An operating subsystem for continuous monitoring studies

HOWARD L. KAPLAN

Addiction Research Foundation, Toronto, Ontario M5S 2S1, Canada

A set of programs has been developed to allow efficient operator control over the recording of physiological and behavioral measures in experiments of several hours' duration. Important features include menu-driven selection among possible measurements, minimal data loss in the event of system failure, on-line correction or deletion of incorrect data both during and after collection, and selective reporting of individual statistics and time periods. These programs can be extended to include new measurements with minimal interference with existing measurements.

A standard research paradigm at the Addiction Research Foundation of Ontario includes monitoring a subject's physiological and behavioral responses to drugs over the course of several hours. These responses may include continuously sampled measures such as heart rate, discontinuously sampled measures such as blood pressure, and inherently discontinuous measures such as memory and pursuit-tracking tests. Depending on the drugs under study, different subsets of these measures will be employed at different times within the experiment.

To manage the program development work associated with servicing these different experiments, we are fitting as many as possible into a framework called the "continuous monitoring system" (CMS). CMS provides an overall scheduling system, operator communication facilities, data storage facilities, and postexperimental graphic output, applicable to a wide range of studies. When novel measures are required for a study, they can be developed in independent program modules and added to the system with little or no impact on existing measurement modules. Once an operator has learned to use the system for one study, only the names of the new statistics need be added to the operator's command repertoire in order to perform the computer-control portions of additional studies. Improvements made in the overall framework generally do not require the measurement modules to be changed, so that if an older study is reimplemented, these improvements are automatically incorporated into the older experiment. In summary, an experiment becomes a menu-driven selection among an easily expanded set of possible simultaneous measures, in which the rules for preventing the measurements' interfering with each other have already been established and tested.

THE LABORATORY ENVIRONMENT

The CMS system was developed for use in the Human Responses Laboratory of the Addiction Research

Foundation of Ontario. The laboratory computer is a Sperry-Univac V76 with 128K words of 16-bit memory and 20 MB of on-line disk storage, running under a slightly modified version of the VORTEX II real-time operating system. This operating system allows concurrent execution of several real-time programs at user-specified priority levels; CMS applications, for example, always require at least three such concurrent programs executing. The peripheral devices include a nine-track tape drive, an electrostatic printer/plotter, a 16-channel analog input multiplexer feeding a 13-bit A/D converter, 4 channels of D/A output, and a DECwriter.

The laboratory is a shared facility, used for experiments that are generally designed by investigators, implemented by laboratory staff, and conducted by research assistants or registered nurses. These roles tend to overlap much less than in the typical university environment, in which it is quite common for one person to design, implement, and conduct a study. This separation of roles among different individuals was a major factor in the design of CMS. In other circumstances, a person with substantial computer experience might be competent to make data corrections using an octal debugging tool. In our environment, such corrections would need to be made by someone having little computer background, using a custom-designed, friendly, data correction tool. System operations, including uncommon ones such as emergency restarts, must be performable in the absence of the programmer who implemented the experiments. In other words, some features that might be considered conveniences in other laboratories are necessities in our laboratory.

AN EARLIER ATTEMPT AT MONITORING

Several studies involving the continuous monitoring of responses had been implemented in the laboratory before this system was developed. The immediate predecessor was a study of responses to the drugs disulfiram and calcium carbamide, which induce

unpleasant reactions if ethanol (beverage alcohol) is subsequently ingested. Four measures were sampled continuously and reported once per minute: heart rate, skin temperature, rectal temperature, and skin flushing. Sampling occurred during the first 50 sec of each minute, and during the last 10 sec the minute's averages were computed and stored in a memory array. Each minute's data were then printed during a 25-sec block of analog input sampling that began the next minute's data collection. At the conclusion of the experiment, the entire data array was reprinted, the array was saved on disk, and graphs were produced. After the experiment, the disk copy of the data array, including any errors, was saved on tape.

A single FORTRAN program managed the input of digitized data, the computation of meaningful measures from that input, and the minute-by-minute printouts. There was a strict sequencing of events, with computations restricted to a small window at the end of each minute, and printing restricted to a somewhat larger but still fixed window during the subsequent minute. Any delays in the printing would prevent the program's completing the next minute's data sampling at the appropriate time. In the event of printer failure, requiring the experiment to be run from a CRT terminal without hard copy, there was no way to inspect data that had disappeared from the screen, being more than 24 min old. The printer (a DECwriter) was operating for only 2.5 sec/min, which was normally reasonable because there was no other immediate use for it. However, if some kind of hard-copy output was needed for another task, there was no way to interrupt the printout of the experiment, use the DECwriter for another purpose, and then catch up with the printing backlog. The single worst feature of the program design was that all of the data, several hours' worth, was kept in volatile semiconductor storage until the experiment concluded. Equipment breakdowns or power failures, which were rather frequent at that time, caused the loss of an entire session's data, unless the data were laboriously reentered by hand. Even if the computer could be restarted, meaning a loss of only 5 min of data, there was no convenient way to integrate the data from the restarted session, which claimed to begin at Minute 1, with the manually reentered data that actually did begin at Minute 1. By the time the experiment was completed, we had spent about as much time in developing special-purpose repair programs for manually correcting the machine-readable data records as we had in implementing the original data collection and statistical computations.

From these difficulties we developed design objectives for the next study. Originally designed to study responses to codeine in the presence of the narcotic antagonist naloxone, the programs were subsequently generalized and became the current CMS. Although we never explicitly listed the objectives at the time, they formed the constraints under which CMS was eventually developed: (1) Separate the timing requirements of data

input, statistical reduction of the data, and data reporting. (2) Allow both continuous measures, which can be reported once per minute, and discontinuous measures, occurring only in occasional minutes, to share peripheral devices (particularly the A/D converter) and data storage facilities. (3) Minimize the loss of data in the event of system failure, by dumping each minute's summary data to disk as soon as they have been computed. Ensure that failure to properly close the output file in the event of system failure does not prevent future data access. (4) Allow an interrupted experiment to be restarted at any minute, with the data between the interruption and restart points automatically recorded with a missing value code. (5) Provide an operator interface allowing for selectively starting or stopping collection of any measure at any time. (6) Integrate facilities for data correction into the system for data collection, so that the operators can delete or replace incorrect data before they are stored on magnetic tape. (7) Provide operator control over the level of data reporting, so that printouts can be limited to specified minutes or statistics, or to statistics that exceed operator-modifiable limits of reasonableness. (8) Provide a mechanism whereby measurement modules can indicate that the data they are receiving are internally inconsistent as a missing value code distinct from data that are simply not collected. (9) Allow printing to be inhibited indefinitely while data collection continues, with the backlog of unprinted data available upon demand. (10) Provide for the integration of manually collected data (rating scales, manually read instruments, etc.) into the same data records as the automatically collected data. (11) Provide a general-purpose graphics facility that produces a record of all measures over the time course of the experiment. (12) Make the programs sufficiently general that new experiments can be implemented by simply adding new measurement modules to the existing framework, with operator control over which modules will be employed in any given study.

THE EXPERIMENTER'S VIEWPOINT

For experimenters whose proposed studies fit into the CMS framework, CMS offers relatively short lead times and a pretested system for the management and reporting of data. The cost associated with these benefits is that there are some important limitations on the kind of measure suitable for CMS implementation. The two most important limitations are in the area of bandwidth, or sampling rate. Both the rate at which the raw data are digitized and the rate at which the summaries of the digitized data are reported are fixed by the system, not alterable for individual measures.

