. The Macintosh II may receive text or data files from either the PDP-11/23A along a serial line via the *umodem* utility (employing the *xmodem* file transfer protocol) or the AT clone via a Tops card interface within the clone coupled to an AppleTalk network. A LaserWriter Plus, which is connected to the Macintosh II, may produce publication-quality documents from the downloaded files as required.

EPR Instrumentation. The major components of the time-resolved EPR system as illustrated in Figure 1 comprise a Varian E-9 series EPR spectrometer with an E-101 microwave bridge, a Biomation Model 805 waveform recorder (8-bit A/D conversion at up to 5 MHz), and a Quantel Model 581C Nd/YAG laser (8-ns pulses at up to 10-Hz flash rate). For conventional magnetic field modulation detection, the analog phase-sensitive (PS) detected signal is obtained from either the J307 ($\sim$250-μs filtered response) or the J308 ($\sim$20-μs unfiltered response after modification of the 100-kHz oscillator circuitry[22]) output on the rear panel of the EPR console, and this signal is then routed to the Biomation for fast detection (steady-state, conventional EPR spectra are obtained by

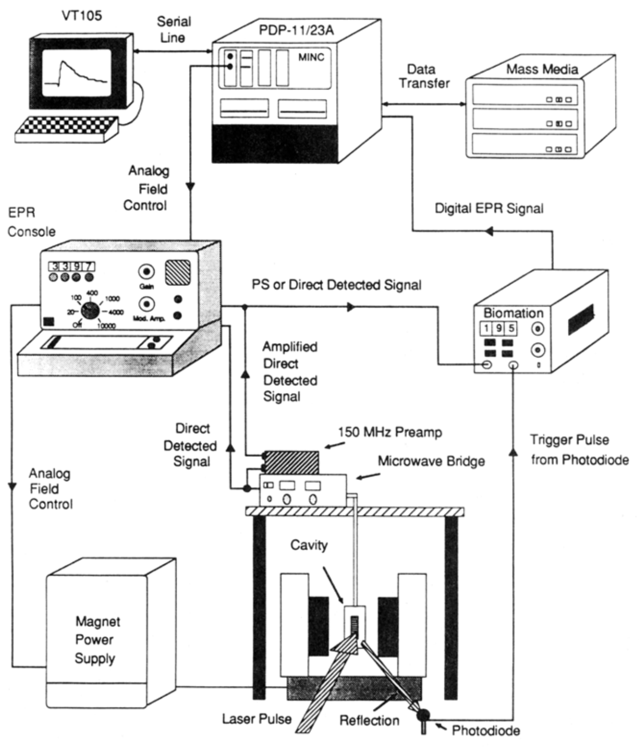


Figure 1. Illustration of hardware configuration and information flow for the time-resolved EPR spectrometer. Information flow and data acquisition timing are directed by C programs operating under Berkeley 2.9 BSD Unix.

routing the PS signal to a MINC A/D converter). Alternatively, for direct detection studies (i.e., no magnetic field modulation) the J1206 (~600-ns response) analog microwave bridge output is amplified by a 150-MHz preamplifier as shown in Figure 1 before being routed to the Biomation. For both detection techniques, a Varian E-231 EPR cavity is utilized, which operates in the TE_{102} mode and has a front window grating (11 × 22 mm) that permits ca. 50% of the incident laser radiation to enter the cavity. Furthermore, both techniques utilize external control of the spectrometer magnetic field (within the magnetic field scan range setting on the spectrometer console) via an analog interface designed by Tschudin Associates (Silver Spring, MD). The other EPR parameters, such as modulation amplitude and gain, are set on the spectrometer console before a scan is started.

These components in conjunction with the computer interface provide for flexibility in the types of EPR measurements that may be performed. For example, to acquire conventional (steady state) or time-sequenced (for slowly decaying species scanned over periods of minutes to hours) EPR spectra, the magnetic field is ramped in discrete steps by a MINC D/A converter, which provides voltage steps as small as ~2.5 mV within a 0–10.24-V range. The field is then stabilized at a given value for a user-specified length of time (see Program Features) by the MINC real-time clock before a MINC A/D converter (conversion rate of ca. 14 kHz) digitizes the J307 output. In addition, time-resolved EPR spectra may be acquired by using the Biomation to digitize the transient signals for either the J307, J308, or J1206 outputs; however, the field stepping and data acquisition timing in this case are performed with the MINC interface according to the steady-state protocol. As shown in Figure 1, the Biomation, operated in the pretrigger mode, records the EPR signal after being triggered by a PIN 10D/SB photodiode (United Detector Technology, Inc., Santa Monica, CA) that monitors the reflection of the actinic laser flash from the EPR cavity. Data from the Biomation are then transferred to the computer through the MINC digital-in and digital-out control blocks by a handshaking protocol (see Program Features). The individual decay traces sent by the Biomation are then summed by the minicomputer; and when the user-specified number of traces is obtained, the computer generates an averaged decay trace that is then plotted on the VT105 terminal to allow the user to monitor the progress of the experiment. A peak data collection rate of 7.9 kB/s (~15 512-pt. decay traces/s), which includes computer averaging of the data, has been achieved.

