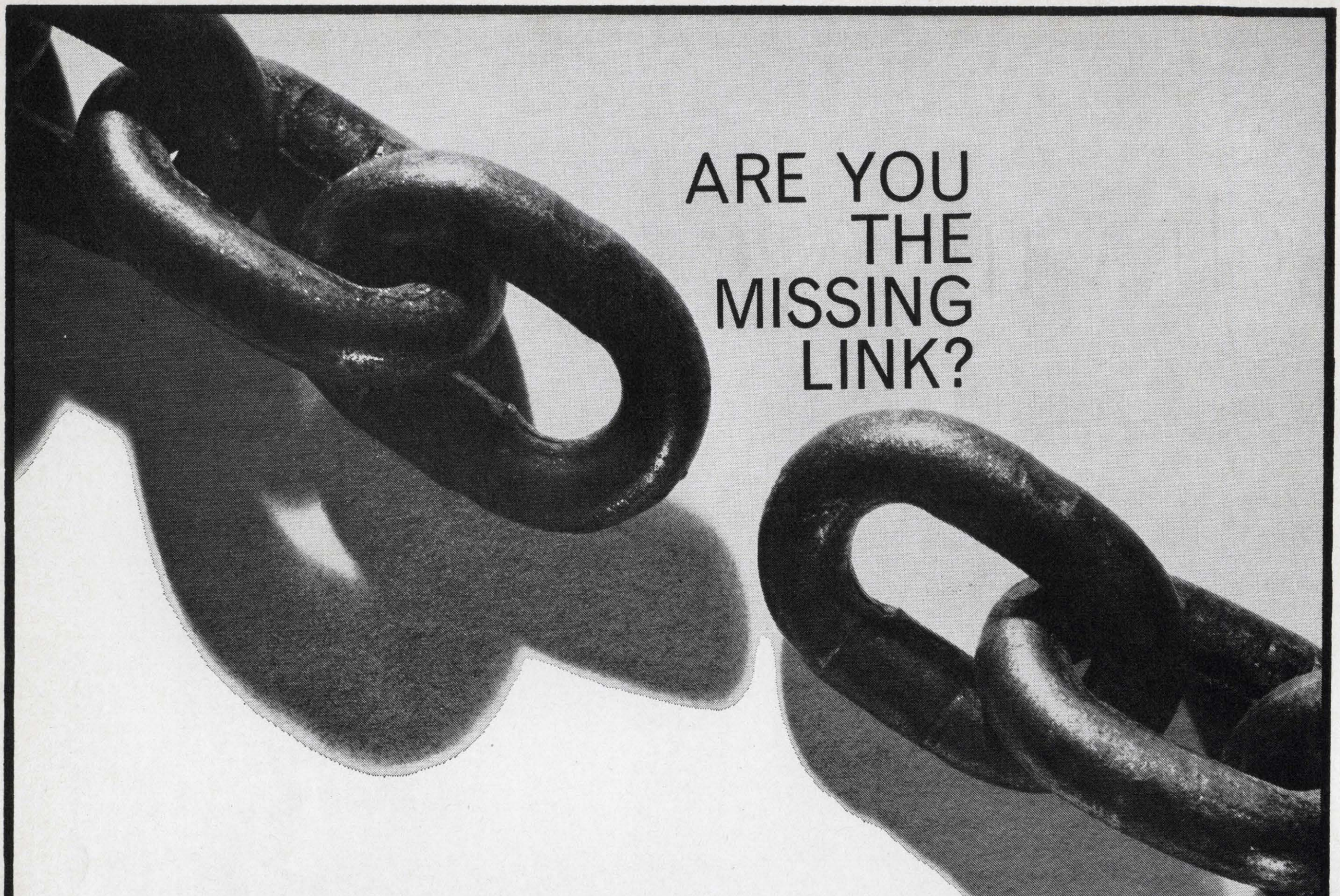
software age

FEBRUARY 1970

new york

CHICAGO

LOS ANGELES



software age®

FEBRUARY, 1970 **Vol. 4–No. 2**

Publisher–Editor .... David W. French
Assistant Publisher .... Bruce H. Dutton
Assistant Editor .... Laura Brandt
Traffic Manager .... Betsy Pavkovich
Resume Control .... Ann Wildeman
Business .... Stephenie Jensen
Circulation .... Linda Klein

CIRCULATION OF THIS ISSUE
MORE THAN 125,000
BPA Audit Applied For

SOFTWARE AGE
2211 FORDEM AVENUE, MADISON, WISCONSIN 53701
PHONE: (608) 249–0128 OR (800) 356–8141

R. John Antil

Los Angeles, Calif.: Richard Faust, 9800 S. Sepulveda Blvd. (213) 776–0100

San Francisco, Calif.: Ernesto Montano, 595 Buckingham Way, Suite 226 (415) 731–8240

Philadelphia, Pa.: Richard D. Clemmer and Phil Servus, 315 Main St., Collegeville, Pa. (215) 489–3130

Houston, Texas: W. Royce House, 4040 Southwest Freeway (713) 622–2868

Birmingham, Ala.: Herbert Martin, Box 3233A (205) 322–6528

SOFTWARE AGE is published monthly by

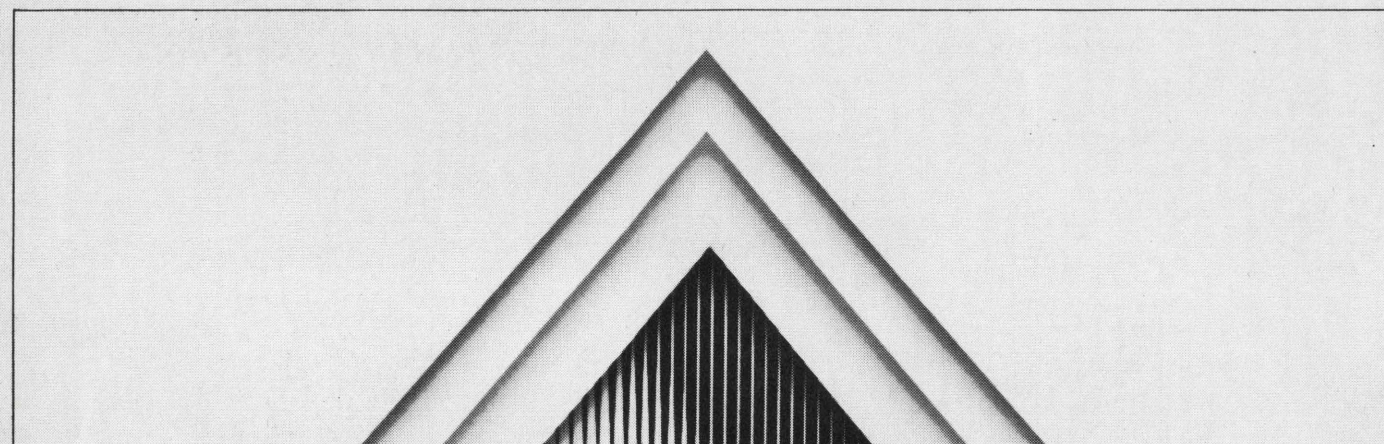
press tech inc.

2211 Fordem Avenue, Madison, Wisconsin 53701

Subscription free to qualified readers. Others, 10/yr. Individual copies, $1. Foreign subscriptions, $15/yr.

For change of address please return your address label from the magazine.

Controlled circulation postage paid at Madison, Wisconsin.



Improved Subroutine Efficiency Through Calling-Sequence Modification*

MARK H. ELFIELD
Ohio State University, Columbus, Ohio

and

CHARLES E. COHN
Argonne National Laboratory, Argonne, Illinois

Improvements in subroutine efficiency are sometimes possible through replacement of the calling sequence by instructions which do all or part of the subroutine's job. A number of FORTRAN-callable subroutines have been written to implement this concept.

In present programming practice, the calling sequence for subroutines has become standardized. With minor variations, it consists of a save-place-and-branch instruction pointing to the subroutine entry, followed by a series of words containing the argument addresses. We have found that some efficiency gains are possible if the words in the calling sequence are replaced by instructions which do all or part of the subroutine's job. The replacement is done by the subroutine the first time it is called from a given location.

Our implementations of this concept have been done on computers whose word length is sufficient to carry any operation code in association with the address of any location in memory (i.e. the 24-bit-word Honeywell DDP-24 and Systems Engineering Laboratories SEL-840). Some of the ideas may not be applicable to shorter-word machines that do not have this capability.

* Work performed under the auspices of the U.S. Atomic Energy Commission, in part under Contract No. AT(11–1)–1545 for the Chicago Operations Office.

A simple example is a FORTRAN-callable two-argument integer function subroutine whose result is the bit-by-bit logical product of the arguments. Before execution, the calling sequence for this function contains the following three words:

1. Save place-and-branch to the subroutine
2. Address of first argument
3. Address of second argument.

On entry, the subroutine replaces the above with the following:

1. Load-accumulator instruction addressed to first argument
2. No-operation instruction
3. And-to-accumulator instruction addressed to second argument.

In the process, any index tags and/or indirect flags associated with the argument addresses are carried over to the respective instructions. This takes care of cases where the arguments are subscripted variables, dummy variables or arithmetic expressions. The no-operation instruction is included just to fill up space.

After performing these substitutions, the subroutine transfers control to the first instruction in the sequence. On subsequent passes, the sequence is executed in-line with no further reference to the subroutine. Such execution saves six memory cycles out of eleven over the tightest-possible conventional subroutine coding on the DDP-24. Similar routines have been written for logical-sum and exclusive-or operations.

The concept is particularly well suited to intrinsic functions in FORTRAN. Sophisticated compilers often implement such functions with in-line coding. However, compilers for smaller computers usually resort to subroutine calls because of memory size limitations at compile time.

An example is IABS, integer absolute value. Its calling sequence has the form

1. Save-place-and-branch to IABS
2. Address of argument.

For machines with sign-magnitude arithmetic, the above may be replaced by

1. Load-accumulator instruction addressed to argument
2. Masking instruction to drop sign bit.

On the DDP-24, it takes 4 memory cycles for each execution after the first, compared to 10 for optimal conventional coding, and 17 for the existing library subroutine. For a machine with a suitable instruction set, still another memory cycle may be saved by the sequence

1. Clear-accumulator instruction
2. Add-magnitude instruction addressed to argument.

The limit on the use of this technique is if the calling sequence is not long enough for all the instructions needed to do the subroutine's job. This is true of the IABS function for machines with two's complement arithmetic, where the accumulator is negated if it is negative but left unchanged if it is positive. Here, the work can be partitioned between an instruction in the calling sequence to retrieve the argument and a special closed subroutine that operates on the accumulator. That subroutine is coded as part of IABS. The calling sequence would thus be altered to contain a load-accumulator instruction addressed to the argument, follower by a save-place-and-branch to the subroutine. This saves 1-2 memory cycles over optimum conventional coding on the SEL-840. The saving arises primarily from the improved efficiency of argument retrieval.