Under CMS, 10 analog channels are digitized 50 times/sec each, in strict rotation. In general, any statistic whose computation requires a higher sampling rate cannot be implemented. Among the measures

typically collected in our laboratory, this rules out wide-band EEG recording (but would allow monitoring an analog-filtered signal for 10-Hz alpha rhythm only); it also rules out any other signal exceeding 25 Hz. Under special circumstances, this limit can be stretched somewhat by combining the input of two channels at antipodal positions in the sequence: For example, alternate readings from Channels 2 and 7 would provide an aggregate bandwidth suitable for 50-Hz signals. The 25-Hz limit is more than adequate for averaging signals that are normally read manually from a panel meter, and it is also comparable to the bandwidth of a good chart recorder. It is also (but just barely) adequate for digitizing three-lead ECG signals, for digital processing to extract the heart rate and some signal-shape parameters.

The other bandwidth limitation is that statistics are normally reported only once per minute. Any parameter that changes rapidly within 1 min would not be reported quickly enough. If we wanted to know the exact point within the minute that the heart rate began accelerating, we could not extract that information by reading the usual heart rate printout: The grain is just too coarse. As in the antipodal doubling above, we could make some improvement by creating separate statistics for the heart rate in the first and second half of each minute and, later, interlacing the two for a somewhat better grain. In general, though, we restrict the study to measures having summary values that can be reported once per minute with no loss of important information. This corresponds well to the time course of the events in which we are usually interested.

Some of the measurements implemented under CMS include heart rate, blood pressure, postural stability, hand tremor, and pursuit tracking. Heart rate is obtained from digital processing of the raw ECG signal, for which the sampling rate is barely adequate. Blood pressure is a semiautomated measurement, with the computer controlling and monitoring the cuff pressure and with a laboratory assistant pressing a button to indicate the start and stop of the Korticoff sounds. Postural stability is measured by having a subject stand on a force platform with a load cell under each corner; the movement of the center of force is converted to x-y coordinates, and the length of the path so traced over 30 sec is used as a measure. As hand tremor has no significant components above 10 Hz, the 25-Hz bandwidth of CMS is adequate for digitally squaring individual readings from an accelerometer probe, for a summary measure of tremor energy. For pursuit tracking, a road-like display is constructed from a 100-point window that progresses through a randomly modulated sinusoid and from a small set of coordinates depicting the top view of an airplane. This display is refreshed 33 times/sec, and the airplane, whose horizontal velocity is controlled by a joystick, appears to be flying over the road. Both the root mean square distance from the center of the road and the percent time over it are used as measures.

Additional measures such as GSR or skin temperature, which are simply rescaled means, medians, or extreme values of an analog input signal over the minute, can be added to the system with about 5 min of work and a simple recompilation of one small program module. Addition of other measures may require considerably more work. Some of the details are described in "The Programmer's Viewpoint" section, below. What is important from the experimenter's viewpoint is that the development time is usually comparable to the time to get final approval and recruit subjects, so that the programming effort is not the limiting factor in scheduling the study.

THE OPERATOR'S VIEWPOINT

In recent years, much has been written about tools with which a programmer can communicate with a computer in the design and testing of an experiment, before it happens. Much has also been written about statistical tools with which the experimenter can communicate with the computer for the analysis and presentation of an experiment, after it has already happened. There seems to be very little attention to tools with which the operator, often a research assistant with little computer expertise, can communicate with the computer while the experiment is happening. Just as package programs such as SAS and SPSS can provide general-purpose interfaces through which nonprogrammers can implement a menu of procedures for arbitrary variables, the CMS package provides a menu of important experiment-control features through which nonprogrammers can instruct the computer how to modify the conduct of an ongoing experiment.

Most CMS operations refer to statistics collected by the system. Each statistic has a name of up to eight characters; typical statistic names are HEARTRAT, SYSTOLIC, DIASTOLC, and ROMBERG (the name of the person who originated the systematic study of postural sway). Any data value computed by a CMS module is stored by statistic name in the minute in which it is collected; a missing value code is the default, stored for any statistic for which no other value is supplied. Any CMS session can manage up to 70 such statistics, 30 of which are recorded as floating-point values and the remaining 40 as integers.

A CMS session can be operating in one of three modes. In a test mode, all statistics computed by the system are printed on the DECwriter as they are available, but they are not stored on disk. This mode is generally used at the beginning of a day to test electrode placement, amplifier settings, and so on. Baseline mode allows the collection of up to 100 min of data, and postdrug mode allows the collection of up to 725 min of data. In both cases, the data are stored on disk in addition to being printed. Baseline data generally precede the first daily administration of the drug of interest, and their

exact timing relationship to the drug administration is not critical. Resetting the experimental clock to zero at the time of drug administration allows for a variable delay after the end of baseline measurement. That variability corresponds well to the realities of drug experimentation.

In the sample dialogues below, "C:" indicates a statement typed by the computer and "O:" indicates one typed by the operator.

The operator begins a CMS session by scheduling Program CM1, using the only necessary CMS command entered in "computerese" rather than in a more comfortable, dialogue-oriented language. In response, the time and date are printed so that the operator can, if necessary, correct them and reschedule CM1. The operator then specifies the mode in which CMS is to run:

```
O: ;SCHED,CM1,30,FL,F
C: CONTINUOUS MONITORING SYSTEM OPERATIONAL
  AT 8:23 ON 06/13/81
  TEST, BASELINE, POSTDRUG, OR RESTART?
O: BASELINE
```

At this point, the computer verifies that baseline mode is not being repeated on a day on which testing has already occurred and that postdrug mode is being requested only on a day on which baseline has already occurred, unless specifically overridden by the operator. Then a second program, CM4, is automatically scheduled by CM1. Almost all subsequent dialogues with the operator are managed by CM4. The standard prompt from CM4 is the word "YES?"; the possible operator replies to "YES?" are discussed below. CM4 starts by printing a brief status report, after which the operator typically requests the initial statistics to be collected in that session:

```
C: THERE ARE NO STATISTICS ACTIVE OR QUEUED NOW
  BASELINE MODE
  YES?
O: ON HEARTRAT, ROMBERG
```

The operator then temporarily stops communicating with CM4 and starts data collection:

```
C: YES?
O: RELEASE
C: THE TIME IS NOW 8:25:42
  PROGRAM WILL START AT THE NEXT EXACT MINUTE
  AFTER "GO"
  READY? TYPE "GO"
O: GO
```

At the next exact minute on the computer's internal clock, data collection begins and an external digital clock begins to display elapsed time to the nearest 1 sec.

Unless specified otherwise, the computer will print statistics computed during each minute 21 sec after the

minute ends. Only those statistics for which data have been collected or for which the collection program rejects the data (for reasons such as too large a coefficient of variation within the minute, rendering the data suspect) will be printed. Floating-point statistics are printed with five decimal places if their absolute values are less than 1.0 and with two decimal places otherwise. All statistics are printed by name, and any statistic that is rejected or outside its usual range is printed with an asterisk and an audible bell code:

```
C: STATISTICS AS OF 0:08
  HEARTRAT= 62.14 ROMBERG = 24.07
  STATISTICS AS OF 0:09
  HEARTRAT=REJECT!*
  STATISTICS AS OF 0:10
  HEARTRAT= 37.13* ROMBERG = 22.16
```

Using the commands to be discussed below, this printout can be limited to out-of-range statistics only or inhibited altogether.