SOFTWARE

Unix and C. All EPR programs are written in the C programming language and execute in the 2.9 BSD Unix environment. This Unix distribution (based on AT&T, v.7), which totals approximately 37 MB, contains a generic version of the Unix operating system, utilities, source code, libraries, documentation, and manuals. The system as distributed is configured with a generic Unix, which supports all PDP-11 minicomputers and a large number of disk, tape, and terminal devices. The availability of the source code for the operating system and for each utility allows the generic release of Unix to be tailored to the specific requirements of a target computer. For example, the Unix kernel for the PDP-11/23A was reconfigured to include device drivers for the RX02 floppy drives and special functions, such as rtp( ) (see below), as well as specifications for the number of serial lines, the available main memory, and the amount and location of swap space on the cartridge disks. Upon reconfiguration, the kernel was recompiled and linked with essential library utilities to provide a modified Unix operating system that is optimized for operation on a PDP-11/23A minicomputer. Subsequently, hardware drivers and Unix utilities, such as the on-line manuals, Pascal interpreter, spell checker, and text formaters, that were not essential for our present requirements were deleted to provide more available disk space for the users. With these deletions, the modified Unix operating system and utilities together require only 8–10 MB of disk space.

A number of system utilities included in version 2.9 BSD Unix are essential for acquiring real-time data in a multiuser, multitasking environment. The first is an AT&T Laboratories systems utility, phys( ),[23] that maps dynamic virtual memory addresses to static physical memory addresses. Through its function, the MINC boards can be accessed at assigned memory addresses so that the critical operations of data retrieval and transmission can be conveniently software-driven. A second system utility that is unique to 2.9 BSD Unix, rtp( ),[24] facilitates time-intensive data acquisition by locking the given process (e.g., data acquisition programs) into computer memory, thereby disabling the Unix swap scheduler to allow the current process to monopolize system resources. However, normal Unix processes (such as system clock functioning) and hardware interrupts, which always require Unix scheduler processing and include the typing of characters on a terminal keyboard, are not handled by rtp( ). Therefore, full real-time status can only be achieved by the program when these processes and interrupts have been eliminated (see Program Features). This real-time status is critical to the efficiency of data accumulation because the Biomation will send data in a slow mode if the data are not received within the allotted time specified for fast-mode data transmission.

In addition to system utilities provided by Unix, the nature of the C language also proves valuable to the data acquisition process. For example, the ability to read and write values to specific memory addresses through pointer structures, such as *inbuf→status* in Figure 2, makes C well suited to the interfacing needs of a data acquisition program (see Program Features). In addition, the fact that C is an intermediate level language allows the C compiler to produce efficient code that is compact and executes quickly. These benefits of C are then

```
#define CLK 0140020        /* address of MINC programmable CLocK */
#define AD 0140000         /* address of MINC A/D converter */
#define DA 0140060         /* address of MINC D/A converter */
struct {
 int status,buffer;        /* pointer structure to CSR and buffer of input device */
};
struct {
 int da1;                  /* pointer structure for magnetic field D/A controller */
};

sweep()
        /**** SWEEP magnetic field and acquire CW EPR data ****/
{
int *inbuf;                /* pointer to input buffer address */
int ictr,ifield,jcount,p,i,del;
 phys(6,54,07710);         /* map virtual addresses 0140000->0146600 to */
                           /* physical addresses   0771000->0777600 */
 inbuf = 0146560;          /* console address */
 inbuf->status = 0;        /* deny any input from terminal keyboard */
 inbuf->buffer = 0;        /* null buffer contents */
 ictr = -(dwell*100. + 0.5); /* # of clock cycles to count at 100Hz */
 DA->da1=ifield=0;         /* initialize magnetic field to lowest value in range */
 jcount = 4096/npts; /* 4096 voltage steps are in AD converter,so jcount is */
                           /* number of digital units to step for each field step */
 for (p=1; p<=nsweep; ++p) {  /** acquire and average nsweep EPR scans **/
  for (i=0; i<npts; ++i) {    /** acquire an EPR data scan **/
   CLK->buffer=ictr;       /* load clock with # of cycles to count */
   CLK->status= 051;       /* clock counts at 100 cycles/sec */
   while (!(CLK->status & DONE));    /* wait for dwell time to pass */
   AD->status = 01;        /* do the conversion; put result in AD->buffer */
   while (!(AD->status & DONE));     /* poll A/D conversion status until done */
   inten(i) = (1./p)*((p-1.)*inten(i)+(float)(AD->buffer)) + 0.5;
                           /* average multiple sweeps into EPR intensity array */
   ifield += jcount;       /* increment field by jcount voltage steps */
   DA->da1 = ifield;       /* set new field */
  }                        /* end for i loop */
  ifield = DA->da1 = 0;    /* reset magnetic field to initial value */
  CLK->buffer = -(del*100);    /* del seconds of delay before starting new scan
*/
  CLK->status = 051;       /* set clock on and going at 100 Hz */
  while(!(CLK->status & DONE));      /* wait for clock to finish counting */
 }                         /* end for p loop */
 inbuf->status = 0100;     /* allow keyboard interrupts */
}                          /* end sweep() */
```

Figure 2. Portion of C source code for conventional and time-sequenced EPR data acquisition illustrating the interface of the MINC real-time clock and A/D and D/A converters to the EPR spectrometer. The general principles shown for memory mapping, elimination of hardware interrupts, device interfacing using structured pointers, magnetic field stepping, and polling for data are also applicable to time-resolved data acquisition.

realized in the execution speed of all the programs that comprise the time-resolved EPR software system.