Another example in this area is the FORTRAN library function FLOAT. In many systems the FLOAT subroutine retrieves the argument and then calls a conversion routine that operates on the accumulator. (This conversion routine is the same one for which the compiler generates a call when an integer-to-real mode shift occurs across an equal sign.) It is then logical to replace the two-word calling sequence with a load-accumulator instruction addressed to the argument, followed by a save-place-and-branch to the conversion subroutine. Again, the increased efficiency of argument retrieval as well as the elimination of one save-place-and-branch plus return yields a saving of 6 cycles over optimal conventional coding and 13 cycles over an existing library subroutine on the DDP-24.

In one of our implementations of the method, i.e., a pair of FORTRAN-callable subroutines for performing shift operations, the need for an auxiliary subroutine is dependent on the nature of the argument. These subroutines have two integer arguments, with a calling sequence as follows:

1. Save-place-and-branch to subroutine
2. Address of word to be shifted
3. Address of shift count

with the result left in the accumulator.

If the address of the shift count is direct, the above may be replaced by:

1. Load-accumulator instruction addressed to word to be shifted.
2. No-operation instruction.
3. Shift instruction indirect-addressed to shift count.

If the address of the shift count is indirect, the above would not work, because the direct address of the shift count would be misinterpreted as the shift count itself. (In most computers, the effective address of a shift instruction indicates the shift count itself rather than its memory location.) In that case, the following is required:

1. Load-accumulator-extension instruction addressed to word to be shifted.
2. Load-accumulator instruction indirect addressed to shift count.
3. Save-place-and-branch to a shift subroutine which takes its operands from the above registers.

Again, the simplification of argument retrieval improves efficiency.

A final application to be discussed allows the FORTRAN programmer to make use of the increment-memory instruction that is available on many modern computers. This is done through a subprogram that may be referenced either as a function or subroutine to perform the work of the statement I = I + 1. That statement as normally compiled requires a load, an add, and a store instruction taking a total of three locations and six cycles. It is replaced by the statement CALL INCRMT(I) which is compiled into a two-word calling sequence. The subroutine INCRMT replaces the calling sequence with

1. Increment-memory instruction addressed to argument
2. No-operation instruction

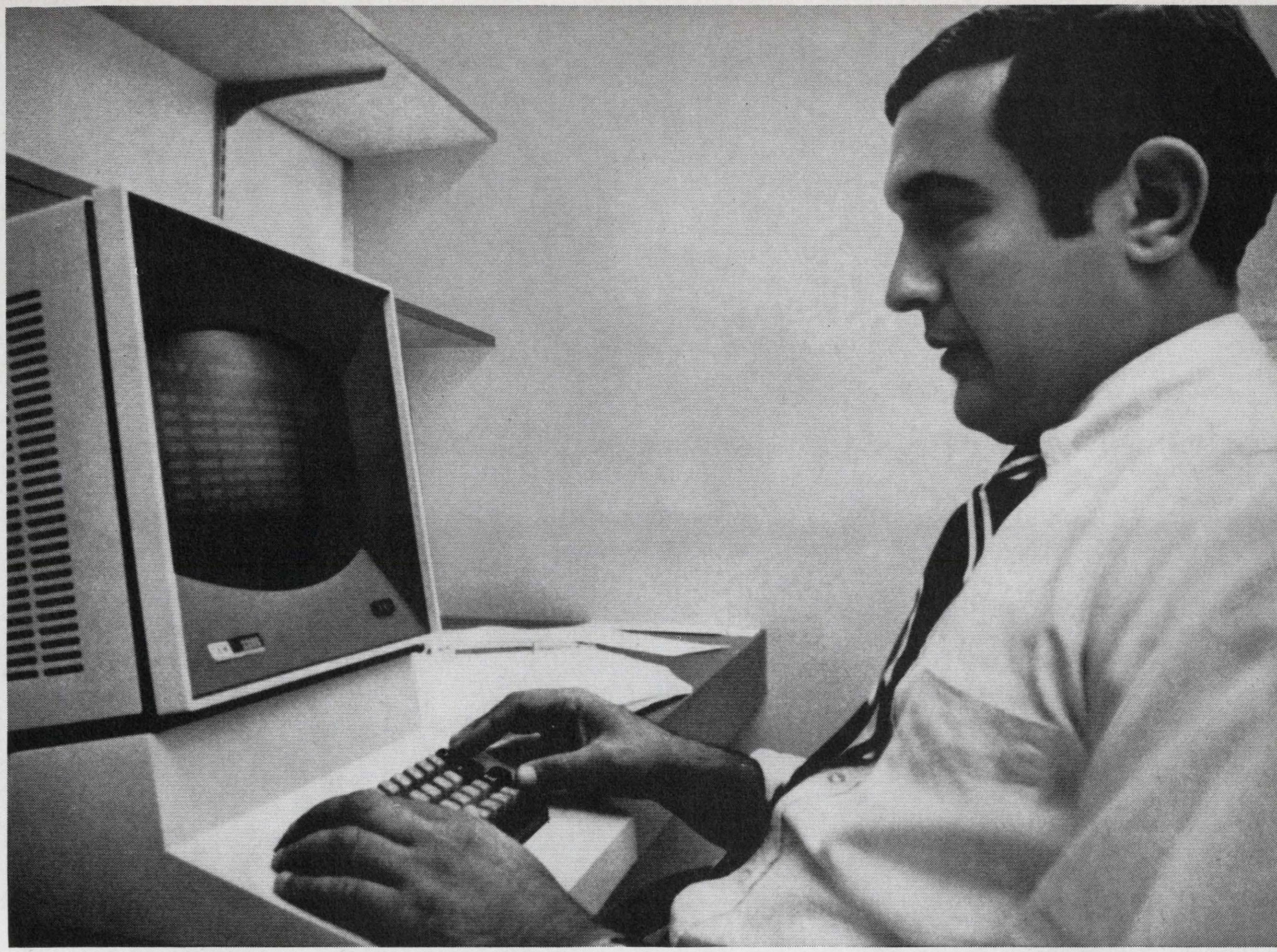
taking a total of four cycles. For the subprogram to be used as a function, the no-operation instruction must be replaced with a load-accumulator instruction addressed to the argument if the increment-memory instruction does not leave its result in the accumulator. That way, the pair of statements

```
I=I+1
IF(I-N)3, 3, 4
```

may be replaced by the single statement IF(INCRMT(I) − N)3, 3, 4.

The time savings yielded by these techniques is at a slight cost in memory space, since a subroutine that substitutes instructions takes a few more words than an optimally-coded subroutine for directly performing the given task. (For INCRMT, that penalty is offset by a saving of at least one word in the calling program for each reference.)

Finally, in contrast to most contemporary programming practices, these techniques take advantage of the original Von Neumann concept of a computer capable of modifying its own instructions.



System Design Via On-Line Simulation

By Herman A. Fischer

Simulation applications constitute a growing area of use for digital computers. The principal purpose of digital simulation is to study alternatives in the design of dynamic systems prior to the actual, real-life implementation of such systems. Up to now, digital simulation has been performed almost exclusively within the conventional off-line batch-processing type of computer environment. In this paper we examine the possibilities and practicalities of conducting simulation on an on-line basis, and point out the benefits that can be derived from on-line simulation. The requirements for an on-line facility are established, and the means of economically implementing them on a standard installation are described.

Herman A. Fischer received a B.S. in physics in 1968 from the University of California at Los Angeles, and has completed course work at UCLA for an M.S. in engineering. Prior to joining Computer Processes, Inc. in 1968, Mr. Fischer was employed by International Business Machines Corporation where he specialized in simulation applications. He participated in the development of simulators of computer systems, hospital systems and inventory systems, and built a pre-compiler to PL/I which translates Simscript-type source code to PL/I.

Mr. Fischer is presently a Systems Analyst with Computer Processes, Inc. where he specializes in advanced business applications. He has participated in the development of a large-scale Management Information System; has programmed a generalized file maintenance system; and has designed and programmed a proprietary computer system for property management firms.

Evolution of Real Systems

Real-life industrial systems generally evolve with time. A general scheme of the steps comprising the evolution of real systems (e.g., engineering systems, economic systems, transportation systems) is given in Figure 1.

One notices that the steps following (d) and (g) require the systems analyst to wait and observe how the system reacts to its environment. This implies that as the operating environment changes, the system is generally behind the times and is not optimal in performance. Consider what happens, however, if system evolution can be accelerated with respcet to time: the system can then be modified rapidly so that it can meet its demands with greater effectiveness. One method of achieving an acceleration of evolution is to create a model of the system and its environment. Simulated environmental pressures can then be applied at an accelerated rate, and the effects on the system can be determined.

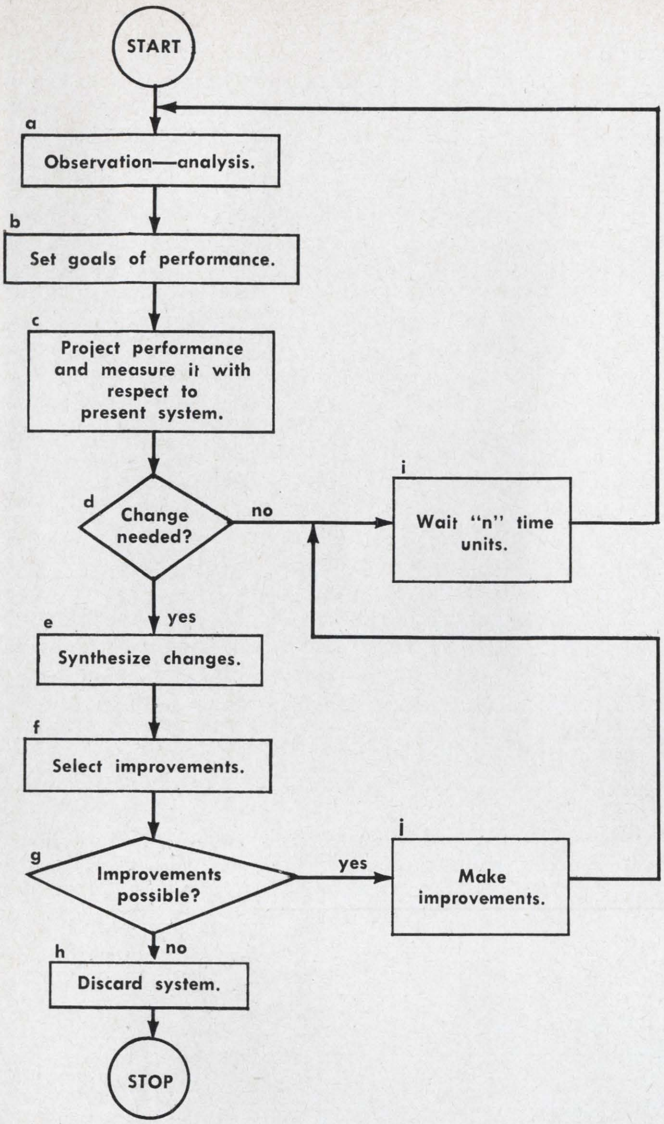
Applying the Model

The general procedure in applying a model is to represent the given system parametrically, run the model, analyze the results, and determine in what direction future improvements can be made to the system. In general, an iterative procedure is carried out wherein the model and its parameters are changed and experimentation is repeated. The procedure of changing the model and its parameters to evolve the simulated system within its given constraints can be performed in a multitude of ways. A simulation run is usually analyzed to find further improvements possible for the simulated, evolving system. Often this means that parameters of the simulated system and sometimes the structure of the model must be changed (especially when debugging the model). The model is thus run many times.