At any time during the session, the operator can press a front-panel pushbutton on the computer to resume communication with Program CM4. This communication proceeds without interfering with ongoing data collection and recording, unless the operator specifically gives a command to initiate or terminate collection of one or more statistics. A few of the possible CM4 commands are of use only to a programmer in designing or diagnosing experiments; the rest will be briefly discussed in the following paragraphs.

```
MENU: print all commands and statistics
O: MENU
C: THESE ARE THE POSSIBLE DIALOGUE COMMANDS:
  SHOW      MENU      STATUS      ON
  OFF        VOLTAGES  REPRINT    MODIFY
  DELETE     SCAN      ENTER      SWAPTERM
  RELEASE    INHIBIT    WARNONLY   EXHIBIT
  NEWRANGE   STOP
  THESE ARE THE POSSIBLE STATISTIC NAMES:
  (V)HEARTRAT  (V)SYSTOLIC  →DIASTOLC
  (V)ROMBERG   (V)GSR        (M)ORALTEMP
  (1)FEELGOOD  (1)FEELSICK   (1)FEELROTN
```

In the list of statistic names, statistics preceded by "(V)" are automatically collected by the V76 computer, those preceded by "→" are implied by the previous "(V)" statistic and automatically collected with the preceding statistic, those preceded by "(M)" are manually collected when specified by the operator by name, and those preceded by a number are on a list of name-prompted statistics requested from the operator as a group, such as a battery of rating scales. These parenthesized numbers are microcoded by the binary bits of the list number; that is, "(3)" means that the statistic is on Lists 1 and 2. The automatically collected statistics can be turned "ON" or "OFF"; the others can be entered into the data records only by the operator, using the "MODIFY" or "ENTER" commands.

SHOW: produce a report of current CMS activity

O: SHOW

C: HEARTRAT ROMBERG +SYSTOLIC +DIASTOLC -GSR
 PROGRAM CM012 AWAITING ROMBERG
 MINUTE 0 WAS AT 8:26:00,
 TOTAL TIME ELAPSED = 0:17:14 AT 8:38:14
 BASELINE MODE
 SYSTEM NOW SCHEDULED TO STOP
 AT 1:15:00 ELAPSED

In the first line, statistics preceded by no sign are being collected, those preceded by a "+" sign will start to be collected for the next minute, and those preceded by a "-" sign will stop being collected when this minute expires. Program CM012, mentioned in the second line, collects some of the discontinuously sampled measures, such as blood pressure, pupil diameter, and postural stability (Romberg platform).

ON: start collecting one or more statistics

OFF: stop collecting one or more statistics

O: ON ROMBERG, GSR

C: +ROMBERG ALREADY

+GSR

YES?

O: OFF HEARTRAT-ROMBERG

C: HEARTRAT ALREADY OFF

-SYSTOLIC

+ROMBERG CANCELLED

Each "ON" or "OFF" command can specify one or more statistics, separated by blanks or commas, or ranges of statistics (using the ordering of the "MENU" listing), separated by a hyphen. Commands to turn statistics on or off do not take effect immediately; before a session, they are deferred until the session starts, and during a session, they are deferred to 30 sec after the minute. In the preceding example, the command to turn on ROMBERG (which was already scheduled to begin) was redundant, but the command to turn on GSR was accepted. The command to turn off HEARTRAT (which was then not active) was unnecessary, the command to turn off SYSTOLIC (implying also DIASTOLC) was accepted, and the command to turn off ROMBERG canceled the request to turn it on.

REPRINT: reprint the data from selected minutes

O: REPRINT 8, 0:12-0:14

C: STATISTICS AS OF 0:08

HEARTRAT= 62.14 ROMBERG = 24.07

STATISTICS AS OF 0:12

HEARTRAT= 62.01

STATISTICS AS OF 0:13

HEARTRAT= 61.13 ROMBERG = 22.16

STATISTICS AS OF 0:14

HEARTRAT= 61.05 SYSTOLIC= 117.05

DIASTOLC= 84.26

The list of minutes may include individual minutes and ranges. Times under 1 h may be specified by minute only; times over 1 h must be specified by hour and minute, rather than by some number of minutes over 60. This restriction corresponds to the way that time is displayed on our laboratory digital display clocks. If the "WARNONLY" command, below, has been issued, then the data reprinted are limited to out-of-range and rejected statistics.

INHIBIT: cease automatic minute-by-minute printouts

EXHIBIT: resume such printouts

WARNONLY: limit such printouts to extreme values

O: WARNONLY

C: YES?

O: RELEASE

[several minutes may elapse]

OUT-OF-RANGE STATISTICS FOR MINUTE 0:46

HEARTRAT=46.13*

Once the "WARNONLY" command has been given, both reprint requests and automatic printouts are limited to data that exceed specified warning ranges. These ranges are set by CM4 when it begins to execute, but they may be modified by the operator at any time. The "WARNONLY" command is generally given after about 15 min of data collection, when there is no need to monitor each minute's data to verify proper conduct of the experiment. The "INHIBIT" command suspends all automatic printing and is generally used only to free the DECwriter for another, temporary use. After that use, the data backlog is printed, when the operator specifies either "EXHIBIT" or "WARNONLY."

NEWRANGE: inspect and modify statistic warning ranges

O: NEW RANGE

C: WHICH STATISTIC?

O: SYSTOLIC

C: CURRENT RANGE FOR SYSTOLIC IS 90.00 TO 130.00

ENTER NEW MINIMUM, OR "NOCHANGE"

[ON EITHER EXTREME]

O: 80

C: ENTER NEW MAXIMUM

O: 120

C: CURRENT RANGE FOR SYSTOLIC IS 80.00 TO 120.00

WHICH STATISTIC?

O: DIASTOLC

C: CURRENT RANGE FOR DIASTOLC IS 60.00 TO 100.00

ENTER NEW MINIMUM, OR "NOCHANGE"

[ON EITHER EXTREME]

O: NOCHANGE

C: WHICH STATISTIC?

O: END

Data are stored whether or not they exceed these ranges. It is possible for the operator to specify an absurd range for a statistic to force it to print whenever collected, along with those values that are truly extreme for any other statistics being collected.

SCAN: reprint selected statistics only
 DELETE: set specified statistics to missing
 O: SCAN 34-36
 C: WHICH STATISTIC?
 O: HEARTRAT
 C: WHICH STATISTIC?
 O: END
 C: STATISTICS AS OF 0:34
 HEARTRAT= 62.14
 STATISTICS AS OF 0:35
 HEARTRAT=REJECT!*
 STATISTICS AS OF 0:36
 HEARTRAT= -10.00*
 YES?
 O: DELETE 35-1:00
 C: WHICH STATISTIC?
 O: HEARTRAT
 C: WHICH STATISTIC?
 O: END

In the preceding example, the operator discovers that sometime around Minute 35 the ECG signal became unobtainable and wishes to find the point at which bad data began. Having found that point, she deletes data from then until the current time, which she knows to be 1:00. The "SCAN" command is also useful for listing those minutes in which a statistic not sampled continuously had been collected.

MODIFY: alter data on disk, or manually enter data
 O: MODIFY 1:12
 C: MODIFICATIONS NOW ACCEPTED FOR MINUTE 1:12
 WHICH STATISTIC?
 O: GSR
 C: VALUE IS 1200.30
 ENTER NEW VALUE:
 O: 12003
 C: WHICH STATISTIC?
 O: END

In this example, the operator discovers that the GSR amplification was incorrect by a factor of 10 and manually corrects the data. The operator's options also include keeping the current value, declaring the value missing or rejected, and entering data outside the current warning range. The "MODIFY" command can also be used to enter a value not collected automatically, such as an oral temperature or a rating scale distance.

ENTER: engage in computer-prompted manual data entry
 O: ENTER 1:15
 C: WHICH LIST?
 O: 2
 C: MANUAL ENTRIES NOW ACCEPTED FOR MINUTE 1:15
 FEELGOOD:
 O: 3
 C: FEELSICK:
 O: 0
 C: FEELROTN:
 O: 0
 C: ANY OTHER ENTRIES OR CORRECTIONS FOR THIS
 MINUTE?
 O: NO

To save time when the same battery of manual items is repeated at various minutes, the "ENTER" command prompts the operator for each item in turn and does not take the time to print the current value (assumed to be initialized to missing). The command can also specify a set or a range of minutes, in which case the prompting runs through the statistic list within each specified minute.