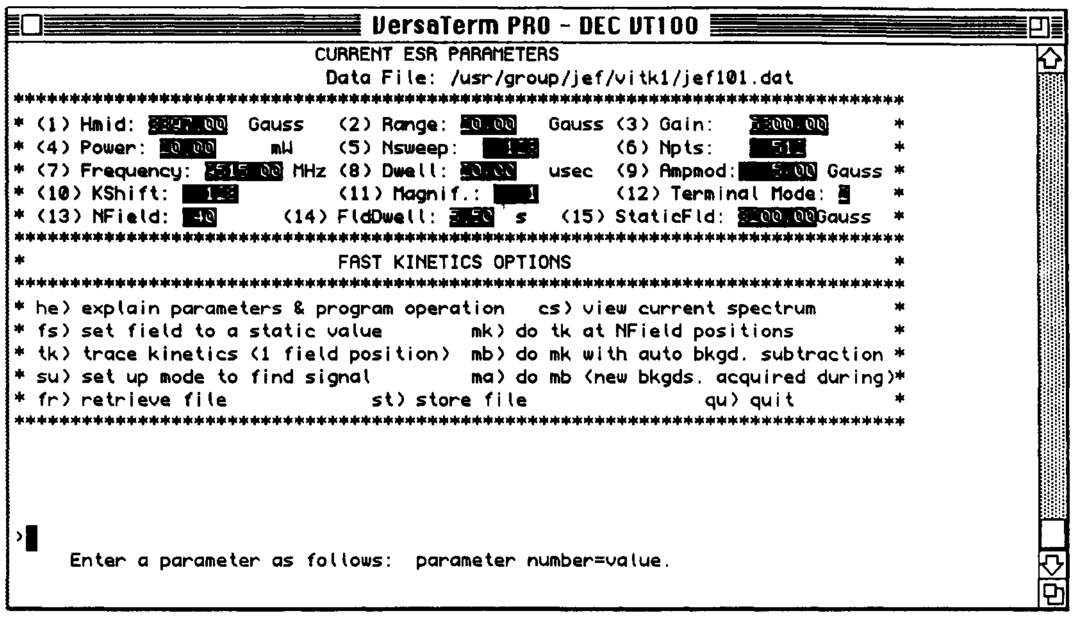
In our implementation, there are three functionally different C programs—**kesr**, **esran**, and **esrplt**—that provide all of the utilities needed for data acquisition, analysis, and presentation, respectively. Organization for each of these programs is provided by header files, which contain global variables and compilation directives, and by multiple source code files, which consist of C functions grouped by the type of task performed. Both types of files, once compiled into individual object code files, are linked together into a single executable file with the loader command, ld( ).[25] Because the complete executable code exceeds available computer memory, only part of the code can occupy main memory at one time; therefore, the memory overlay option of ld( ) is chosen to be utilized during compilation. Each of the three programs are menu-driven, and on-line help menus are available to aid a user who is not familiar with the system.

Program Features. All of the data acquisition applications previously described (i.e., conventional, time-sequenced, and time-resolved) require at some point in their algorithms some of the software features illustrated in Figure 2. The most important of these features is the phys( ) call (see Unix and C for description), which is the first executable statement in the sweep( ) function. Furthermore, the use of pointer structures (e.g., *inbuf→status*) is a second feature shown in the figure; and their use is necessary as a means for the program to specify the operations of reading and writing to the memory addresses previously allocated by phys( ). For example, the *inbuf→status* = 0; statement writes a 0_8 to 2 B of memory starting at the console terminal virtual address (which is set as 0146560_8 by *inbuf* = 0146560;), which is physical address 0777560_8. (Note that since *status* and *buffer* are integer pointer structures, *inbuf→buffer* points to virtual address 0146562_8.) Setting all bits to zero in the console address causes the console keyboard to be deactivated so that spurious keystrokes will not cause the CPU to interrupt, and later, the *inbuf→status* = 0100; statement writes a 0100_8 to the console address to reactivate the keyboard. This is the general method of controlling keyboard interrupts during all data acquisition operations. A third feature shown in Figure 2 is the use of the MINC real-time clock for data acquisition timing, which is shown in the last three statements of the *for* (*p*=1; *p*<= *nsweep*; ++*p*) loop. Writing 051_8 (or 101001_2) to the clock status register sets bits 0, 3, and 5 to 1_2 within the register, which programs the clock to start counting at a rate of 100 Hz. Since the clock zeroes its register when counting begins and ultimately sets bit 7 (overflow flag bit) to 1_2 when counting is complete, the register will contain $010000000_2 = 0200_8$ upon completion of counting. Therefore, the program continuously polls the clock status register until its contents are bitwise equal to DONE (defined as 0200_8) with the *while* (!(*CLK→status* & *DONE*)); statement. The fourth common feature illustrated in Figure 2 is the adjustment of the magnetic field, which is done by setting one of the MINC D/A converters to have a specified digital value using the pointer structure technique (*DA→dal* = *ifield*; statement). The amount of time in which the magnetic field is allowed to stabilize (*ictr* value) before data acquisition occurs is controlled by the MINC clock in the first three statements within the *for* (*i*=0; *i*<*npts*; ++*i*) loop in Figure 2.

Two averaging modes are utilized for improving the signal-to-noise ratio of weak signals in conventional or time-sequenced EPR spectra. The first mode, which is shown in Figure 2, acquires only one A/D conversion per magnetic field position but averages successive scans point for point, while the second mode (not shown) acquires and averages multiple A/D conversions for each magnetic field position in a single scan. The exact number of conversions performed in mode two is user selectable up to about 4000 conversions/s. Spectra may be averaged exclusively by either of these modes or by both modes simultaneously.

For time-sequenced EPR studies, two different modes are available to monitor the temporal decay profile of the EPR signal. In the first mode, the EPR signal intensity in the time domain is observed alternatively at two user-selectable spectral peaks, thereby providing a record of the peak-to-peak signal intensity as a function of time. To provide the necessary flexibility in the utility, the user may also select the delay time between EPR scanning of the two spectral peaks, the number of points to be acquired at each peak's field position, and the dwell time between acquisition of spectral points. Alternatively, the second mode simply acquires multiple conventional EPR spectra (i.e., with magnetic field scanning), but the acquisition of each spectrum is sequenced as a function of time with each spectrum then being appended to a single, large data file until the user-specified number of spectra has been obtained. In this second mode, the user may select the delay time between spectral acquisitions, and, after the completion of data acquisition, the EPR signal decay profile may be generated at any magnetic field value within the spectrum with the number of points in the profile being equal to the number of spectra recorded. Furthermore, in either of the time-sequenced modes, both averaging modes (discussed previously) are operational.