Usually, a model is run on the computer in a batch-processing mode; the modeler receives the results of a model's run remotely, analyzes these results, determines and implements changes to the model and its parameters, and resubmits the job to be re-run. In this manner, one cannot expect more than two or three exercises of the model per day. An average system study may require 10 to 100 iterations of a model, and a few weeks of debugging the model, before a useful set of results is obtained. Thus, batch-processing of a simulated system is a rather time-consuming process, and the performance of the analyst as well as the evolution of the system may be degraded since the modeler may conceive of an idea or the effect of a parameter change, and then lose his train of thought as a result of being forced to wait for computer turnaround.

In this paper we will explore the proposition that the modeler be placed on-line with the simulator, so that he can observe and control the simulation while it takes effect. In such a situation the simulator issues periodic reports as it progresses and if an interesting or questionable trend is noticed by the modeler during simulation, he can request any of the following: a slowdown in the *speed* of

Figure 1. Evolution of Systems



simulation, more detailed reports, *re-prints* of previously issued reports, and even a backtrace or *replay* of a certain simulated period. The modeler can change certain parameters, and then he can allow the model to proceed, while observing the effects. If the modeler wishes to change the model in structure, he can make the change on-line and then *re-simulate* up to the previous point in simulated time (at a high rate of simulation *speed,* much as the *fast-forward* position functions on a home tape recorder). (Note that in interpretive languages, such as GPSS, which are relatively *slow* in simulation *speed,* minor structural changes can be performed on-line without the necessity of re-simulating up to the previous point.) In this manner the problem of debugging a model is eased and the modeler is given a chance to more intimately observe the nature of the evolving system while obtaining more favorable use of the analyst's time.

It is expected that even though a given simulation run ought to cost about the same—whether run on-line or by remote batch processing—the presence of the modeler and the encouragement to experiment will cost more in terms of computer time. It is assumed, however, that the shorter amount of analyst time needed to debug and complete the project will offset this cost.

The technique of on-line simulation has been suggested in the literature[1, 2, 3, 4, 5] and attempts at implementation have been made. Some systems, such as GPSS/360–NORDEN,[1] are highly versatile (in an on-line sense), especially those which use GPSS as the basic simulation tool. However, most observers still consider on-line simulation prohibitively expensive for use as a standard technique. The successful on-line GPSS-based simulation tools are easy to use, although they are slow in running *speed* and are expensive. This paper differs from previous work in that we will point out that the technique of on-line simulation can be applied successfully with all the previously suggested features at costs comparable to those incurred conventionally.

Economic Feasibility

In evaluating the economic feasibility of on-line simulation, one must consider the objectives. One of the objectives is to be able to run on readily available systems. Special operating systems which have been designed specifically for on-line or real-time users are available, but will not be considered here due to the difficulty of obtaining and setting them up, and the fact that most of them are special purpose in nature. The use of the batch processing computer centers that are widely available will be explored. These computer centers use systems which permit the batch processing or the multiprogramming of several jobs at once.

Often these systems have available graphical display devices. (An on-line typewriter terminal can be used in lieu of a graphical device.) If one were to dedicate a computer to provide a display and to provide graphical support, as was done in the GOLD[5] Simulation or the BELL[6, 7] display analysis studies, there would be little doubt that any of the available simulation systems would be suitable. However, the objective of performing the simulation at a cost comparable to the conventional, iterative style must be observed. One way to accomplish this is to utilize a computer job shop with the necessary graphical input-output devices (a teletype or on-line typewriter) and with a computer system having a multiprogramming capability. It is assumed that only one (or perhaps two or three) simulation programs of this type would be operating at a given time, and that there would be a backlog of background jobs for the computer to perform (or time-share) when it is not busy on the simulation. For instance, while the user is studying a display, or a report, any one of a given number of background jobs could utilize the computer, so that it would not be idle for the duration of the modeling process.

When the modeler decides to let the simulation proceed, he can do so by entering on a keyboard-type input device a record that completes the simulator's (outstanding) input request. The program can then take over the central processor unit for another cycle of simulated time (typically, perhaps, a few seconds). At the conclusion of this cycle, the computer can again issue a report and wait for acknowledgement by the on-line modeler. Meanwhile, the computer can be busy executing background jobs. Thus, the user need not pay for a computer to be devoted full-time to his task as in the previously-described on-line methods.

One system which can accomplish this on a typical computer is an IBM's System/360 Operating System with the MVT option.[8] The MVT option permits several jobs of various priorities to be executed concurrently, and it provides for dynamic allocation of machine resources including core storage. The system is organized on a priority basis with the highest priority CPU task, or job, taking precedence. In the on-line simulation, the model is expected to be a high priority task on the central processor. When the model is not in use it can conveniently be *rolled-out* by the MVT operating system onto a drum storage unit, and the lessor priority tasks can be given a chance to compete for use of the central processor. If several on-line graphical (or typewriter) facilities are available, it is possible that various models can be operated by various modelers at the same time—all in a batch processing, multiprogramming environment.

To effect communication, the model, upon reaching a predetermined state or point in simulated time, branches to an appropriate routine to issue reports on the on-line device to apprise the modeler of the status of the system. The model then issues a GET-type statement to get a record from the keyboard associated with the on-line device. In this way the modeler enters requests for further reports, requests the model to proceed as previously, or (by default) permits background jobs to be processed. The modeler does not pay for any central processor time while making decisions.

To study the absolute cost involved in performing this type of simulation, one must assume that sufficient background jobs are available to keep the computer system busy during the times it is not being used by the simulation program. One must also assume that the model is written so that it would be executed quickly if no on-line interaction were required. (I.e., it would be an efficient classical model.) For example, the model would run perhaps three minutes on, say, a $380 per hour computer, such as the IBM System/360 Model 65 at certain service bureaus. (The use of a non-interpretive simulation language is assumed.) When on-line intervention is required, however, one ought to expect a moderate overhead (in CPU time) due to the additional demands on the system from the on-line input and output software and devices, which check whether or not the modeler has responded to an input request and which issue extra displays or reports requested by the on-line user. The expected systems overhead should be less than 50%. (Since the analyst will cause more overhead by his experimenting, it is desirous to keep system overhead low.)

Implementation

We have outlined a method to illustrate how an on-line input-output system can be integrated into a batch-processing operating system on, for instance, an IBM System/360 with an OS/360 MFT or MVT operating system.[8] It is assumed that the program is written in a language which supports the required input-output facilities. Available graphics aids, such as the PLAN graphics support package,[9] permit sophisticated reports to be generated; however, simple alphabetical reports, as they would appear on a typewriter, would be sufficient to provide the output needed. Note also, that if simple reports are used, the equivalent of IBM 2260 type terminals and typewriters can be used in lieu of the more expensive 2250's. One convenient language for coding on-line simulation problems is PL/I[10] or PL/:SL.[11] PL/I with a simulation-type pre-compiler is currently under experimental development. (See the Appendix for a description of PL/I:SL's simulation capabilities.) PL/I:SL permits one to code at a high, machine-independent level with source statements that are suitable for simulation. There are other available simulation languages that aid in the construction of simulation programs, but they suffer somewhat in supporting operating-system features, for example, multi-programmed and direct-access I/O (helpful for on-line uses). SIMSCRIPT I.5[12] or SEAL[13] can be used, with appropriate assembly-language[14] subroutines to provide on-line capability; SIMSCRIPT I.5 can call PL/I subroutines (a non-trivial systems programming problem). The NSS process-type language[15, 16] is an interpretive-type language (in the available version), operating much like GPSS, and for that reason, would have to have its compiler-interpreter modified for on-line usage. GPSS-type languages are available in on-line versions,[1, 2] and are ideally suitable to some simulation problems, notably those which follow transaction performance rather than system performance. However, they provide less flexibility than frequently is desired for the problem at hand, especially in queueing problems, and they are inherently expensive to run (due to their interpretive nature).

Since PL/I:SL contains the features of most of the above simulation languages, and supports necessary I/O capabilities, it will be used as the example language in the discussion that follows.

The simulation program has an endogenously caused process regularly occurring (in simulated time) for the purpose of (a) issuing reports to the modeler and (b) accepting input from the modeler. This input might either request more reports, or it might request a change in a parameter or variable of the simulated system. (See the section summarizing these interactions.) The high-level language removes the problem of how such data is input, since there are sufficient features to accomplish this at a simple level. For instance, DATA-type input-output permits one to enter a parameter name followed by its new value, and the operating system—not the programmer—makes the appropriate change to the correct word in the computer. On the other hand, one can write a rather involved input-output routine to accept a variety of input commands from the modeler. For feasibility purposes, it is assumed the program will use the type of I/O which transfers input service handling to available PL/I service routines. For output, it is assumed that the user will write a report, much as those formatted by SIMSCRIPT's report generator;[17] such a report provides the programmer with meaningful on-line displays containing statistics on both (1) the state of the system and (2) the dynamics of the system over simulated time.

Much has been written about the types of statistical output that are most useful for simulation and also about the best ways to display them. For instance, Bell's dissertation[6] gives the utility of types of information to on-line observation and decision-making. The essence of the available material indicates that the most useful way for the modeler to see different types of data displayed is side-by-side. This poses a difficulty since it conflicts with the desirability of simple report generation procedures. It requires rather involved programming for side-by-side reports to be issued. A two-fold attack on this problem is suggested. First, all reports generated on-line are also routed to the standard system printer. This permits the analyst to make a detailed examination of the simulation at his convenience. Second, it is proposed that reports be designed to issue output in multiples of pages. A page could be 50 lines by 120 print positions, or the like. Each report page would fill

```
DECLARE 1 PAGE,
        2 LINES (50) CHARACTER (120);
```

a PL/I:SL data structure, PAGE, as shown above. Each of the lines can be individually *written* by *Put String* PL/I instructions, thus enabling the use of standard output formatting mechanisms. In addition, as a page is filled and printed, it is stored as a 6000 byte page image on a *Direct Update Keyed Regional (1)* PL/I data file, for example on a 2314 disk track. An ON CHECK condition can do this automatically, as in the example in the last section of this paper. Thus, when the analyst later asks to see the i^{th} page re-displayed, the program re-reads the i^{th} record (i.e., the i^{th} page image) from the disk file of pages and displays it. (If the page number is forgotten, the user can *flip* through the pages by having a DO loop read and display a new page every half-second, or so, until the user enters an acknowledgement reply.) Thus, the need for a complex page-retrieval package is averted by using appropriate PL/I functions.