STOP: tell the system when to stop
 O: STOP
 C: STOP AT WHAT MINUTE?
 O: 1:10
 C: SYSTEM NOW SCHEDULED TO STOP AT 1:10
 YES?
 O: STOP
 C: SYSTEM ALREADY SCHEDULED TO STOP AT 1:10
 CHANGE TO ANOTHER MINUTE?
 O: YES
 C: STOP AT WHAT MINUTE?
 O: NOW
 C: SYSTEM NOW SCHEDULED TO STOP AT 1:04

The operator can specify a specific minute or can simply specify "NOW," which stops the system at the next available minute. Any time before the system actually stops, the operator can change the stopping time.

RELEASE: stop CM4 dialogues, allow other terminal uses
 O: RELEASE

After this command is given, the CM4 program will catch up on its backlog of unprinted minute-by-minute statistics, unless they have been inhibited or the system is in test mode.

At the end of the session, the computer stops gathering statistics and returns control to the operator. After test mode, the computer asks "MORE?"; after baseline mode, the computer returns to the question "TEST, BASELINE, POSTDRUG, OR RESTART?"; after post-

drug mode, CMS exits from the computer. Any data collected during the baseline and postdrug sessions can then be copied to nine-track tape and plotted. The abscissa (time) is divided into subaxes for the baseline and postdrug parts of the session. In general, each statistic is plotted as the ordinate of its own graph, but in some cases, such as systolic and diastolic blood pressure, several related statistics are graphed on the same set of axes.

Should the session be interrupted by a system crash or accidental deletion of the active programs, all data collected through the last complete minute are recoverable from disk. Should the problem be temporary, such as a momentary power outage, then CMS can be restarted with the loss of only a few minute's data. The computer keeps track of the elapsed and clock times and the active statistics at the time of the interruption, but it does not (under our current configuration) have any way of knowing the duration of the interruption. A typical restart dialogue would proceed as follows:

```
O: ;SCHED,CM1,30,FL,F
C: CONTINUOUS MONITORING SYSTEM OPERATIONAL
  AT 10:23:14 ON 06/13/81
  TEST, BASELINE, POSTDRUG, OR RESTART?
O: RESTART
C: HEARTRAT    SYSTOLIC    DIASTOLC    GSR
  POSTDRUG MODE
  YES?
O: RELEASE
C: PREVIOUS RUN STOPPED AT 10:18, AT 1:15 ELAPSED
  PREVIOUS RUN STARTED AT 9:03, AT 0:00 ELAPSED
  RESTART AT WHAT ELAPSED TIME?
O: 1:23
C: WHEN YOU TYPE "GO", SYSTEM CLOCK WILL BE
  RESET TO 10:26:00 TO CORRESPOND TO ELAPSED
  TIME REQUEST
  READY? TYPE "GO"
O: GO
```

When typing "GO," the operator refers to the elapsed time clock available on the laboratory wall, which keeps functioning through most computer problems, and times the terminating carriage return to correspond to the beginning of the specified minute. This allows restarting the session within an accuracy of about 1 sec, which is quite adequate for measuring drug effects that occur over minutes to hours. Data between the time of the last minute before the problem and first minute after the restart are all set to missing; should a noncomputerized backup record be available for any of those data, it can be manually entered into the disk file.

Under CMS, the amount of responsibility given to the operators is considerable. There is wide scope for them to take measurements, to schedule them incorrectly, or to alter vital data with no audit trail of its having been altered. There is also wide scope for them to

duplicate measurements when the result is questionable, to schedule additional measurements when changes are occurring rapidly, and to delete credible but incorrect data when it is first realized that the data are incorrect. Our experience has taught us that complex experiments rarely work exactly as planned, but they often work well enough that the data are scientifically valid, providing that intelligent operators are allowed to make decisions about minor variations in the testing schedule to accommodate unplanned interruptions and malfunctions. Given the cost and complexity of much of our research, it is much better to adjust a timing schedule than to discard a day or a subject. The control we allow our operators under CMS maximizes the ability to make such adjustments and retain valuable, accurate data in the process.

THE PROGRAMMER'S VIEWPOINT

CMS is designed to execute in a multiprogramming environment. There are at least three concurrent programs executing simultaneously, synchronizing their operations with each other at critical points but otherwise depending on different external events to control their timing. These programs execute in a mixture of shared and separate memory, with two of them also sharing a direct-access disk file. A relatively sophisticated, laboratory-oriented, real-time operating system is necessary to support these program interactions.

CMS is implemented under the Sperry-Univac VORTEX II operating system for V77-series mini-computers. This is a disk-based operating system that supports coding in FORTRAN and assembler, concurrent execution of multiple tasks, priority levels assignable both to user tasks and to system device drivers, shared memory, and concurrent CPU and I/O activity both within a task and across tasks. Among the many extensions and modifications that we made to the operating system, two are of particular importance to the CMS system. As supplied by the vendor, the A/D converter device driver would process only one block of data at a time, using the address and length of a data buffer supplied by the calling program. When the block finished, several milliseconds of operating system overhead were consumed in getting back to the calling program to find the address and length of the next block of data to be input. In our modification of the driver, parameters for each data block and the subsequent block are passed to the driver simultaneously, so that no time is lost in restarting input at the end of each data block. This modification required changing the nucleus, or resident part, of the operating system. The other major modification consists of additions to the subroutine library for real-time program delays. While the vendor-supplied delay function allowed only the request to have execution resume after a specified delay, we added a routine to resume execution at a specified

time of day, to the nearest 10 msec. This routine allows the various components of the CMS system to synchronize many of their activities by reference to the system clock, rather than to flags or semaphores set by other components of the system.

The three major real-time components of the CMS system are Programs CM1 and CM4 and the statistical modules. CM1 is responsible for starting and restarting data collection activity, for reading A/D converter data into shared memory, for updating software clocks and flags indicating the elapsed time and the current analog input block, and for updating the initial disk record of each minute's activity. Program CM4 is responsible for conducting almost all user dialogues, for setting and clearing 1-bit flags indicating which statistics are currently active and queued to be active, and for all processing of disk data after they have initially been written to disk by CM1. There are one or more statistical modules active at any time that statistics are being collected. Each module uses the shared buffers of A/D data and any information collected through nonshared peripherals to compute one or more statistics each minute and to leave the statistics in shared memory, from which CM1 will write them to disk.

The shared memory available to CMS is 7,872 words long and is contained in an area known as Foreground Blank COMMON. Real-time FORTRAN programs can reference this area by use of unlabeled COMMON statements. The first 120 words contain the data that are written to a 120-word disk sector once per minute, including all information needed to restart CMS after an unscheduled interruption. Table 1 provides a detailed listing of the contents of this shared memory area. Its most important contents include a bit array of active statistics, indicators of starting and elapsed session time, and the current measurement values supplied by the various statistics modules. Table 2 lists the other important shared memory arrays in Foreground Blank COMMON; these include complete and sampled analog

input data, the list of statistics queued to be collected soon, and the names of all statistics available for collection by CMS in this session.