To acquire the time-resolved EPR spectra of short-lived species, the user begins by executing the setup (*su*) data acquisition mode, which continuously polls the Biomation for the reception of a trigger impulse. The availability of the *su* function allows the user to verify reliable triggering of the unit, optimize the trigger threshold, or find and maximize EPR signals to utilize the full dynamic range of the waveform



```
VersaTerm PRO - DEC VT100
                          CURRENT ESR PARAMETERS
                         Data File: /usr/group/jef/vitk1/jef101.dat
*****************************************************************************
* (1) Hmid: [illegible].00  Gauss  (2) Range: 20.00  Gauss (3) Gain:    2500.00     *
* (4) Power: 10.00    mW    (5) Nsweep:   128        (6) Npts:     512        *
* (7) Frequency: 9515.00 MHz (8) Dwell: 10.00  usec  (9) Ampmod:    5.00 Gauss *
* (10) KShift:   128        (11) Magnif.:   1      (12) Terminal Mode: [illegible]  *
* (13) NField:  40       (14) FldDwell: 2.50  s  (15) StaticFld: 3200.00Gauss  *
*****************************************************************************
*                          FAST KINETICS OPTIONS                              *
*****************************************************************************
* he) explain parameters & program operation   cs) view current spectrum     *
* fs) set field to a static value          mk) do tk at NField positions      *
* tk) trace kinetics (1 field position)   mb) do mk with auto bkgd. subtraction *
* su) set up mode to find signal          ma) do mb (new bkgds. acquired during)*
* fr) retrieve file              st) store file               qu) quit        *
*****************************************************************************

>
    Enter a parameter as follows:  parameter number=value.
```

Figure 3. Time-resolved EPR data acquisition menu, as observed by the user during execution of the programs, from which appropriate parameters or options may be specified.

recorder's A/D converter. Having obtained reproducible triggering and acceptable signal-to-noise ratio, the user may choose from several data acquisition modes, as shown in Figure 3, to perform the desired measurement. The basic acquisition mode is provided by the *tk* option, wherein a kinetic trace of the transient species is acquired from the Biomation after a laser flash. This trace is obtained at a specified magnetic field, which is set by using the *fs* option, and the user may specify the number of averages, *NSweep*, to be performed. The second mode, *mk*, is similar functionally to the *tk* mode, but multiple traces are obtained at discrete magnetic field values within the field range specified on the EPR console. The field is stepped to progressively larger values as previously described for conventional EPR acquisition, and the traces are appended in one data file until the user-specified number of them, *NField*, has been obtained. The remaining two modes, *mb* and *ma*, acquire on-resonance signal traces utilizing the *mk* type of acquisition but in addition provide capabilities for subtraction of superfluous signals (residuals) by obtaining an averaged trace of such residuals at a static, off-resonance magnetic field position. Residuals may arise, for example, from disturbances of the resonant microwave field distribution within the EPR cavity due to scattered light from the sidewalls of the cavity or changes in cavity Q due to the heating of the sample by laser flashes. A common residuals trace is subtracted from each composite signal trace (real and residuals signals) as they are obtained in the *mb* mode. However, in the *ma* mode a new residuals trace is obtained (off-resonance) for each field position and then subtracted from the composite signal trace during the scan (or a series of composite/residuals difference traces may be obtained at each field position and then averaged to yield a single difference trace). Any of the modes described above may be aborted by pressing a "q" (quit) on the keyboard.

All data acquisition modes for time-resolved studies utilize the flash( ) function shown in Figure 4. Keyboard interrupts are disabled prior to entering the function, while other processes are stopped upon execution of the first two lines of code in the function. With a real-time mode established, the system is polled for reception of a trigger impulse, and if it does not receive it within 30 s (code not shown), flash( ) is aborted because a failure in the data acquisition process is assumed. Alternatively, if a trigger is detected, the recorded data from the Biomation are transmitted in parallel to the MINC DI buffer via a handshaking protocol wherein the DI line repeatedly strobes the Biomation for data, waits to receive it, and then sums the data directly into a long integer array element [see *for* (*i*=0;*i*<*npts*;++*i*) loop within *for* (*kk*...) loop]. If more traces are to be acquired, the Biomation is rearmed [see first three statements within *for* (*kk*...) loop), and the handshaking protocol is repeated. After all traces have been acquired, the points within the long integer array are averaged and multiuser status is renewed by restarting the system clock and calling rtp(0). It should be noted that averaging multiple scans postacquisition permits faster program execution.

```
#define SYSCLK 0146546                /* address of SYStem CLocK */
#define DI 0140160                    /* address of Digital-In block of MINC */
#define DO 0140260                    /* address of Digital-Out block of MINC */
struct {
 int status,buffer;                   /* address of CSR and buffer of input device */
};

flash(n)
      /**** Retrieve and average data from Biomation after a laser FLASH ****/
int n;                                /* n: # of sweeps to average */
{
int i,kk;
long linten(1024);
 rtp(1);              /* kill Unix multi-user; only process is fast kin. data acq. */
 for (i=0; i<=100; ++i)  SYSCLK->status = 0;           /* turn system clock off */
 for (kk=1; kk<=n; ++kk) {            /** acquire and average n EPR sweeps **/
  DO->buffer = 00;                    /* sequence to arm biomation */
  DO->buffer = 01;
  DO->buffer = 00;
  while ((!(CLK->status & 0100200)));            /* test for an interrupt (trigger) */
  CLK->status = CLK->status & ~051;   /* clear clock status register */
  DO->buffer = DO->buffer | 02;       /* put biomation in output mode */
  for (i=0; i<npts; ++i) {            /* load biomation points into computer */
    DI->status = DI->status & ~0200;  /* computer is ready for a point */
    while(!(DI->status & 0200));      /* when true, biomation has sent point */
    linten(i) += DI->buffer;          /* computer reads from biomation */
  }                                   /* finished handshaking with biomation */
 }                                    /* all sweeps completed; end for (kk..) */
 for (i=0; i<npts; ++i)               /* average points to number of sweeps */
  inten(i) = (float)(linten(i))/( (float)(n) ) + 0.5;
 SYSCLK->status = 0100;               /* turn system clock on again */
 rtp(0);                              /* turn off real-time process */
}                                     /* end flash() */
```