A graphic device (such as an IBM 2260 or equivalent, or a typewriter, if no graphic devices are available) can be defined as an on-line data set, operating as a sequential unit-record device, and one can use (1) WRITE-type instructions to write lines and (2) READ-type instructions to interrogate the keyboard. Again, producing hard copy and retrievable *page images* is recommended.

The method is constructed to have the simulation (1) issue a report and (2) issue a READ instruction to the on-line keyboard through which the programmer can enter commands or changes to allow processing of the simulator to continue asynchronously while the modeler enters items. In this way, the program can—through the regular occurrence of an endogenous process or event—continually check to see if the modeler has entered any input. These features are all found in presently used versions of the high-level language PL/I.

Types of Interfaces Permitted Between Modeler and Computer Program

When the modeler enters input to the computer program the following types of requests can be made:

The modeler can request a change in the duration of time simulated per interaction cycle to speed up or slow down the simulation. If he slows down the simulation, the simulator schedules its interaction cycles and reports more frequently than before in simulated time, and thus makes reports available at more frequent intervals. This feature entails the change in value of a simulation parameter, where the parameter being changed indicates the amount of time per interaction cycle.

The modeler can request the simulation to stop so he can make a detailed observation. To do this the modeler enters a reply that interrupts any simulated processes and temporarily prevents the processing of any scheduled events. The simulation model then enters the *wait* state of the CPU. During a period when the simulation is stopped, the modeler can request detailed reports which might aid him in examining a trend in the simulation and determining its cause. It is assumed that the modeler has provided reports of sufficient depth to provide answers to questions not anticipated when writing the program. A simple way of accomplishing this is to have one or more reports which provide complete printouts of all processes or events in the system and their attributes, much in the way that GPSS permits statistical and table output formats. (Along these lines, see the SNAP block of Boeing's GPSS.[2]) These can be facilitated in part by using PL/I's DATA-type output instructions. Further programming enables the modeler to request a *trace* to be in effect, to provide output similar to that of GPSS's trace option. This sets up ON CHECK conditions in PL/I or PL/I:SL, which print out changes in the values of attributes of an entity or a process entity. This feature is very useful for on-line debugging of a computer model.

The modeler can request a change or modification in one or more parameters of the simulation. This parametric variation is accomplished by changing system parameters (permanent entities in Simscript; static structures in PL/I and PL/I:SL) and attributes of temporary entities. When debugging a simulation program by this method, one can correct a value that is in error and continue the simulation. When exercising a model, this type of parametric variation aids evolving or optimizing the system.[18]

In addition, one frequently wishes to modify elements of the model. A modification of the model can take one of two forms. One can modify a single element of the model, such as a single procedural statement (in GPSS, a single block) and then either continue simulation or start over. In GPSS the change of a single block is facilitated by the interpretive nature of the language, and a slight change during simulation does not necessitate re-assembly and reloading of the simulation program. The nature of a PL/I:SL procedural change will be discussed later. In another type of model change, one is faced with a modification of an involved logical nature (i.e., one involving many GPSS blocks, or several PL/I:SL process procedures). Even in GPSS this type of change will generally require re-assembly and the re-starting of the simulation. Because of the nature of the large change, it is often desirable to forfeit the on-line use of a console for a while, so as to have the time to carefully plan out the change and to determine its repercussions. Thus, in the type of model under consideration, a major change need not be performed on-line.

Generally, when making modifications in batch processing versions of languages, such as Simscript, PL/I or PL/I:SL, it is necessary to recompile a procedure, process, or event subroutine containing the procedural step(s) to be changed. Whereas in GPSS the change can be accomplished during simulation, in PL/I:SL the change requires stopping the simulation, making the correction to the proper subroutine or process procedure(s), recompiling that procedure and re-link editing the model. (This takes a minute or two with PL/I:SL on an IBM System/360 Model 65.) If frequent changes of this nature are anticipated by the on-line user, the user might wish to use certain on-line compiler tools, such as the graphic job processor,[19] which simplify the job control language requirements. The proper use of libraries of link-edit modules on disk aids is to assemble previously compiled procedures and overlays for the link-edit step. A compilation need only delay the online user a minute or so, rather than preclude his use of the computer during a scheduled time period. Thus, the capability exists in batch processing languages to make on-line changes, although the procedure for this is not as elegant as GPSS's *change* block since it requires re-starting the simulation. (Note that a PL/I:SL simulation ought to require less time to restart than GPSS due to the slower interpretive nature of GPSS.)

An additional feature which might be desirable in online simulations, is the ability to alter the time status of the simulation. That is, it might be desirable to be able to backtrace the simulation to *re-observe* a simulated event or to observe the effect of altering a parameter and retracing over a portion of the simulation. This would be very valuable in debugging where one *retraces* and then requests additional reports and trace printouts to locate a *bug*. Backtracing can be provided for a simulation study in any language on an IBM System/360 computer, in a somewhat primitive manner. It is required, at certain points in simulated time, that the simulation model perform an action to record its status on a drum storage device, so that when backtracing is desired the operating system can restore the computer program to what it was at the point in simulated time when it was recorded on the drum. On a System/360 the operating system provides a checkpoint-restart feature,[20, 21] which can be used at certain points in simulated time to issue checkpoints (i.e., to save the status of the model). Then, when backtracing is desired, the modeler executes a *restart* job which will restore the model to what it was at the time of the checkpoint. The checkpoint-restart feature is accessible at the source statement level from PL/I and PL/I:SL and is available through *help* blocks to a GPSS program. The checkpoint-restart feature is presently available on the PCP and MVT versions of OS/360. In this way the use of the checkpoint feature permits backtracing with any simulation language on System/360.

Example

Figure 2 gives an example of how a high-level simulation language, such as the PL/I language with a simulation pre-compiler, can be used to perform a PUT-type of online operation (2260 CRT or 1050 typewriter) in a simulation program with a minimal coding effort. The

Figure 2. Example: Outpatient Clinic Simulator

```
  CREATE PATIENT;
  N = N + 1;
  APPT-TIME = ROUNDOFF (DESIRED-TIME);
GO-TO-COFFESHOP:
  TAKE SERVICE-TIME/MINS;
  TAKE EATING-TIME/MINS;
  FILE PATIENT IN WAITING-ROOM-QUEUE;
  •
  •
  •
ROUNDOFF:
  PROCEDURE (DESIRED-TIME);
              /* THIS IS A FUNCTION SUBROUTINE */
              /*   TO ROUNDOFF THE APPOINT-    */
              /*   MENT TIME.                  */
              /* FUNCTION "DOES" FOR EACH IN-  */
              /*   TERVAL IN AN HOUR; I. E., 45 */
              /*   MIN, 30 MIN, 15 MIN, AND 0 MIN */
              /*   PAST THE HOUR—IF INTERVAL   */
              /*   IS 15 MINUTES.              */
    DO I = TRUNC (59/INTERVAL) TO 0 BY —1;
    IF MPART (DESIRED-TIME) >= 1*INTERVAL
       THEN RETURN (HPART (DESIRED-TIME)
       +1* INTERVAL);
              /* RETURNS PROPER APPT TIME—     */
              /*   MPART IS FUNCTION RETURN-   */
              /*   ING MINUTES OF ARG, HPART   */
              /*   RETURNS HOURS OF ARG (AS IN */
              /*   SIMSCRIPT).                 */
    END;
  END ROUNDOFF;
REPORT:
  PROCESS;
              /* PROCESS-TYPE SUBROUTINE       */
              /*   PRINTS REPORT AND ACCEPTS   */
              /*   INPUT REQUESTS FROM THE     */
              /*   ON-LINE USER.               */
  DECLARE
    1 PAGE,
      2 LINES (50) CHARACTER (120);
              /* PAGE IS A DATA STRUCTURE CON- */
              /*   TAINING 50 LINES.           */
  ON CHECK (LINE#)
    BEGIN;
              /* BEGIN BLOCK IS ENTERED EACH   */
              /*   TIME THE VALUE OF THE VARI- */
              /*   ABLE "LINE#" CHANGES. IF A  */
              /*   PAGE IS FULL (WHEN 50 LINES */
              /*   HAVE BEEN WRITTEN) IT IS    */
              /*   PRINTED ON THE ON-LINE DE-  */
              /*   VICE AND IS STORED ON THE   */
              /*   DISK.                       */
```

```
IF LINE# > 50 THEN
   DO;
   LINE# = 1;
   WRITE FILE (PAGE-FILE) FROM (PAGE) KEY
     FROM (PAGE#);
             /* PAGE-FILE IS ON THE DISK. */
             /*    PAGE# IS THE NUMBER OF THE */
             /*    PAGE.                      */
   PAGE# = PAGE# + 1;
             /* INCREMENT NO. OF PAGES.       */
   LINES = ' ';
             /* BLANK OUT LINES ON "NEW"      */
             /*    PAGE.                      */
   PUT FILE (CRT)
        EDIT ('PAGE', PAGE#, ':')
             (SKIP,A,F(5),A);
             /* WRITE ON-LINE PAGE'S HEADING. */
   END;
END;
             /* END OF AUTOMATIC LINE-        */
             /*    CHECKING, BEGIN BLOCK.     */
DISPLAY:
  PUT STRING (LINES (LINE#) )
    EDIT ( N 11 'PATIENTS HAVE BEEN CREATED
       . . . ')
       (A);
             /* FORMAT A LINE IMAGE.          */
             /*                               */
             /* WRITE LINE IMAGE ON CRT AND   */
             /*    ON PRINTER (SYSPRINT) FOR  */
             /*    HARD COPY.                 */
  PUT FILE (CRT)
    EDIT (LINES(LINE#))
       (SKIP,A);
  PUT FILE (SYSPRINT)
    EDIT (LINES(LINE#))
       (SKIP,A);
GET-INPUT:
  GET FILE (CRT)
    EDIT (CHARACTERS)
       (SKIP,A);
             /* READ INPUT REQUEST FROM       */
             /*    CRT'S TYPEWRITER.          */
  IF CHARACTERS = 'STOP' THEN STOP
       SIMULATION;
             /* REQUEST TO STOP THE SIMULA-   */
             /*    TION.                      */
    ELSE IF CHARACTERS = 'WAIT' THEN
             /* WAIT FOR 1 MIN = 60E3 MILLI-  */
             /*    SEC.                       */
   DO;
   DELAY (60E3);
   GO TO GET-INPUT;
             /* AT END OF INTERVAL, GET A     */
             /*    NEW REQUEST.               */
   END;
  ELSE IF CHARACTERS = 'PAGE' THEN
             /* REQUEST TO RETRIEVE I-TH      */
             /*    PAGE.                      */
   DO;
   GET FILE (CRT)
        EDIT (I)
             (F(3));
             /* GET I (PAGE# OF PAGE TO BE    */
             /*    REDISPLAYED)               */
   DECLARE 1 DUMMYPAGE LIKE PAGE;
   READ FILE (PAGE-FILE) INTO (DUMMY-
     PAGE) KEY (I);
   PUT FILE (CRT) EDIT (DUMMYPAGE)
     ((SKIP,A));
             /* PRINTS WHOLE STRUCTURE OF     */
             /*    "LINES".                   */
   GO TO GET-INPUT;
   END;
    ELSE GET STRING (CHARACTERS) DATA;
             /* ACCEPTS A PARAMETER CHANGE    */
             /*    IN THE FORM "XXX = YYY, QQ */
             /*    = RR.R . . .;" OR A NULL REPLY */
             /*    (I.E., NO PARAMETER CHANGE) */
             /*    IF BLANK REPLY GIVEN.      */
  TAKE 10/MINS;
             /* WAIT FOR TEN MINS OF SIMU-    */
             /*    LATED TIME TO ELAPSE.      */
  GO TO DISPLAY;
  END REPORT;
```