With the use of a scanning multiplexer, our one A/D converter can service up to 16 analog channels; in CMS, we restrict sampling to 10. The 10 analog input channels that are sampled do not necessarily correspond to 10 statistics on the list. The cuff pressure is read from a single channel, but its values at two different times yield statistics for systolic and diastolic blood pressure. Similarly, a computer-controlled switch allows the same electronic thermometer system to read values from two different sensors, during the first and second halves of each minute. The Romberg force platform data arrive through four analog channels, which are reduced to two (x and y coordinates of force) by software. Normally, this leads to one statistic, the length of the path traced by the center of force, but summary statistics of a spectral analysis of force on each axis are also available. While no single experiment is likely to require more than 10 channels, we may have more than 10 channels of data potentially available to the system. To control the routing of these data to the computer, all analog inputs pass through a mechanical switch matrix, capable of connecting 48 sources to 58 sinks. All sources are low impedance, and all sinks are high impedance, so that a source can drive multiple sinks, but a sink can be driven by only one source. For each experiment, we generate a list of switch assignments and a simulated picture of the switch settings, using the DECwriter, as a guide to the operator in setting the matrix to route the relevant signals through the multiplexer, filters, chart recorders, and other analog devices.

Some measures, such as heart rate and GSR, require only sampling of continuously available data. For others, such as postural stability and pupil diameter, the computer must be informed when the procedure has begun and the data are available. Currently, the communication between the operator performing these measurements

Table 1
Contents of the 120-Word Disk Sector Written Once per Minute

16-Bit Words	Name	Meaning
1	GOTIME	Minutes since midnight when this run began
1	GOELAP	Minutes elapsed when run was (re)started
1	CURTIM	Current minutes past midnight
1	THRSEC	Last 3-sec data buffer completed, 0-19
1	ELAPSD	Current minutes since start of run
1	SAMPLG	Three-second data buffer (1-20) now in progress
1	LSTDSK	Latest minute's data written to disk
1	DJKSTR	Semaphore to control which module may use terminal
1	STPREQ	Elapsed minute at which CMS is scheduled to stop
1	RUNMOD	Run mode (0 test, 1 baseline, 2 postdrug)
1	SIMTYP	Simulation type (0 none, 1 summary stats, 2 raw data)
4	TODAY	Eight-character date in MM/DD/YY ASCII format
5	ACTIVE	Seventy-bit mask indicating active statistics
60	FLSTAT	Array to hold 30 floating-point statistic values
40	INSTAT	Array to hold 40 integer statistic values

and the computer, located in another room, is rather primitive. We are using a small box with three LED signal lamps and three push buttons to indicate when such measurements are to be taken. The computer flashes one of the lights in a preset pattern (such as two short flashes, short pause, short flash, long pause) to indicate that the computer is ready for a measurement. The operator replies by pressing a button meaning either "go" or "bypass," the latter proceeding to the next measurement in sequence. A third button allows the operator to abort and restart a measurement in progress, and the other lights signal acceptance or rejection of the data by the computer. We hope to replace this box with a more versatile, alphanumeric display in the near future, probably using a hand-held unit rather than a bulkier CRT terminal.

Four disk data files are essential to the functioning of CMS, in addition to the source and object code files for the various programs. Files CMDHIS, CMDBAS, and CMDDAT contain 1, 100, and 725 120-word records, respectively. Each record is a copy of the 120-word area in COMMON described above, written by Program CM1 at 21 sec after the minute. Files CMDBAS and CMDDAT are direct access files in which one sector is allocated to each minute's data; these files contain the baseline and postdrug data, respectively. Before the session starts, those files are filled with missing data codes, indicating that no data have been collected in those minutes. File CMDHIS is overwritten each minute with the latest update of the 120 words. When the experiment ends in a scheduled or unscheduled manner, that sector indicates how much of the current baseline or postdrug data file has been used. This information can be used to restart data collection or to limit the amount of disk scanned for tape storage or graphing purposes.

File CMDTBL contains 70 card-image records. Each fixed-format image gives the name of one statistic that CMS can collect, whether it is automatically collected or manually entered, the initial range outside which data are flagged for operator attention, whether and how it is graphed, the name and priority of the module (if any) collecting the statistic, an extended title for

graphing, and the axis endpoints. All operations on statistics refer to tables derived from these card images at run time. For example, if a statistics module is collecting heart rate data, it will look up the word "HEARTRAT" in the statistics name list to find what index that statistic has in the data array and it will store the data in the corresponding memory location. This run-time allocation of storage corresponding to each statistic name allows the statistics list to be reordered as necessary, inserting new statistics whose collection is automatically implied by existing ones, without any recompilation of the unaffected statistics modules.

The key to the noninterference of statistical modules with each other is the strict timing and message-passing protocol implemented in CMS. The existence of a multi-programming operating system is not sufficient to guarantee that various measures can be collected simultaneously; their use of shared resources must be planned by the system programmer, not left to the default mechanisms inherent in the operating system. In other words, it is generally impossible to write two stand-alone measurement systems to collect independent statistics and then simply ask the operating system to run the two programs simultaneously. While the details to be presented here may be peculiar to VORTEX II, the same notions of memory mapping and block transfers are applicable to other systems designed to allow the simultaneous execution of independently developed programs.

In order to reduce the interference of programs with each other, all I/O operations under VORTEX II occur on a block-by-block basis. Whenever data are to be transferred, applications programs specify the starting and ending memory addresses of a block of data to be transferred and let a device driver manage the details of getting the data moved to or from memory. In some cases, such as the DECwriter and CRT drivers, an interrupt occurs after each character and the driver transmits or receives the next one at the soonest available opportunity. For high-speed transfers or those with critical interword timing requirements (such as sampling of an analog waveform), the driver is active only to initiate and terminate the block transfer. The transfers of the

Table 2
Memory Allocation Within 7,872-Word Foreground Blank COMMON

16-Bit Words	Name	Meaning
120		One hundred and twenty-word status and statistics report, as in Table 1
3000	ADBUFR	Two 3-sec analog input buffers, at 50 samples/sec on each of 10 channels
600	AD60	Sixty-second backlog of data, 1 sample/sec on each of 10 analog channels
5	UPDACT	Next minute's 70-bit array of active statistics
16	AOVOLT	For each of two analog channels, four real numbers representing voltages to be output automatically at the four quarters of each minute, for purposes such as switching the input channel of multiplexed sensors
16	MASCH	Message in ASCII from Program CM012, explaining which operator-initiated measurement it is awaiting or conducting at that time
280	SN70	Eight-character names of the 70 possible statistics
3835	UNUSED	Not assigned any special meaning by CMS

individual words are managed by hardware, not software. Programs cannot interrupt such data transfers, but they may access the data buffers involved while the transfer is still occurring. In such circumstances, programs that appear to be using a peripheral, such as a disk, simultaneously are actually using it in alternation. Entire blocks of data, rather than individual words, are being transferred alternately for the separate programs. For disk drives, the block is some number of consecutive 120-word sectors; for tape drives, it is a physical tape record; for terminals, it is a line of text; for the A/D converter, it is a time period during which the converter is dedicated to dumping data (from one or more channels) into one program's consecutive memory locations.

Unless special steps are taken to ensure that separately compiled programs share memory, separate programs occupy nonoverlapping memory space and lack the facilities to access other programs' data. When using the A/D converter, they cannot monitor the external world at the same time. When writing small amounts of data to disk, such as one or two words of statistical summary per minute, programs must make multiple disk accesses, even though the new information written may be less than one sector's worth. When printing statistical summaries, they must write on separate lines, not make use of the unused space after short messages printed by each other. In other words, a rational sharing of the computer's facilities would share units of time, disk space, and paper that are much smaller than the units into which VORTEX II normally divides the world. In order to achieve that subdivision, the programs must arrange to bypass the normal protection against sharing memory and then must adhere to an additional set of rules to ensure that their mutual interactions are not destructive. Operating under such rules, the programs may then access shared A/D input buffers and disk output buffers and leave messages that get combined into single, efficient lines of text display.