Figure 4. Portion of C source code for time-resolved EPR data acquisition illustrating the interface of the Biomation waveform recorder. Keyboard interrupts are disabled and re-enabled before and after the routine according to the method in Figure 2, and in the same figure the address of the CLK structured pointer is defined. Data are summed in a long integer array, linten[ ], to avoid integer overflow.

Once data have been acquired, as either conventional, time-sequenced, or time-resolved spectra, the **esran** or **esrplt** programs may be invoked to analyze or graph the data, respectively. In **esran**, for example, a variety of spectral manipulations may be performed such as subtraction or addition of two spectra, smoothing, zooming, base-line correction by least-squares curve fitting algorithms, *p–p* intensity and line-width measurements, *g* factor determinations, or integration (single or double) by trapezoidal or Simpson's rule method. Integration may be done either manually, whereby the user may view the singly and the doubly integrated EPR spectra, or automatically, whereby only the average double-integral value over a range of six different base-line corrections is given along with the standard deviation. The **esrplt** program, however, is devoted to graphics applications. Spectra may be plotted in either in a 2-D or 3-D perspective, the 3-D plots being generated by rotation of coordinates through the Euler angles. From multispectral data files, 2-D plots may be made of individual spectra (i.e., conventional or kinetics EPR traces) or of a spectrum composed of points calculated from a specified region of points in each spectrum within the file (i.e., time-integrated EPR spectra). This specified region of points is user selectable, and the region may be either averaged or integrated according to the user's needs. The ability to generate such a spectrum from a time-resolved EPR data file allows the user to extract the EPR spectrum at a specific time from the multiple kinetic traces, and the ability to integrate a range of points centered about a desired time (within the time limits of the kinetics traces) allows the user to perform time-integrated spectroscopy (TIS) using only computer software on existing raw data.

PERFORMANCE AND LIMITATIONS

The most notable limitation associated with the time-resolved EPR interface is that the phys( ) function is machine dependent, which means that the memory mapping criteria must be reevaluated for a different computer. However, another approach for accessing peripheral devices on a different computer via software commands would be to write device drivers for the specific devices that are to be interfaced, modify the /dev file to include these new devices as part of the Unix kernel, and finally recompile the modified Unix kernel. The /dev file is used by Unix to locate peripheral devices by utilizing pointers that are comprised of a major device number (to represent the device) and a minor device number (to represent its address). Another limitation associated with the interface is that the rtp( ) function (or equivalent real-time sensitive kernels) is not generic to all Unix releases, so the interested parties should plan on checking that their prospective Unix release has capabilities similar to those described for rtp( ). Otherwise, they would have to write a function that memory-locks processes under Unix as does rtp( ). Although a real-time interface for data acquisition under Unix has been achieved through the use of rtp( ) and hardware interrupt control (see Unix and C), it is presently difficult to evaluate the ultimate speed of data acquisition in the real-time mode because of communication limitations (see discussion below). However, the data collection speed of the Unix real-time mode (written entirely in C) even in its present configuration is ca. 75% as fast as that for the dedicated microprocessor implementation of McLauchlan and Stevens[13] (data acquisition code written in Assembler language), the comparison being made in terms of kilobytes per second of data recovered.

As assessed by speed, flexibility, and capabilities, the performance of the modified version of 2.9 BSD Unix operating system for the PDP-11/23A is outstanding. For example, the previously discussed peak data acquisition speed of ca. 15 traces/s that is obtained by using Unix more than matches our present fastest laser flash repetition rate. Moreover, the computer data acquisition rate is limited by the speed of data transmission by the Biomation, so the operating system could in theory provide even faster data recovery. This fact is important because the ultimate speed of the time-resolved EPR spectrometer has not yet been achieved. With the fastest output currently available (i.e., J1206 at ca. 600-ns rise time), the speed of the Biomation is adequate (three points on the rising signal) for recording all transient signals from the present spectrometer design. However, the inherent rise time of the spectrometer is ca. 100 ns (for direct detection prior to the microwave bridge preamplifier) or faster (if cavity Q is spoiled). Consequently, with ongoing developments that include modifications to the microwave bridge to recover faster signals and the computer interface to a boxcar averager (as fast as 2 ns/point) to record such signals, the time-resolved EPR spectrometer will require the additional speed that Unix can provide.

Unix also provides flexibility, which is shown by the previous discussion of its configuration for our minicomputer. This flexibility allows Unix to run on faster, larger memory computers, which means that the software that has previously been developed on the PDP-11/23A can execute on such a computer; this is consistent with our plans to use a host computer in the future for networking several dedicated, data acquisition computers.