example shown consists of a portion of a routine creating records for patients walking into an outpatient clinic. In this example, the entering patient is given an appointment time for the same morning; goes to a coffee shop for a doughnut and coffee; and then waits for his appointment. A roundoff function gives the appointment time on the proper interval, i.e., every 15 minutes, 20 minutes, or hourly, when given the time the doctor is expected to finish with his previous patient. The roundoff function insures that the patient arrives on the interval before the time when the doctor is expected to be free, so that there is a continual supply of patients for the physician. A process called REPORT issues a report every ten simulated minutes; it outputs the number of patients created (and other data), and then accepts a reply from the modeler. If the reply is STOP, then this simulation run is to cease. If the reply is WAIT, then the simulation program frees the computer for one minute (60 x 10^3 milliseconds) for other tasks. If the reply is PAGE, it is followed by the page number of a previously printed page to be re-displayed on the CRT. If these replies aren't entered, the program assumes the modeler has entered any number of parameter changes. (A *null* reply is equivalent to zero parameter changes.) For instance, it is shown how the modeler could indicate that patients are to be given appointments on the hour, at twenty after the hour, and at forty after the hour, instead of on 15 minute intervals.

This example indicates the facility with which on-line simulation can be accomplished, and how straightforward it is to mechanize the simulation with present day tools (languages and operating systems).

Summary of Interfaces Between Modeler and Computer Program

The model is expected to (1) issue periodic reports to the modeler via a graphic display terminal and (2) permit inquiries of the system by the modeler. Requests by the modeler can include the following:

(1) Change the duration of time simulated per *interaction cycle.*
(2) Stop the simulation for detailed observation.
(3) Permit communication between modeler and model of the form:
  (a) Change of one or several parameters.
  (b) Change or modification of an element of the model.
  (c) Alteration of the time status (i.e., backtrace).
(4) Permit communications between the modeler and a stored program. I.e., permit the modeler to write a sub-routine to do something on his command or request.

Summary

Digital simulation has traditionally been carried out *off-line* within a batch-processing environment. In this paper the feasibility and practicality of conducting digital simulation on-line is discussed. We have considered the computer systems aspects of going *on-line,* specifically in terms of the IBM System/360 and its PL/I language, and have also considered the economics of so doing. The benefits to the modeler of on-line simulation have been pointed out. In conclusion, it appears that on-line simulation (1) is of considerable benefit to the modeler; (2) can be installed within the modern multiprogrammed, batch-processing environment without a major effort in terms of time and cost; and (3) can be operated at programmer-machine costs comparable to those of conventional, off-line simulation.

APPENDIX

Features the experimental implementation of PL/I:SL adds to PL/I, version 4.

DECLARE Statement Attributes:

- level-1-structure-name TEMPORARY ENTITY° (PROCESS NOTICE)° . . .
- level-1-structure-name PERMANENT ENTITY° . . .
- any-element-set-variable SET° [{ *SEQUENTIAL*† | DIRECT† | *OWNER*° | MEMBER° }] [{ *FIFO*° \| AT-TAIL‡ | LIFO° \| AT-HEAD‡ | RANKED { HIGH | LOW } (ranking variable) }]
- process-procedure-name PROCESS‡°
- element-variable PROCESS CONDITION‡
- any variable . . . SIGNAL on-condition (permits automatically signalling other than the "CHECK" on-condition)

Built-In Functions:‡	*Value Returned*
FIRST-OF (set-expression)	pointer (except seq sets not containing ptr's)
LAST-OF (set-expression)	pointer (except seq sets not containing ptr's)
NEXT-OF (set-expression)	pointer (except seq sets not containing ptr's)
PREVIOUS-OF (set-expression, ptr-expression)	pointer (except seq sets not containing ptr's)
RANK-OF (set-expression, ptr-expression)	pre-defined precision of ranking attribute
TIME-OF (scheduled process-notice-ptr-expression)	float dec (6)
MEMBER-OF (set-expression, ptr-expression)	bit (1)
EMPTY-OF (set-expression)	bit (1)
TYPE-OF (set-expression)	bit (1) (0 = sequentially organized set; 1 = direct organization)

° See SIMSCRIPT[17] Manual for further details. Note that a Simscript "event" is a zero-length PL/I:SL "process."
† See PL/I:SL Specifications[11] for further details.
‡ See NSS[15, 16] for further details.

Set Operations°

- CREATE temporary-entity [SET pointer-variable]
- DESTROY temporary-entity [CALLED pointer-expression]
- FILE pointer-expression IN set-name [set-qualifier to over-ride declared organization]
  - Where set-qualifier has the form
  - set-qualifier = { AT-HEAD | AT-TAIL | BEFORE (pointer-expression) | AFTER (pointer-expression) }
- REMOVE pointer-expression [SET pointer-variable] FROM set-name
- DO ptr-var = specification [, specification] . . . [, specification]
  - Where specification has the form
  - { { pointer-expr 1 | ALL-OF } [TO pointer-expr 2] [{ FWD-IN | BKW-IN }] set name | pointer-expression }

Process Verbs:

- START \| CAUSE process [CALLED ptr-exp] [{ AT time-exp | BEFORE \| AFTER (ptr-expr) }]
- TERMINATE \| CANCEL [{ process-pointer-expression | THIS-PROCESS }]
- SUSPEND [{ process-pointer-expression | THIS-PROCESS }]
- RELEASE process-pointer-expression
- WAIT [n] FOR process-condition [, process-condition . . .]
- SIGNAL { process-condition | (process-pointer) } [n TIMES]
- TAKE n [/ units]
- STOP { SIMULATION | SIM }
- PROCESS

Miscellaneous Verbs:°

- ACCUMULATE
- FIND \| COMPUTE

REFERENCES

1. Hunter, S. and Reitman, J., "GPSS/360-NORDEN, A Partial Conversational GPSS," *Digest of the Second Conference on Applications of Simulation,* (1968), p. 147.
2. Stephenson, M. E., "Graphic Simulation with GPSS," ob. sit. 1, p. 151.
3. Lackner, M. R., "Graphic Forms for Modeling and Simulation," *Simulation Programming Languages,* IFIP Working Conference on Simulation Programming Languages, North Holland Publishing Company, Amsterdam, (1968), p. 33.
4. Greenberger, M. and Jones, M., "On Line, Incremental Simulation," ob. sit. p. 13.
5. Holley, M., *An Online Graphical System for Digital Design,* Ph.D. Thesis in Engineering, University of California, Los Angeles, (1968).
6. Bell, T. E., *Graphical Analysis Procedures for Simulation and Scheduling,* Ph.D. Thesis in Business Administration, University of California, Los Angeles, (1968).
7. Bell, T. E., "GAPSS (Graphical Analysis Procedures for System Simulation)," ob. sit. 1, p. 145.
8. IBM System/360 Operating System: Concepts and Facilities, *IBM Systems Reference Library,* C28–6535.
9. IBM System/360 Operating System: PLAN Graphics Support Package Application Description, *IBM Systems Reference Library,* H20-0535.
10. IBM System/360 Operating System: PL/I Reference Manual, *IBM Systems Reference Library,* C28-8201.
11. Corey, C., "Preliminary Specifications of an Experimental Simulation Language Based Upon PL/I," IBM Los Angeles Scientific Center, (Nov. 1968), No. 320–2621.
12. *User's Manual, SIMSCRIPT I.5,* Version 1 for IBM System/360, Consolidated Analysis Centers, Inc., Los Angeles, California, (1968).
13. "Simulation, Evaluation and Analysis Language (SEAL)," *IBM Contributed Program Library,* 360D-15.1.005.
14. IBM System/360 Operating System: System Programmer's Guide, *IBM Systems Reference Library,* C28–6550.
15. Parente, R. and Krasnow, H., "A Language for Modeling and Simulating Dynamic Systems," *Communications of the ACM,* Vol. 10, No. 9, (Sept., 1967).
16. Blunden, G., Leavenworth, B., Parente, R. and Pierce, S., "Language Specification for a New Simulation System," IBM Corporation, Advanced Systems Development Division, Yorktown Heights, New York, (1967), 17–206.
17. Markowitz, H., *SIMSCRIPT: A Simulation Programming Language,* Prentice–Hall, Englewood Cliffs, N. J., (1963).
18. Molnar, G., "Self-Optimizing Simulation," ob. sit. 3, p. 248.
19. Brown, S. H., "Conversational Job Control," IBM Systems Journal, Vol. 7, No. 3–4, (1968).
20. IBM System/360 Operating System: Checkpoint/Restart, *IBM Systems Reference Library,* C28–6656.
21. IBM System/360 Operating System: PL/I Programmer's Guide, *IBM Systems Reference Library,* C28–6594.

Reducing The Simulation Credibility Gap

Richard A. Kaimann*

It has been said that the computer is one of the few inventions that has enabled man to expand his mind rather than his muscle. The applications of computer technology have reached staggering proportions and the use of the computer is now commonly viewed in the firm—from the largest to the smallest. The realm of applications has evolved from the most sophisticated, scientific applications down through a spectrum of applications to the most trivial of bookkeeping tasks. The future of the applications in business apparently is bounded only by the imaginations of the systems analysts and designers of computer activities.

One of the manifestations of the growth of computer applications is in the area of simulation. The use of simulation in the firm has become widespread. Companies have examined the use of simulation from the point of view of determining the optimum number of operators in a tool crib to problems of machine replacement and breakdown and to the point of simulating the entire firm.

Many analysts have run into difficulty in selling the results of a simulation exercise to top management. Perhaps there are good reasons for management's lack of acceptance of simulated results, but just as likely, the credibility gap may be attributed to a rather hidden reason instead of the obvious ones.