Under CMS, the modules that measure statistics may read data from any of the shared area (Foreground Blank COMMON), but they will usually write to only two arrays, that for collecting each minute's statistics and that for indicating which statistics are active. The former is obviously necessary, so that the statistics can be recorded and printed; the latter is to allow for some statistics to turn themselves off once collected, until specifically reactivated by the operator. (In addition, Program Module CM012 writes brief ASCII messages describing its current activity to another short array in COMMON.) Program CM4, which conducts the dialogues with the operator, builds the initial list of statistic names and can alter the arrays of active statistics. It can also set an indicator specifying when the system is scheduled to stop. All other data in COMMON are written by Program CM1, directly or indirectly. By making repeated requests to the A/D converter driver, CM1 causes two 1,500-word buffers to be alternately filled with digitized

data. At 500 samples/sec, no data over 6 sec old will remain in these buffers. Therefore, for modules with much lower sampling rates, CM1 copies the first data point per second from each of the 10 channels to another, 600-word, buffer capable of holding a 1-min data backlog. CM1 also updates counters indicating the elapsed time, the current buffer being sampled, the last second's data transferred to the 1-min buffer, and the last minute successfully written to disk. In addition to this shared memory area, managed by CMS, all CMS programs have read-only access to a system area indicating the current time of day to the nearest 10 msec.

Every 3 sec during the minute, CM1 officially requests that another block of analog input data be read into COMMON. Because the driver has already begun reading that block, no error occurs if the request is delayed by up to 3 sec after the sampling started. The request usually occurs about .1 sec after sampling starts. Just after issuing that request, CM1 copies the 30 beginning-of-second data points from the previous block to the 1-min buffer and updates counters indicating the current progress of the analog sampling and copying. In addition to maintaining the analog input, CM1 has special tasks to perform at other points within the minute. At about .1 sec, it updates indicators of the current and elapsed session time. At 21.1 sec, it initiates writing the previous minute's summary data to disk. This operation is normally complete within 1 sec, even if a lower priority editor or background compilation is competing for disk access. At the successful conclusion of the transfer, it updates the counter of minutes available on disk. At 29.0 sec, CM1 sets the data array for the subsequent minute's data to missing, so that statistics modules may begin entering data 1 sec later, at 30.0 sec. At 30.1 sec, CM1 also replaces the last minute's array of active statistics with the array of ones queued to be active for the forthcoming minute. Program CM1 and the A/D input driver have the highest priorities in the system, so that no other task can interfere with the continued input of digitized analog data.

Once a monitoring session has started, Program CM4 spends most of its time executing a small loop. It first checks to see whether any data are on disk (or still in COMMON, in the case of test mode sessions) that have not been printed; if so, it copies these data to local memory and prints the nonmissing statistics for that minute and returns to the top of the loop. If not, it checks to see whether the operator has set a sense switch on the front panel of the computer; if so, it prints "YES?" as a prompt and proceeds to process the commands described above. The dialogue continues until the operator types "RELEASE," at which point the small loop resumes. All activity by CM4 is at a priority lower than that of CM1 or any of the statistics modules that may be active at the time. Except for setting the indicator that CMS is to stop, or for harmlessly delaying another disk transfer, no action taken by Program CM4

can delay or interfere with the actions of Program CM1, and only the arrays showing which statistics are currently wanted will affect active measurement modules. This ensures that data continue to be input despite any dialogues that CM4 is conducting with the operator.

The measurement modules process the digitized data in COMMON and pass their summary statistics to the array that is written to disk each minute. Those modules that use only the analog input generally compute statistics from 0 through 59 sec of each minute and store the data during the first few seconds of the next minute. Thus, what they record for Minute 13, for example, is a summary of activity during the 13th minute. Other modules, using nonshared peripherals, may not have data available every minute. When data are available, they can be stored in COMMON at any time except between 20 and 30 sec after the minute. By 20 sec, there is a risk that the module might be interrupted by another program and miss the initiation of disk writing at 21 sec. Just before 30 sec, the data arrays are cleared for the next minute. Such discontinuously measured data tend to be stored at the nearest available minute to the termination of data collection, rather than at the next minute. For example, a force platform measurement conducted from 20 sec before the minute through 10 sec after it will be stored with the continuous data ending in the previous minute.

To facilitate adherence to these message-passing protocols, a primitive macrosubstitution facility was added to the procedure that invokes the FORTRAN compiler. With the use of this facility, the presence of "%INCLUDE" in a comment statement incorporates the text of any file named in the include statement, up to a nesting three files deep. This procedure is used to ensure that all program modules reference identical COMMON blocks and selected short subroutines, which subroutines are recompiled each time they are needed. Among these subroutines are ones to locate any statistic name in that run's array of names, to test the bits indicating that the statistic is active or queued to be so, or to fetch analog data specified by channel and elapsed minute and second from the COMMON data arrays.

DEVELOPING A NEW MEASURE

As an example of what is involved in adding a new measurement module to CMS, Figure 1 shows the code to estimate the respiration rate (in breaths per minute) from the output of a nasal thermistor, which detects the temperature difference between inhaled and exhaled air. The identical code could also process the output of a belt fastened around the chest such that its electrical resistance varies as the lungs alternately fill with air and then empty. As it is possible (but not frequently) to see respiration rates over 30, sampling the signal once per second is not adequate, and therefore, the data must be obtained from the 6-sec 50 sample/sec buffers. Ten

samples per second will be more than adequate to resolve individual breaths, so 80% of these data will be discarded, allowing the remainder to be stored in 1,200 words of memory (60 sec by 10 samples by 2 words per floating-point voltage). As 1,200 words is not a large array for our system, an entire minute's data will be stored and processed only at the end; the alternative is to discard data from each few seconds after the breaths have been located. While the latter technique saves memory, it requires more complex code to interlace the collection of data with data analysis. Making the reasonable assumption that neither the ambient room temperature nor the subject's body temperature changes much during the minute, the program uses the minimum and maximum recorded temperatures during the minute to calculate a region 5% above and below the midrange as boundaries for hysteresis, used for detecting new zero crossings of the temperature signal. The program then reprocesses the data array, looking for alternate periods entirely above and entirely below the boundary region. The sign of the first period beginning after the start of the minute is noted, and the starting time and serial number of the last similarly signed period in the minute are also noted. The time difference and breath count are then rescaled to give the count of breaths per minute.

One of the interesting features of this code is that it appears to be entirely free of input, output, and timing commands. Obviously, there are timing commands buried in the CMVLT subroutine, which will delay until the data requested are available. At that point, however, the data are simply copied from one memory array to another. Similarly, when the minute is over, the output is not written to a terminal or disk drive but is simply deposited in another array. To some extent, this is no more than an extension of the usual way of handling I/O in a multiprogramming system, in which a FORTRAN WRITE command really means "deposit into a buffer from which the operating system will write." However, this code goes beyond the convention by which reading and writing are converted into spooling operations. Data will continue to be input and output despite any action taken by this module. The data flow is a continuing feature of the environment in which this module executes, and the module simply uses the data's existence as a resource, without taking any action to request the resource. One way of looking at the situation is that we have a real-time FORTRAN module coded in a form reminiscent of memory-mapped I/O. Input operations consist of finding data in specified memory locations, and output operations consist of depositing other data into other locations.