The outstanding performance of the modified Unix operating system may also be appraised by the many capabilities of the system, in terms of both data acquisition and analysis applications. For example, as previously shown a dedicated CPU may be created within the Unix environment for real-time applications, but upon the execution of a few software commands (see Program Features) the normal multiuser, multitasking operations of Unix are reestablished. The result of this capability is that programs for data analysis or conventional EPR data acquisition may be executed simultaneously because neither one is time sensitive. Moreover, since time-sensitive data acquisition programs [i.e., using rtp(1)] do not require the real-time mode continuously, analysis programs are executable even under these conditions. In addition, file-handling capabilities, which are not easily coded in standard operating systems such as RT-11, are an integrated part of the Unix operating system, thereby allowing the multiple spectra derived from data collection programs to be saved in a single large data file and subsequently retrieved in random or sequential order. Furthermore, these capabilities facilitate the transfer of data files to the Macintosh II (a smart terminal) for subsequent printing on the LaserWriter Plus, thereby permitting the production of publications, such as this paper, by completely electronic means.

The performance of the software as measured by ease of modification of existing code and debugging abilities is outstanding primarily because of the fact that all programs are written in C. With just one language, the programs also become more portable to another computer than they would be if part of the software (e.g., data acquisition) was written in a different language, such as Assembler. In addition, the modular organization of C programs has permitted the flash( ) function to be utilized without changes for both the time-resolved EPR program and a program that controls data collection from an optical flash photolysis spectrophotometer. Moreover, the organization of the C functions into separate files based on similarity of task permits modifications of a general class of functions (e.g., data acquisition functions) to be made faster because only the files that are changed must be recompiled into object code files, the unchanged files requiring only relinking to prepare the executable module. All compiling and linking operations are automated via the Unix *make* utility.

The overall capabilities of the time-resolved EPR spectrometer interface and analysis programs are demonstrated by the acquisition of room temperature transient EPR kinetic traces of the P700$^+$ cation in TSF-1 photosystem 1 particles (3.71 mg of Chl/mL), the cation being formed by repetitive actinic flashes (27 mJ/flash, $\lambda = 532$ nm) from the Nd/YAG

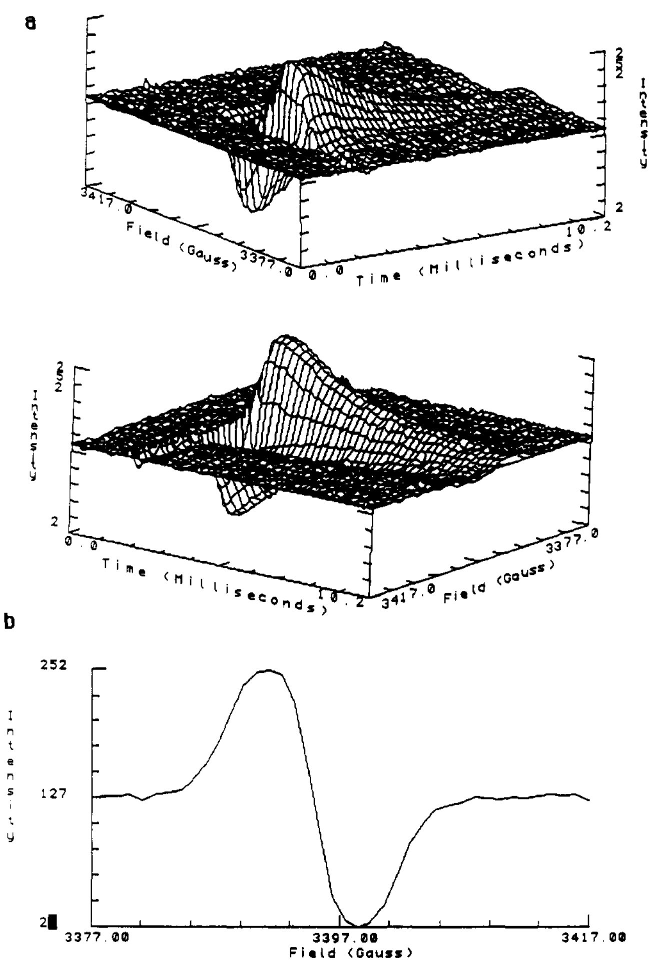


Figure 5. Time-resolved EPR measurements of P700$^+$ in a TSF-1 preparation of photosystem 1 particles from spinach leaves. Measurements were obtained with EPR parameters as follows: frequency 9.512 GHz; microwave power 20.0 mW; amplitude modulation 0.5 mT; modulation frequency 100 kHz; gain 6300. The measurements show (a) two three-dimensional views of the EPR kinetic decay traces of P700$^+$ at progressively increasing magnetic field values and (b) time-resolved EPR spectrum of P700$^+$ generated from points obtained by specifying an integration window of 100 μs in each trace in (a) centered at 350 μs after the flash.

laser. These experiments are currently being performed to further probe the role of vitamin K_1 in the initial photochemical processes of photosystem 1.[26] Two 3-D profiles of the kinetics traces are displayed in Figure 5a, the observed traces arising from the back-reaction of the electron from $F_{A/B}$ to P700$^+$. As shown in Figure 5b, the transient EPR spectrum of P700$^+$, generated by integration of a 100-μs window centered at 350 μs in each kinetic trace, exhibits a 0.72-mT line width for P700$^+$, which is equivalent to that obtained by steady-state, conventional (field-swept) EPR measurements. This cross-sectional spectrum is generated by the **esrplt** program, and certain spectral analyses, such as double-integration calculations, may be evaluated by the **esran** program. Future studies will focus on such analyses for the comparison of control and vitamin K_1 depleted TSF-1 particles.