** Dr. Kaimann holds a B. S. degree in Management Engineering from Rensselaer Polytechnic Institute, the M. B. A. degree from Marquette University, and the Ph. D. degree from the University of Iowa. He has served as a senior systems analyst with the Allis–Chalmers Manufacturing Company, Director of the Educational Data Processing Research Project at Marquette University, and as associate with the Iowa Educational Information Center in Iowa City, Iowa. Currently an Associate Professor in Industrial Management at Marquette, he is the co-author of "Educational Data Processing: New Dimensions and Prospects" (Boston: Houghton–Mifflin Company, 1967), as well as numerous other articles.*

PROGRAM TO COMPUTE
CHI-SQUARE VALUE

Language: FORTRAN II
Machine: IBM 7040
Running Time: 2.70 minutes

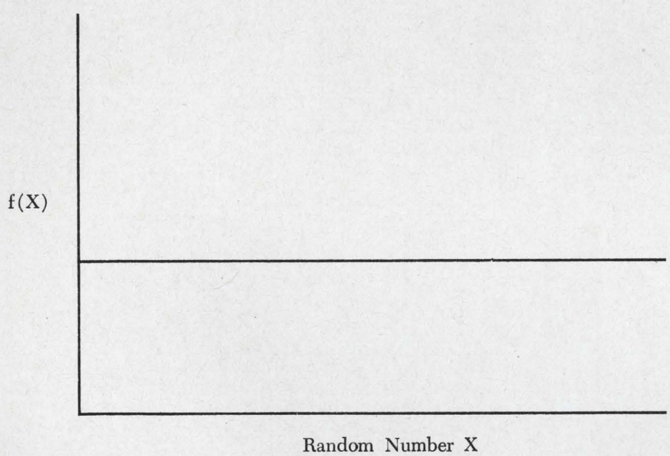
```
$JOB   R3550      INITIALIZE RANDOM NO.      DR. KAIMANN      010  015
$IBJOB            NODECK
$IBFTC
C       INITIALIZE RANDOM NO. GENERATOR AND NO. OF VALUES REQUIRED
        DIMENSION A( 1000)
        ALB = 12493
        ALC = 87345
        FLIMIT = 146850
        CT = 0
C    GO TO RANDOM NUMBER GENERATOR
      1 GO TO 111
C    RETURN
    222 A ( LA + 1 ) = A (LA + 1 ) + 1.
        CT = CT + 1.
        IF (CT — FLIMIT ) 1, 2, 2
C    REQUIRED NO. OF VALUES HAVE BEEN GENERATED AND TABLED
C    COMPUTE DIFFERENCES
      2 CHISQ = 0
        DO 30 K = 1, 1000
     30 CHISQ = CHISQ + (((A(K) — (FLIMIT/1000.)) ** 2) / (FLIMIT/1000.))
C    PRINT RESULTS
        WRITE (6, 9000) CHISQ, CT
   9000 FORMAT( 1H1, 5X, F20.10, F10.0)
        STOP
C    RANDOM NUMBER GENERATOR
C    NUMBER GENERATED IS 3 DIGIT FIXED POINT
C      LOCATED IN LA
    111 ALA = ALB
        ALB = ALC
        ALD = ALA + ALB
        IF ( ALD — 100000.) 12, 3, 3
     12 ALC = ALD
        GO TO 4
      3 ALC = ALD — 100000.
      4 LA = ALC * .01
C    RETURN TO MAIN PROGRAM
        GO TO 222
        END
$ENTRY
```

Exhibit 1

On the surface, managers may merely distrust the language of the computer analyst. It is alien to them and hence evokes a *natural* barrier. Beneath this surface distrust lies the very real possibility that the simulation itself may not be acting in the fashion that the analyst expects. It is to this particular delimited problem area that this paper is devoted.

Simulation may be defined as the process of conducting experiments on a model of a system in lieu of either: (1) direct experimentation with the system itself, or (2) direct, analytical solution of some problem associated with the system.[1] The model is merely a representation of the actual situation or system coming under scrutiny. This may be a set of statistical distributions that represent such things as frequency of occurrence of machine breakdown or length of service required on the machine, etc. One method used in the simulation of such activities is the Monte Carlo Method, a technique with which most systems analysts have become quite familiar.

Figure 1
EQUAL PROBABILITY OF OCCURRENCE

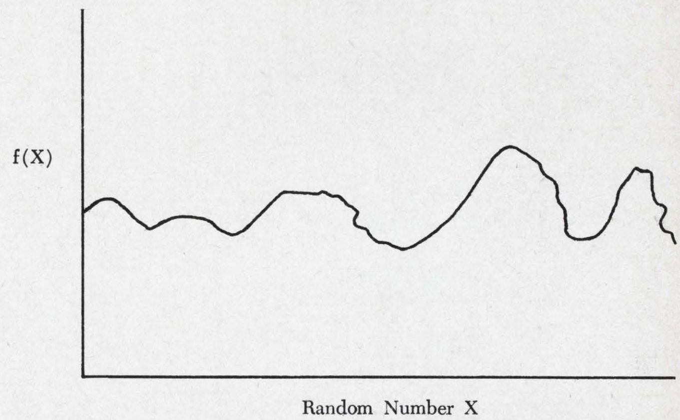


In the design of the simulation study, the analyst utilizes a random number generator to dictate that point to enter in the frequency distribution. Then the value associated with that point in the distribution is used in the simulation activity and in any subsequent decision process. Since it is not the purpose of this discussion to treat simulation per se, it will be assumed that the process of conducting a Monte Carlo simulation is fairly well known.

Of prime importance, then, is whether or not the random number generator is actually generating what may be considered random numbers. It is imperative, therefore, that in a simulation exercise, the random number generator should be validated. The question is how to do this. The remainder of this discussion is devoted to a methodology for at least making the preliminary, cautionary checks on the random number generator.

Two simple techniques will be shown that will serve this purpose. These will indicate if the random number generator is not non-random. This is to say that an initial test will indicate if the numbers that are being generated are skewed significantly to the right or to the left. In other words, is there a disproportionate frequency of occurrence? There are relatively simple tests to show that the random number generator is not non-random. To prove true randomness, on the other hand, would require a higher level of statistical competence that is frequently not available if one has concentrated on an education in computer technology as opposed to statistics. Validation tests should at least provide some degree of assurance that the random number generator is behaving reasonably well.

Figure 2
UNEQUAL PROBABILITY OF OCCURRENCE



The first of two criteria to provide this assurance is that each number in the series offered by the random number generator have an equal probability of occurrence. If one were to plot the frequency of occurrence of each random number on the Y axis as opposed to the number itself on the X axis, that distribution should be rectangular or a horizontal line. If there is not an equal probability of occurrence, the plot would be a jagged line. These are shown in Figures 1 and 2. The Chi-Square test is a statistical technique available that will test the degree of conformance to this postulated distribution.

The second criteria to validate randomness is that the random number generator not become repetitive while the numbers are still being used. This is a test for recursiveness. In other words, when does the random number generator begin recycling or producing the same series of numbers in the same order. The method to test for this condition is merely to see when a sequence of one hundred numbers, for example, occurs again in the same sequence. This would be an indication that the random number generator had become recursive. It would be useless to continue a simulation once the generator was recursive because the results would be identical to those which have been accumulated to that point.

To demonstrate how these tests may be accomplished, a particular random number generator will be described and illustrated. The method for testing its conformance to the rectangular distribution will also be presented, and finally, the test for recursiveness will be demonstrated.

The Random Number Generator

The following set of FORTRAN II program steps is postulated to produce three digit random numbers.

[1] J. H. Mize and J. G. Cox, *Essentials of Simulation* (Englewood Cliffs, New Jersey, Prentice–Hall Publishing Company), 1968, p. 1.

Exhibit 2

PROGRAM TO DETERMINE IF RECURSIVENESS OCCURS
WITHIN RANGE OF SIMULATION DURATION

Language: FORTRAN II
Machine: IBM 7040 16K
Running Time: 22 minutes

```
$IBJOB          NODECK
$JOB    R3550   RECURSIVENESS    DR. KAIMANN    300  015
$IBFTC
C   DETERMINATION OF RECURSIVENESS OF RANDOM NUMBER GENERATOR
C   INITIALIZE RANDOM NO. GENERATOR AND NO. OF VALUES REQUIRED
      ALB = 12493
      ALC = 87345
      FLIMIT = 146850
      CT = 0
      L = 1
C   ESTABLISH TABLE FOR FIRST 100 NUMBERS GENERATED
      DIMENSION FST100( 100)
C   FILL THE TABLE USING THE RANDOM NUMBER GENERATOR
      DO 20 I = 1, 100
      GO TO 111
   20 FST100( I ) = LA
      WRITE (6, 9020) CT,L
 9020 FORMAT (1H1, 3HT1L, F10.0, 15)
C   ESTABLISH TABLE FOR SECOND 100 NUMBERS GENERATED
      DIMENSION SECND( 100)
C   FILL THE TABLE USING THE RANDOM NUMBER GENERATOR
      L = 2
      DO 30 I = 1, 100
      GO TO 111
   30 SECND(I) = LA
      CT = 200
      WRITE (6, 9030) CT, L
 9030 FORMAT (1H, 3HT2L, F10.0, 15)
C   COMPARE THE SEQUENCE IN THE TWO TABLES
   38 DO 31 I = 1, 100
      IF (FST100(I) — SECND(I)) 32, 31, 32
      31 CONTINUE
C   AT THIS POINT THE FIRST 100 NUMBERS ARE REPEATED
      WRITE( 6, 9003) CT
 9003 FORMAT( 1H1, 22HGENERATOR REPEATING AT, F10.0)
      STOP
C   SHIFT THE SECOND TABLE
   32 DO 36 I = 1, 99
      J = J + 1
   36 SECND(I) = SECND(J)
      L = 3
C   GET A RANDOM NO. FOR THE LAST POSITION IN THE TABLE
      GO TO 111
   40 SECND (100) = LA
      CT = CT + 1.
      CALL SSWTCH(1,J)
      IF (J — 1)54,55,54
   55 WRITE(6, 9006) CT
 9006 FORMAT(1H, F10.0)
   54 IF (CT — FLIMIT) 38, 39, 38
   39 WRITE(6, 9005) CT
 9005  FORMAT( 1H1, 26HGENERATOR NOT REPEATING AT, F10.0)
      GO TO 38
C   RANDOM NUMBER GENERATOR
C   NUMBER GENERATED IS 3 DIGIT FIXED NUMBER
C   LOCATED IN LA
  111 ALA = ALB
      ALB = ALC
      ALD = ALA + ALB
```

```
1 ALA = ALB
  ALB = ALC
  ALD = ALA + ALB
  IF (ALD - 100000.) 2,3,3
2 ALC = ALD
  GO TO 4
3 ALC = ALD - 100000.
4 LA = ALC * .01
```

Each time statement 1 is reached, a three digit fixed point random number will be available for use at statement 4 in the location called LA. By defining initial values of ALB and ALC, a different pattern of random numbers can be generated. It assumed for this evaluation that initially:

ALB = 12493
ALC = 87345

Tracing the first three cycles of the generator, the following is found.

	ALA	ALB	ALC	ALD	LA
initial		12493	87345		
cycle 1	12493	87345	99836	99838	998
2	87345	99836	87183	187183	871
3	99836	87183	84019	184019	840

Thus the first three random numbers generated are 998, 871, and 840.