To test new modules, data simulation Program CM2 can be incorporated into the measurement system. This module, when scheduled, executes at a priority higher than that of any of the statistics computing modules. Its function is to fill the buffers that usually hold A/D input data with simulated data instead. As the kind of simu-

```

C      PROGRAM TO ESTIMATE RESPIRATION RATE FROM A NASAL THERMISTOR
C
C      THE COMMON BLOCK INCLUDES THESE SPECIFICATIONS:
C      IMPLICIT INTEGER(A-Z) REAL FLSTAT(30) LOGICAL BITASK
C      %INCLUDE G2,,CMCOM
C      REAL RAWDAT(600),ONESEC(50),DATUM,MINVAL,MAXVAL,BPM
C
C      FIND OUT WHEN DATA WILL FIRST BE AVAILABLE
C      CALL CMGO(MINUTE)
C
C      ACCUMULATE 60 SECONDS' DATA FROM A/D CHANNEL 2 AND FIND THE
C      MINIMUM AND MAXIMUM VALUE. THE A/D CONVERTERS RETURN VALUES
C      IN THE RANGE -10 VOLTS TO +10 VOLTS.
C      1  MINVAL = -10.
C          MAXVAL = -10.
C          CURSOR = 0
C          DO 10 SECOND = 1, 60
C      C      IF CMS HAS STOPPED REQUESTING DATA, THEN EXIT
C      C      IF (STPREQ.GT.0.AND.ELAPSD.GE.STPREQ) CALL EXIT
C      C      CMVLT CONVERTS THE RAW INTEGERS TO APPROPRIATELY SCALED
C      C      FLOATING POINT NUMBERS IN THE ONESEC ARRAY. IF THE DATA ARE
C      C      REQUESTED BEFORE THEY ARE AVAILABLE, THE CMVLT ROUTINE WILL
C      C      SIMPLY DELAY UNTIL THEY ARE DUE TO HAVE BEEN COLLECTED.
C      C      THE ABNORMAL RETURN IS TAKEN IF DATA IS REQUESTED AFTER IT
C      C      IS NO LONGER AVAILABLE IN COMMON. THIS EXIT IS USUALLY
C      C      TAKEN ONLY IF THE CMVLT PROGRAM IS STARTED IN THE MIDDLE
C      C      OF A MINUTE, AFTER CMS HAS ALREADY STARTED EXECUTING.
C      C      CALL CMVLT(2,MINUTE,SECOND,ONESEC,&100)
C      C      CHECK FOR EXTREMES WHILE COPYING TO ANOTHER ARRAY
C      C      DO 10 IPOINT = 5, 50, 5
C      C      DATUM = ONESEC(IPOINT)
C      C      IF (DATUM.LT.MINVAL) MINVAL = DATUM
C      C      IF (DATUM.GT.MAXVAL) MAXVAL = DATUM
C      C      CURSOR = CURSOR + 1
C      C      10  RAWDAT(CURSOR) = DATUM
C      C
C      C      SETUP A REGION FOR HYSTERESIS AROUND THE MIDRANGE OF THE DATA
C      C      DATUM = (MINVAL+MAXVAL)/2.
C      C      MAXVAL = DATUM + .1*(MAXVAL-DATUM)
C      C      MINVAL = DATUM - .1*(DATUM-MINVAL)
C      C      SET HALVES TO -1 TO DISCARD THE SEMI-BREATH IN PROGRESS WHEN
C      C      THE MINUTE FIRST STARTS
C      C      HALVES = -1
C      C      OLDSGN = 0
C      C      BCOUNT = 0
C      C      SCAN FOR THE START OF SEMI-BREATHS ON EACH SIDE OF THE
C      C      HYSTERESIS ZONE, AND NOTE THEIR LOCATIONS
C      C      DO 20 IPOINT = 1, 600
C      C      SIGN = 0
C      C      IF (RAWDAT(IPOINT).LT.MINVAL) SIGN = -1
C      C      IF (RAWDAT(IPOINT).GT.MAXVAL) SIGN = +1
C      C      IF (SIGN.EQ.0.OR.SIGN.EQ.OLDSGN) GO TO 20
C      C      WE HAVE NOW CROSSED OUTSIDE OF THE ZONE IN A NEW DIRECTION
C      C      OLDSGN = SIGN
C      C      HALVES = HALVES + 1
C      C      IF (HALVES.EQ.1) TIME1 = IPOINT
C      C      IF (MOD(HALVES,2).EQ.0) GO TO 20
C      C      THIS SEMI-BREATH IS ON THE SAME SIDE OF THE CENTER REGION
C      C      AS THE FIRST WHOLE SEMI-BREATH WAS
C      C      LASTIM = IPOINT
C      C      BCOUNT = (HALVES-1)/2
C      C      20  CONTINUE
C      C
C      C      IF THERE HAVE BEEN AT LEAST 5 BREATHS (OTHERWISE, SOMETHING
C      C      IS WRONG), CALCULATE BREATHS PER MINUTE FROM NUMBER OF WHOLE
C      C      BREATH CYCLES OBSERVED, TIME SPAN OBSERVED (AT 10 POINTS
C      C      PER SECOND), AND 60 SECONDS PER MINUTE.
C      C      THE CODE THAT MEANS DATA REJECTED IS -32766.
C      C      BPM = -32766.
C      C      IF (BCOUNT.GE.5) BPM = 60.*(BCOUNT/(.1*(LASTIM-TIME1)))
C      C      STORE THIS DATA ONLY IF IT IS STILL BEING REQUESTED. THE
C      C      CALCULATIONS, WHICH SHOULD HAVE BEGUN WITHIN 1 SECOND AFTER
C      C      THE END OF THE MINUTE, ARE PERFORMED SO RAPIDLY THAT THERE
C      C      IS NO NEED TO TEST WHETHER WE ARE STORING THE RESULT BEFORE
C      C      THE 20-SECOND-PAST-THE-MINUTE DEADLINE. ALSO, AT THE TIME
C      C      THE CALCULATIONS ARE COMPLETE, THE FIRST FEW SECONDS' DATA
C      C      FROM THE NEXT MINUTE SHOULD STILL BE AVAILABLE IN COMMON.
C      C      IF ENOUGH MODULES ARE ADDED SO THAT THE SYSTEM IS TOO BUSY
C      C      TO CALCULATE ONE MINUTE AND CATCH THE BEGINNING OF THE NEXT,
C      C      THEN THE SAMPLING PERIOD CAN BE REDUCED TO 59 OR 58 SECONDS
C      C      TO ALLOW MORE TIME FOR CALCULATIONS.
C      C      INDEX = CMNAM('RESPRATE')
C      C      IF (BITASK(ACTIVE,INDEX)) FLSTAT(INDEX) = BPM
C      C
C      C      COME HERE AFTER FINISHING THE MINUTE, EITHER AFTER SUCCESSFULLY
C      C      STORING A RESULT OR AFTER MISSING A DATA-FETCHING DEADLINE,
C      C      AND PREPARE TO SCAN FOR THE NEXT MINUTE
C      C      100  MINUTE = MINUTE + 1
C      C      GO TO 1
C      C      END
C      C      %INCLUDE G2,,CMVLT (FOR CMVLT AND CMGO ROUTINES)
C      C      %INCLUDE G2,,CMNAM (FOR CMNAM ROUTINE)
C      C      %INCLUDE G2,,CMBIT (FOR BITASK ROUTINE)

```

Figure 1.

lated data required changes with the module to be tested, changes are frequently being made to the CM2 program. For example, to test the respiration rate code above, various clean and noisy sinusoids were placed into the part of the buffer that represents Input Channel 2, and it was verified that the reported breath rate corresponded to the frequency being simulated at the time.

PROSPECTS FOR PORTABILITY

Whenever a large, complex, and potentially useful software system such as CMS is developed to solve a common problem, one question that arises is that of assessing the practicality of exporting the system to another laboratory. There are several levels at which this export can occur: the design philosophy, hard copies of the FORTRAN code for reference, or machine-readable code for reinstallation. In the case of CMS, there are substantial increases in difficulty over these three levels of portability.