CONCLUSIONS

We have described a novel implementation of Unix on a computer with limited memory, and we have further shown that time-resolved EPR spectroscopy can be performed in such an environment. In addition, with the results obtained from the initial time-resolved EPR studies of P700$^+$, we have illustrated that data acquisition programs written entirely in C and operating under 2.9 BSD Unix can execute with the speed required by a time-resolved EPR spectrometer. Furthermore, Unix and C have been shown to provide numerous immediate advantages, including multiuser, multitasking operations, efficient file manipulation capabilities, and ease of software development. Moreover, the time-resolved EPR software not only serves our present experimental needs but also remains portable with minor modifications to another computer running Unix. Therefore, our ultimate plans for networking several dedicated computers to a central computer workstation operating under Unix, such as a SUN workstation or Micro-Vax, can be realized with minimal difficulties. This point emphasizes a philosophy of building current data acquisition and analysis facilities with the future in mind. The source code described herein is available from the authors upon request.

ACKNOWLEDGMENT

We thank Ted Weidner and Larry Feeser for their assistance in the configuration of Unix in addition to Norwin Malmberg and Roger Fuji for their assistance in developing the software interface. This work was supported in part by NIH Grant 5-R01-26133.

REFERENCES AND NOTES

(1) For a review of the principles and experimental techniques of time-resolved EPR spectroscopy see: McLauchlan, K. A.; Stevens, D. G. Flash Photolysis Electron Spin Resonance. *Acc. Chem. Res.* **1988**, *21*, 54–59.
(2) Carlier, M.; Devolder, P.; Pouwels, J. F.; Sochet, L. R. A Microcomputer-Based Data Acquisition and Analysis System. Application to EPR. *Comput. Chem.* **1987**, *11*(1), 25–28.
(3) O'Connor, Sally E.; Spraggins, Thomas A.; Grisham, Charles M. Interfacing a Microcomputer with the Varian E-Line Series EPR Spectrometer. *Comput. Chem.* **1981**, *5*(4), 181–184.
(4) Lipscomb, John D.; Salo, Rodney W. Electron Paramagnetic Resonance Spectrometer Data Accumulation and Reduction System for Microcomputers. *Comput. Enhanced Spectrosc.* **1983**, *1*(1), 11–15.
(5) Rich, Edwin S.; Wampler, John E. EPR Spectrometer Control and Data Acquisition: Using a General-Purpose IEEE-488 Bus Interface and Structured Software. *Am. Lab.* **1982**, Sept, 17–21, 23–24, 26, 28.
(6) Trousson, Patrick; Rinne, Michel. Microcomputer-based system for on-line EPR data accumulation. *Rev. Sci. Instrum.* **1984**, *55*, 1989–1994.
(7) Schultz, R.; Hurst, G.; Thieret, T. E.; Kreilick, R. W. An EPR Data System Based on S-100 Bus Home Computer Components. *J. Magn. Reson.* **1983**, *53*, 303–312.
(8) Maple, S. R.; Hensel, J. P. Computer-Assisted Electron Paramagnetic Resonance Spectroscopy. *J. Magn. Reson.* **1986**, *66*, 445–455.
(9) Joela, Heikki; Salo, Esa. Micro Computerized EPR. *Acta Chem. Scand. B* **1985**, *39*(2), 131–139.
(10) Momo, F.; Ranieri, G. A.; Sotgiu, A. Microprocessor Driven Electron Spin Resonance (ESR) Spectrometer. *Comput. Enhanced Spectrosc.* **1983**, *1*(2), 79–82.
(11) Goldberg, Ira B.; Crowe, Harry R.; Carpenter II, Richard S. Computer Control of Electron Spin Resonance Spectroscopy. *J. Magn. Reson.* **1975**, *18*, 84–96.
(12) Basu, S.; McLauchlan, K. A.; Sealy, G. R. A novel time-resolved electron spin resonance spectrometer. *J. Phys. E.* **1983**, *16*, 767–773.
(13) McLauchlan, K. A.; Stevens, D. G. Field-time two-dimensional transient electron spin resonance spectroscopy, MISTI methods extended to complete spectra, and a comparison of existing time-resolved methods. *Mol. Phys.* **1986**, *57*(2), 223–239.
(14) Bartels, D. M.; Lawler, R. G.; Trifunac, A. D. Electron T_1 measurements in short-lived free radicals by dynamic polarization recovery. *J. Chem. Phys.* **1985**, *83*, 2686–2707.
(15) Suehiro, Tadashi; Masuda, Seiichi; Nakausa, Ryuichi; Taguchi, Masamichi; Mori, Atsushi; Koike, Akemi; Date, Munehiro. Decay Reactions of Aryldiazenyl Radicals in Solution. *Bull. Chem. Soc. Jpn.* **1987**, *60*, 3321–3330.
(16) Yang, Paul W.; Moffatt, Douglas J.; Mantsch, Henry H. A Unix Operating System Based Computer Workstation for Spectroscopic Data Manipulation. *Comput. Enhanced Spectrosc.* **1986**, *3*(2), 63–67.
(17) White, Robert L. Advantages of UNIX Based Laboratory Data Processing. *Comput. Enhanced Spectrosc.* **1983**, *1*(4), 185–189.
(18) Aseo, Joe. UNIX Tweaked for Real-Time Applications. *ESO: THE Electronic System Design Magazine* **1987**, June, 60.
(19) Cramer, Bill. Writing Real-Time Programs Under Unix. *Dr. Dobb's J.* **1988**, June, 18–20, 24, 26, 28, 33.
(20) Levy, George C. Heirarchical Computer Networks for the Spectroscopy Laboratory. *Spectroscopy* **1988**, *3*(1), 30–36.
(21) *Berkeley Unix Programmer's Manual: PDP-11 Version 2.9*, 7th ed. (Second Berkeley Software Distribution); University of California at Berkeley: Berkeley, CA, 1983.