The Rectangular Distribution Check

The Chi-Square (χ^2) test for conformity to the rectangular distribution is described as follows. Assume that the above random number generator was used in a simulation exercise. Further assume that at the point at which the exercise was terminated 146,850 random numbers had been required. Thus it would be necessary that these 146,850 numbers be distributed in a fashion that is not statistically significantly different from a rectangular distribution.

The value of χ^2 for the sample is computed with the formula

$$\chi^2 = \sum_{i=1}^{1000} \left[\frac{(0_i - e_i)^2}{e_i} \right]$$

where:

0_i = observed frequency of each number
e_i = expected frequency of each number

Since the random numbers are three digits, there would be 1000 possible numbers (000 to 999).

Exhibit 1 displays the FORTRAN II program required to perform these calculations using this particular random number generator. The calculated value of χ^2 was 61.28^+.

The formal statistical test to determine if this frequency of occurrence of these 146,850 numbers is statistically significantly different from a rectangular distribution is as follows:

Ho: The generated sample has a rectangular distribution
Ha: The generated sample does not have a rectangular distribution
$\alpha = .05$
d.f. = 999
χ^2 critical = 1073.65[2]

Since the χ^2 value from the sample was less than the χ^2 critical, H_o is accepted. That is, the distribution may be considered rectangular within the range of the first 146,850 random numbers.

[2] To calculate the χ^2 critical for large values of degrees of freedom see W. Meredith, *Basic Mathematical and Statistical Tables for Psychology and Education* (New York: McGraw-Hill Book Company, 1967), p. 247.

The Recursiveness Check

To determine if the random number generator has become repetitive, two tables are established. The first contains the first 100 random numbers generated. The second table contains the next 100 random numbers. The two tables are compared. If they are equal, repetition has set in. If they are not equal, the entire second table is shifted up one position and a new random number is generated and inserted at the bottom of the second table. Again the two tables are compared.

This procedure is performed repeatedly until the required number of random numbers has been generated. Exhibit 2 contains the FORTRAN II program to perform these calculations. In this case, the problem did not become repetitive for 146,850 numbers.

On the basis of the two tests performed, it is reasonable to assume that the random number generator is behaving according to expectations.

Conclusions

The two methodologies and accompanying programs presented here are suggested as guidelines for use with any simulation procedure using randomly generated numbers. Careful adherence to the procedures will aid in avoiding one of the pitfalls associated with performing simulations in the business environment.

The procedures will not aid in selling top management on simulated results. They will however, offer the analyst an additional degree of assurance in the validity of his simulation study.

TROUBLE-TRAN PRESENTS XTRAN'S ADVENTURES IN FORTRAN

By GEORGE N. VASSILAKIS

Send your ANSWER to the problems posed here in each issue to:

TROUBLE-TRAN EDITOR

software age

P. O. Box 2076
2211 Fordem Ave., Madison, Wis. 53701

You can also profit by submitting PROBLEMS for this feature. If your problem in FORTRAN programming is selected for use in this feature, you will receive $50.00

TROUBLE-TRAN'S Objectives:

1. To have fun.
2. To promote USA Standard FORTRAN by pointing out differences and inconsistencies of existing FORTRAN Compilers.
3. To alert programmers to the physical limitations of hardware.

PROBLEM OF THE MONTH

Problem 22: THE LOGICAL FUNCTION. Submitted by Mario DeNobili, Brooklyn, New York.

Main Program
```
      COMMON K
      LOGICAL XTRAN
      K=0
      IF(K.EQ.0 .OR. XTRAN(.FALSE.))
      K=K+1
      WRITE(6,100) K
100   FORMAT(I5)
      STOP
      END
```
Subprogram
```
LOGICAL FUNCTION XTRAN(X)
COMMON K
LOGICAL X
K=K+2
XTRAN=X
RETURN
END
```
What will be the value of K printed by this program?

Does this program violate the rules of USA Standard FORTRAN?

Answer to Problem 20:

This problem was state was as follows:
```
         DO 40 I=1.3
         A=I
       IF(I/2.EQ.A/2.) GO TO 20
         WRITE(6,10)
     10 FORMAT(9H I I ODD)
         GO TO 40
     20 WRITE(6,30)
     30 FORMAT(10H I IS EVEN)
     40 CONTINUE
         STOP
         END
```
What is the output?

The answer, of course, depends on the compiler used. Those compilers which follow the rules of the USA Standard FORTRAN would not even compile this program. When standards are violated, we enter the gray areas of FORTRAN, where we have many pros and cons as to what a compiler should do. The results listed below indicate that mixed mode arithmetic is allowed not just by the third generation computers, but also by some first generation (small) computers. Unfortunately, not all of them agree on what method should be used to compute mixed mode expressions.

Group 1

This group has a clear majority. The method used here is to first perform the division I/2, then convert the result to a floating point number before it is compared with A/2.0. The method of divide-convert-compare generates the following three lines of output:
```
I IS ODD
I IS EVEN
I IS ODD
```
The following systems belong in this group:

IBM-360; IBM-1440
GE-645; GE-255
UNIVAC 1108; UNIVAC 1230
Sigma 5/7
Com-Share's XTRAN
SDS-940
B-8501

Group 2

Instead of converting I/2 to floating point, the minority converts A/2.0 to integer, or converts every operand to floating point before any arithmetic is performed.

continued on page 43

conference countdown

FEBRUARY

16–19 Annual Computer-Aided Circuit Analysis and Design Institute, Tampa, Fla. Contact: Dr. George Zobrist, Department of Electrical Engineering, University of South Florida, Tampa, Fla. 33620.

17–19 Computer Software and Peripherals Show and Conference, Midwestern Region, Chicago, Ill. Contact: Show World, Inc. 37 West 39th St., New York, N. Y. 10018.

19–20 ADAPSO's 28th Management Conference, Los Angeles, Calif. Contact: ADAPSO, 551 Fifth Avenue, New York, N. Y. 10017.

23–25 Data Processing Supplies Association, Winter General Meeting, New Orleans, La. Contact: DPSA, 1116 Summer St., P.O. Box 1333, Stamford, Conn. 06904.

23–25 American Management Association, 16th Annual Conference on Electronic Data Processing, New York, N.Y. Contact: John McClane, 135 W. 50th St., New York, N.Y. 10020.

27–28 National Association of Computer Assisted Analysts (NACAA) Multiposium, Los Angeles, Calif. Contact: NACAA, Multiposium, Mary Bragg, P.O. Box 2802, Fullerton, Calif. 92633.

MARCH

23–25 INFO–EXPO–70, Washington, D.C. Contact: Paul Zurkowski, Information Industry Association, 1025 15th St. N.W., Washington, D.C. 20005.

23–26 IEEE International Convention and Exhibition, New York, N.Y. Contact: IEEE, 345 E. 47th St., New York, N.Y. 10017.

25 QUEST—Symposium Exposition of Time-Sharing Systems & Techniques, Los Angeles, Calif. Contact: Merrill Goulding, 4612 W. Jefferson, Los Angeles, Calif. 90016.

APRIL

3 Computer Graphic Workshop, Association for Computer Machinery, Rosslyn, Va. Contact: Spec. Int. Group for Graphics, Box 933, Blair Station, Silver Spring, Md. 20910.

7– 9 Computer Software and Peripherals Show and Conference, Western Region, Los Angeles, Calif. Contact: Show World, Inc., 37 West 39th St., New York, N.Y. 10018.

8–10 Numerical Control Society, 7th Annual Meeting and Technical Conference, Boston, Mass. Contact: NCS, 44 Nassau St., Princeton, N.J. 08540.

MAY

13–15 Association for Educational Data Systems, 8th Annual National Convention, Miami Beach, Fla. Contact: AEDS, 1201 Sixteenth St., N.W., Washington, D.C. 20036.

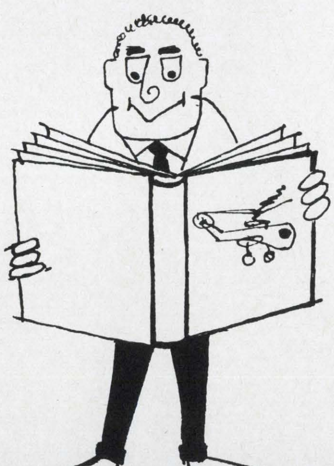
ANALYSTS

PROGRAMMERS

SYSTEMS PROGRAMMERS

help write the book on management information systems

...IN CONNECTICUT



If you're ready for a wide-open career, consider this.

Finding the answers to mass transportation is a dynamic growth situation at Sikorsky. Right now, we're designing and building tomorrow's helicopters. Unique aircraft like our SKYCRANES . . . with an 18-ton lift capability. Or sleek, Vertical Take-Off and Landing Vehicles to transport people in airliner comfort to and from congested airports.

We'll need new, greatly expanded Management Information Systems to back up these years-ahead programs. And EDP people with financial, manufacturing or administrative experience to insure their success. We need *all* kinds of people—at *all* levels of experience. People who enjoy working on standard batch applications. Or people who relish the challenge of graphics, teleprocessing and time-sharing. And people with interests in-between. Naturally, a knowledge of COBOL and/or assembler language would be an important consideration.

You'll like our environment. We pay well. There are friendly, talented people to work with. The tools you need to work with are here—including system 360 models. And Connecticut living with its cultural and recreational facilities can be abundantly satisfying.

So, if you feel you qualify, get in touch. For a prompt response, air mail your resume, stating salary requirements, to:
MR. LEO S. SHALVOY,
PROFESSIONAL EMPLOYMENT

Sikorsky Aircraft

DIVISION OF UNITED AIRCRAFT CORPORATION

STRATFORD, CONNECTICUT

An Equal Opportunity Employer

new products

Data Project Management System Junior Version

Lutter and Helstrom, Inc., Chicago, have the availability of an assignment control system called DPMS Jr. (Data Project Management System junior version). This system has been designed especially for the dp system/programming group of the majority of computer installations (typically having less than 10 or 12 systems designers/programmers on the staff) rather than the large shops.

A backlog file is maintained for each person on the staff, with job numbers, description of the job, target dates, hours estimated, revised estimates, hours spent and remaining plus percentage completion. Summary information includes total time available, assignment time spent, time unassigned, number of projects open, number of projects closed.

This simplified system was created for the smaller shops that now must accumulate statistics for the new IRS rulings.

For more information, circle No. 2 on the Reader Service Card

Interactive Data Retrieval for IBM 1130

Aeronautical Research Associates of Princeton, Inc. (ARAP) has developed an interactive data retrieval program for use on an IBM 1130. It is designed to operate directly from the console or, if the 1130 is so equipped, via a remote terminal. Batch mode processing is also possible.