The design philosophy of CMS did not depend heavily on the Sperry-Univac V76 hardware configuration or upon particular features of the VORTEX II operating system. In fact, there were a few cases (such as the original device driver for the A/D converter) in which the operating system needed to be upgraded to serve the needs of programs like CMS. However, it is fairly clear what kind of hardware and software system is needed to support such a continuous monitoring system, and several vendors now offer appropriate configurations. With a 16-bit architecture and 256 KB of memory, we tend to use about 80% of our system memory to support CMS sessions. Other architectures and instruction sets might require somewhat more or less memory, but this gives a rough idea of the total memory required. A hardware floating-point processor is also needed, not only for speed, but also to reduce the memory requirements of the various modules, most of which would otherwise require independent copies of the floating-point subroutines.

A real-time clock is essential, but not necessarily one with a very fine grain. Our clock, for example, has a grain of 10 msec, and CMS could have been developed with a grain on the order of 100 msec. The A/D converter and associated multiplexer must have their own internal clock and direct memory access hardware, so that sampling can continue at a high, fixed rate with only occasional CPU intervention. Either the hardware or the software must support immediate restart of analog sampling at the end of any data block. We accomplish this with software, but both DEC and Hewlett-Packard offer hardware that continues sampling automatically with subsequent block addresses. It is necessary, however, for the sampling and real-time clocks to be perfectly synchronized, so that the number of samples per minute is exactly predictable. To accomplish this, we are deriving our real-time clock from the same crystal that controls sampling. Disk accesses are fairly infrequent, so that the speed and capacity of floppy

disks might be adequate. However, it is necessary for disk transfers, like A/D transfers, to be done on a direct memory access basis so that other programs can proceed independently of ongoing disk activity on any one program. Such transfers tend to be associated with hard disks, and in fact, the cost and complexity of the rest of the system are such that a hard disk would be a normal component of such a system. A plotter is useful but not essential for data display and review, and either magnetic tape or easily removable disk cartridges are needed for archival storage of each day's data from the disk files. In summary, we are looking at a medium-scale laboratory computer, well above the range of the microcomputer and the low-end minicomputer.

It is somewhat harder to describe appropriate software than appropriate hardware, because the vocabulary is less well-defined. In the vocabulary with which I am most familiar, what is required is a multiprogramming system with user-settable priorities for both device drivers and application tasks, with independently compiled tasks able to share memory arrays or have independent ones as needed. It must be possible to schedule and resume tasks on a time-of-day basis, with a grain of at worst 1 sec. Direct-access disk files must be available, and it must be straightforward for different tasks to access the same files concurrently. Separate tasks must also be able to share one terminal, so that communication can easily be shifted among different programs requiring initialization information, even if only one is communicating at run time. If there is an independent, battery-backed, real-time clock interfaced to the system, then hardware and software for power failure recovery might be useful for restarting monitoring without operator intervention.

For printed listings of the code to be useful to another laboratory, it would be necessary to supplement the printed listings with functional descriptions of part of our subroutine library. We have made a number of system facilities available through FORTRAN by the use of this library, generally functions that are difficult to code in FORTRAN or that interact with the operating system in a way not anticipated in the original language design. For example, one routine allocates additional memory and passes it to a FORTRAN subroutine as one or more arrays. This allows us to leave large arrays out of the disk image of the link-edited object code, conserving our limited library space. Another routine accesses individual bytes (characters) of an array, that being essential in text scanning as it occurs in the CM4 program. Other routines access disk or tape files without going through format processing, allow free-field key-word or value input in dialogues (such as allowing the question "HOW MANY FILES SHOULD BE SKIPPED?" to be answered with a number or "ALL"), copy or zero long strings of consecutive memory locations, and suspend task execution until a specified time of day, to the nearest 10 msec.

Any other laboratory wanting to implement CMS would need to code equivalent subroutines using the

peculiarities of its own particular FORTRAN and assembler. It might also need to make some operating system changes to implement continuous analog sampling, shared access to disk files, or an appropriate priority relationship among the user tasks and the operating system device drivers. At that point, large amounts of the code structure might be usable "as is," but many program constants would need to be changed. For example, our 16-bit integers and 32-bit floating-point numbers, combined with the 240-byte disk block size, give us room for 70 statistics (30 floating, 40 integer) along with some system status information. A system with different representation of numbers or a different disk block size would need to change these limits everywhere they occur in the programs. Some timing relationships that make use of the relatively high speed of our disks might need to be changed if slower (or busier) disks were being used; the sampling rate might be doubled if enough system memory and logical address space were available to accommodate larger A/D input buffers. In summary, the code structure might be reasonably compatible with many real-time operating systems, but it would require substantial tuning to make effective use of any such system. A machine-readable copy of the FORTRAN code would be useful as input to an editor, but a skilled programmer would be needed to provide much additional input and some possibly straightforward redesign and tuning.

Even if copies of all the routines coded in assembler and FORTRAN were supplied to another V77-series installation with similar hardware, it would not be trivial to install CMS, although it might be much less work than transferring it to another hardware configuration. Software is usually installed within the limits of an operating system, and VORTEX II is an operating system with great scope for installation tailoring and optimization. Some of the problems, such as different device numbers for similar hardware, would be trivial to solve, but others might require system regeneration. The shared memory area in our generation, Foreground Blank COMMON, is 8,200 words long, or more than 25% of any program's 32K-word logical address space. Of these 8,200 words, 328 are needed to hold the replacement analog input driver, but the remainder are available for shared data. Most VORTEX II installations run with shared COMMON of at most a few hundred words and share very little information among concurrent tasks. We configured VORTEX II for substantial shared information, in order to make use of its multiprogramming features in running different parts of the same experiment at asynchronous rates.

WHAT IS CMS?

CMS is more than a set of programs but less than an operating system. It is more than a set of programs, because it is open-ended: New modules can be added to it to implement measurements not now being taken. While programs can be added to any set of programs,

the way in which new ones are added to CMS is very restricted. There is a protocol that must be followed in order for the new program to work and yet not interfere with old ones. In exchange for adherence to this protocol, the new program gains access to a body of analog input data whose collection is managed without any specific request to do so. Just as an operating system has system calls, conventions, and key words, so does CMS, and for the same reason: to minimize the amount of code required to specify a command to the computer. Under CMS, the operations most often required in its measurement modules, sampling, timing, and data reporting become system primitives that can be assumed to always be available.

CMS is less than an operating system because it runs under an existing operating system causing no more interference than a set of independent programs with similar I/O and CPU requirements would cause. During CMS operations, program editing, compiling, and non-real-time debugging can continue at lower priorities, with very little apparent effect on such activities. No operating system features have been sacrificed to implement CMS: when it exits from memory, other programs can execute without any need to reload the standard laboratory operating system. Were the standard operating system less suitable for the kinds of information and resource sharing than VORTEX II is, then these claims could perhaps not be made. In such a case, the programs would not so much be running under the operating system as bypassing it, sacrificing its protection and comforts in order to do necessary laboratory work. Also, implementations of CMS under different hardware and software configurations could be made to resemble each other much more than the underlying operating systems do, at least from the viewpoint of the operator if not from that of a programmer. This similarity is of a sort expected from package programs such as SPSS, not from most traditional operating systems.

It is reasonable to think of CMS as an operating subsystem, an additional layer of operating system-like functions that can be placed between measurement modules and the underlying operating system, which is in turn a layer next to the hardware. No vendor can reasonably be expected to provide an operating system that is tailor-made for a laboratory hoping to do innovative research; the most we can expect from a vendor is a versatile system that can be tailored for the laboratories purchasing it. One way of performing such tailoring is the creation of a subroutine library to ease common laboratory operations and calculations. Another way is to impose standards as laboratory policy that cannot be enforced by machine security mechanisms. A third way is to provide tempting opportunities to ease program development by limiting paradigms to those for which much of the groundwork has already been laid. No user is required to utilize these routines, standards, and limitations, but the user that does is executing within a framework that remains recognizable from user to user. I choose to call such a framework an operating subsystem.