(22) Norris, J. R.; Warden, J. T. A Rapid Response 100 kHz ESR Spectrometer. *EPR Lett.* **1980**, *1*, 2-3.
(23) PHYS(2) in *Berkeley Unix Programmer's Manual: PDP-11 Version 2.9*, 7th ed. (Second Berkeley Software Distribution); University of California at Berkeley: Berkeley, CA, 1983; Vol. 2.
(24) Ferrin, Thomas. RTP(2) in *Berkeley Unix Programmer's Manual: PDP-11 Version 2.9*, 7th ed. (Second Berkeley Software Distribution); University of California at Berkeley: Berkeley, CA, 1983; Vol. 2.
(25) LD(1) in *Berkeley Unix Programmer's Manual: PDP-11 Version 2.9*, 7th ed. (Second Berkeley Software Distribution); University of California at Berkeley: Berkeley, CA, 1983; Vol. 1.
(26) Palace, Gerard; Franke, James E.; Warden, Joseph T. Is phylloquinone an obligate electron carrier in photosystem I? *FEBS Lett.* **1987**, *215*, 58-62.

CHEMLEARN: Microcomputer-Based Training for CHEMLINE, an Alternative to Formal Classroom Training[†]

MELVIN L. SPANN* and MIRIAM L. PERKINS
National Library of Medicine, Bethesda, Maryland 20894

Received September 8, 1989

CHEMLINE (CHEMical Dictionary OnLINE), the National Library of Medicine's online chemical dictionary file, is primarily used by librarians and information scientists to enhance the retrieval of bibliographic information associated with chemical substances. This paper will discuss CHEMLEARN, a microcomputer-based training program designed to provide inexpensive, easily accessible self-instruction as an alternative to formal classroom training in the use of CHEMLINE. CHEMLEARN allows novice users to learn at their own pace and with considerable program feedback. In addition, it provides the skilled searcher with a way to reinforce or recall previously learned search techniques without incurring online charges. CHEMLEARN may be used in place of formal training or as a precursor to or a refresher following formal training.

INTRODUCTION

CHEMLINE is the National Library of Medicine's (NLM) online, interactive chemical dictionary file created by NLM's Toxicology Information Program in conjunction with Chemical Abstracts Service (CAS).[1] It serves as an adjunct file for the search and retrieval of bibliographic and factual information pertaining to chemical substances. In fact, CHEMLINE can be viewed as the focal point for chemical searching of the Library's online retrieval services. The file provides a mechanism whereby information on over 900 000 chemical substances can be searched and retrieved online. It contains CAS Registry Numbers, molecular formulas, CAS chemical index nomenclature, generic and trivial names, classification codes, and a locator designation that points to other files in the NLM system and the Toxic Substances Control Act (TSCA) Inventory. In addition, where applicable, each record contains ring information including number of component rings within a ring system, ring sizes, ring elemental compositions, and component line formulas. This permits the searcher to refine and amplify chemical search terms in searching the selected databases.

The majority of CHEMLINE users are search intermediaries, librarians or information scientists conducting searches for scientists or health professionals.[2] NLM has been training search analysts since the implementation of its online computer system, MEDLARS (Medical Literature Analysis and Retrieval System), in the mid 1960s.[3] Currently, NLM offers two formal training courses geared toward the search intermediary which include segments on CHEMLINE. They are a 3-5-day modular introductory course titled "The Fundamentals of MEDLARS Searching" and a week-long modular sequel course titled "The Follow-Up to the Fundamentals of MEDLARS Searching". As useful as these courses are, they have the drawback of requiring the search analyst to travel to a training site for a scheduled course which may be offered in their locale only a few times per year and which can be both costly and time-consuming. In addition, these courses do not permit the degree of self-paced instruction that some learners desire or need, especially considering the highly technical nature of CHEMLINE. Furthermore, due to recent increases in the number of users, the capacity to train these individuals in a timely manner is being fully taxed.

A main-frame computer-assisted instruction program for CHEMLINE was developed at the National Library of Medicine in 1979[4] to meet the needs of searchers who wanted to learn basic concepts involved in searching CHEMLINE in the privacy and convenience of their own offices. It was designed to be accessed through the then predominant hard-copy terminal connected via modem to the main-frame computer at NLM. With the increased availability of microcomputers, it became clear that yet another method of instructing searchers in the use of CHEMLINE would be perhaps more practical. Microcomputer-based training programs provide economical, convenient self-instruction for database searchers and allow for use of enhanced visual and graphic capabilities in conveying technical information. Therefore, in the spring of 1988 NLM released CHEMLEARN, a microcomputer-based training program, to address the needs of CHEMLINE search analysts who want to learn or continue learning at their own pace, on their own equipment, in their home or office, and at an economical rate.[5] After over a year of user feedback, CHEMLEARN, version 2.0, was released incorporating many new capabilities and techniques.

A technical database such as CHEMLINE is only worthwhile if the audiences for whom it is intended are using it. As with any technology or tool, some training is necessary for searchers to use it effectively. NLM has developed a variety of learning methodologies including formal classes, printed manuals, and computer-based training to address this need. One of the few well-established principles of educational psychology is that people learn via a variety of styles, rates, and sequences. Whereas some people learn most efficiently in a group-paced classroom environment, others benefit more from a self-paced, highly structured computer-based environment. At the same time, temporal and financial logistics may prohibit some potential users from availing themselves of in-person classroom instruction, and, therefore, independent

[†] Presented at the 198th National Meeting of the American Chemical Society, Miami Beach, FL, Sept 1989, paper CINF 30.