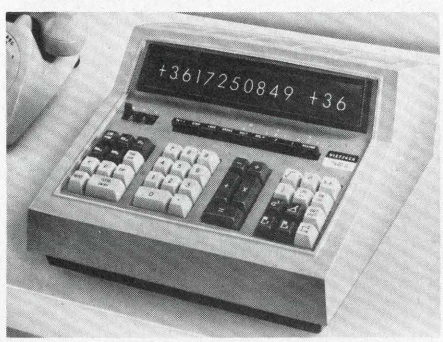
The program handles both the numerical and textual fields of a data bank with equal ease. Selection of a particular subset of data can be accomplished by prescribing that certain properties be "greater than," "less than," "equal," or "not equal," to some value or that they "start with," or "contain," some element.

Sorting of the selected data can be on the basis of numerical value or alphabetical order. The output of the selected and sorted subset of data can be directed to either the console, a remote terminal, or a line printer. Furthermore, the particular subset items to be listed can be chosen interactively, or prestored instructions can be utilized. Complete freedom, including the insertion of global headings, local headings, date and paging, is available to the user.

For more information, circle No. 3 on the Reader Service Card

Dietzgen Desk-Top Computer

A unique, extremely compact desktop computer, combining greatly expanded calculator functions with big computer programming capabilities, has been introduced by Eugene Dietzgen Co., Chicago based manufacturer of drafting, surveying and reproduction equipment.



The new Dietzgen Desk-Top computer weighs only 12 pounds, using large scale integrated circuitry throughout. It is capable of creating comprehensive programs for about 85% of engineering and scientific problems commonly encountered. The computer is said to eliminate dependency upon large, expensive computer systems for all but the most complex problems.

For more information, circle No. 4 on the Reader Service Card

On-Line Data Management System

Proprietary Software Products, Inc., Arlington, Va., has developed a software package that provides the architecture and programming support for implementation of on-line applications. It is intended to be used by organizations that plan to go on-line or desire to upgrade their existing on-line operations. It is a multi-application, multi-terminal system that uses IBM/360 equipment and operates with DOS, OS-MFT, OS-MVT and supports CRT terminals in a local/remote environment.

Through a combination of user written programs and program modules provided in DIALOG II an on-line application is developed in days rather than months. The design philosophy reduces application development to a practical level without restricting the creativity of a programming staff.

For more information, circle No. 5 on the Reader Service Card

Version 2.0 of MURS

Webster Computer Corporation, Danbury, Conn., has announced the availability of Version 2 of its DOS MURS (Machine Utilization Reporting System). The DOS MURS, an extension of IBM's S/360 Disk Operating System, provides an automatic accounting of all programs run under DOS on a S/360 computer. Version 2 of MURS accounts for all program executions in each partition, whether run in batch mode or in single program initiation mode.

Included in the machine utilization record created for each program are Wait time, Supervisor time, and Problem Partition time. A unique time capturing method accumulates Supervisor time separately for each program based on the partition being serviced.

MURS provides a Daily Log in all program executions and four Production and Testing Summary Reports.

For more information, circle No. 6 on the Reader Service Card

Fixed Asset Accounting Package

The McCormack & Dodge Company, Needham, Mass., has announced the availability of its fixed asset accounting package. The package has already been implemented and is written for any S/360 tape or disk configuration. A version is also available for the Model 30 MFCM card system.

The package handles the four major depreciation methods. Conversion from declining balance to straight line is effected automatically by program control.

There are 15 programs in all, including file load and maintenance, updating, and reporting capability.

The year-end calculation of tax depreciation products a report in the format required on the Form 1120 Federal Tax Return.

For more information, circle No. 7 on the Reader Service Card

FINANCIAL CURRENTS

Value Computing Inc. has been formed in Cherry Hill, N.J., headed by three computer systems experts, Vincent J. Bannan, James H. Garrett, and Henry R. Heidler, Jr. The company has developed a unique system to enable computer users, particularly IBM 360 users, to obtain a much higher degree of productivity from their equipment.

The Value Computing system solves a problem which has frustrated computer users since the advent of the third generation equipment. The system provides the means for scheduling and matching programs to be executed on third generation computers, giving the user all of the advantages of multiprogramming.

In addition to its scheduling service, Value Computing offers a wide variety of expertise in systems to control and measure data processing. The company specializes in multiprogramming, multiprocessing and other techniques for maximizing computer system efficiency, in addition to the unique online performance scheduling system.

* * *

Call-A-Computer today announced that it had reacquired from *Conley Corporation* the rights to market CAC service in the Upper Midwest area and will establish its own sales force in Minneapolis on February 1, 1970.

Call-A-Computer, formed in 1967, is a major time sharing firm providing local time sharing service in 35 cities to customers in 34 states, and the District of Columbia. Conley Corporation, a Minneapolis firm originally formed in 1967 as Conley Associates, will continue to specialize in computer software services, terminal sales and leasing, and related computer consultation services.

* * *

International Telephone and Telegraph Corporation has announced its entry into the U. S. data peripheral equipment market and also reported a substantial expansion of its worldwide data services operation in 1969.

The ITT Data Equipment and Systems Division will have its headquarters in a new facility at East Rutherford.

The division, headed by Andreas H. Kruse, vice president, who formerly held a group staff marketing position with ITT Data Services Division, will market and service ITT-manufactured equipment and systems designed particularly for the computer-based communications industries. These products range from peripheral equipment, such as data printers and data visual terminals, to security monitoring systems, front-end processors and concentrators.

Internationally, new data-processing service centers were opened in England, West Germany and Denmark and ITT's first computer center in Latin America was established in Rio de Janeiro.

* * *

URS Systems Corporation plans to acquire *Remote Computing Corporation,* a Los Angeles based computer network information service, according to a joint statement made by URS Systems Corporation's President, Richard De Lancie and Remote Computing Corporations' President, Joseph Hootman.

Terms of the proposed acquisition, which will not become effective until 1971, call for two annual payments each of URS common stock. The total number of shares to be issued will depend on Remote Computing's earnings. The agreement is subject to approval by the Boards of Directors of both companies and the shareholders of Remote Computing, a closely held company.

Remote Computing provides time sharing and other data processing services, principally through remote access to computer centers in Los Angeles; Palo Alto, California; and Detroit.

* * *

Bernard Goldstein, president of *United Data Centers, Inc.* (OTC), a national network of electronic data processing centers, announced today that UDC has purchased the assets and business of *Sports Data Corporation* of New York City. Sports Data provides golf handicapping and billing services to country clubs. An agreement in principal to purchase Sports Data was previously announced.

Sports Data, which does its major billing from April through October, balances United Data Center's Computafuel System, an application for fuel oil dealers, which runs in the data centers during the winter months.

* * *

Formation of a new company, *Applied Computer Marketing Corporation* (ACMC), to provide professional sales and marketing within the computer industry, has been announced by George M. Barr, vice president of *KDI Corporation's* Computer Products and Systems Group.

Initially, the new firm will sell, service and lease nationally a line of KDI Interactive Data Systems data communication products, disc controllers, and disc systems. In addition, ACMC will be sales representative in the Western U.S. for the digital computers and core memory products of Datacraft Corporation, Ft. Lauderdale, Fla., and is currently negotiating with a number of other firms.

Co-founders of the new firm are Alan Spafford, president; Jack Walkins, and Nick Matsoukis.

* * *

The acquisition of *Comps Inc.* by *Growth Industry Computing* has been completed. This announcement was made jointly by Robert L. Thaler, Chairman of Los Angeles based Growth Industry Computing, and Arthur E. Bartlett, President of Orange based Comps Inc. Comps was acquired for an undisclosed amount of Growth shares.

The merger provides added dimensions and capabilities to both companies which offer computer based services to the marketing community.

* * *

The Air Force has awarded *EDP Technology, Inc. (OTC)* a prime contract to develop a system capable of simulating the response of an unmanned space vehicle to ground commands.

The interactive simulation system would enable the Air Force Space and Missile Systems Organization more thoroughly to test ground data handling of a space mission prior to the actual flight. It could also be utilized for training of ground personnel.

The agreement is the first prime Air Force contract awarded to EDP Technology, Inc., of Washington, D. C., a diversified organization that develops and applies advanced technology to education, health, industry and a number of other fields.

* * *

new applications

Over 27,000 students at UCLA registered for spring semester classes using a new computer-based enrollment system. Of that number, 10,000 students registered "on-line" through thirty TV-like communications stations manufactured by Computer Communications, Inc.

Less than a second after the student identified himself through the communications station, his pre-registration class selection list was flashed on the screen. If all the classes the student had previously selected were available, the schedule was confirmed. If a class was filled or for any other reason not available, the student requested an alternative course through the keyboard/console. Subsequently, the computer confirmed the class selection by printing an official study list. Average enrollment time per student was about one minute.

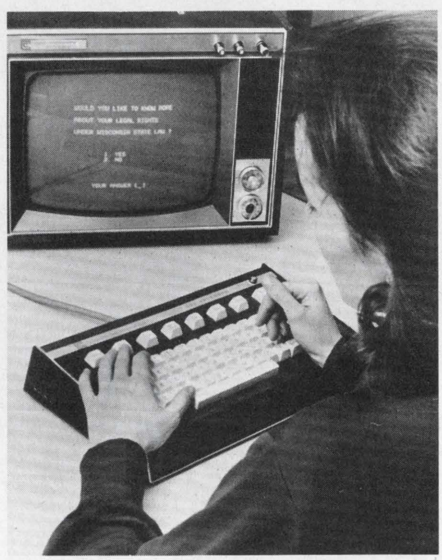
In addition to speeding the enrollment process, the computer safeguards certain registration procedures. For example, the computer prevents a student from shopping and enrolling in several classes and deciding later which classes to discard. The computer also prevents a student from enrolling in more classes than his course limit allows. Ultimately, it is intended that the computer assign classes to rooms tailored to the enrolled number of students to make optimum use of campus space.

The enrollment system was developed by the Office of Systems and Procedures utilizing the remote console system of the UCLA Campus Computing Network.

In addition to the enrollment system, the Campus Computing Network handles thousands of different jobs weekly for UCLA and more than 100 other colleges and universities in the Western States. The jobs processed in some 20 different computer languages, touch on almost every facet of classroom instruction, scholarly research, and administration of UCLA and participating schools.

The system hardware includes forty CC-30 Communications Stations consisting of a keyboard, controller, and television display. The multiplexer and interface units to the IBM System 360, Model 91, were also manufactured by CCI.

* * *



This visual display unit, above, with a typewriter-like keyboard is linked to a computer as part of an experimental program to improve the quality and availability of legal services to the disadvantaged.

The IBM 1130 computing system at the Dane County Legal Services Center, Madison, Wis., contains client records, interview sequences and various legal documents. Staff members use the system to interview clients and gather information on their cases. As questions appear on the screen, an operator types responses on the keyboard. The computer can display data on the screen or print it out.

The computer-based program frees attorneys from routine work to devote more time to personal consultation with clients.

* * *

A computer that can help save the lives of policemen has been at work for the State of Delaware for over ten months, and has proven itself as a tool for administration and law enforcement.

The computer, an IBM System/360 Model 40, is the nucleus of the information network called DOVER (*D*elaware *O*n-Line *V*ehicle and *E*nforcement *R*ecord system). DOVER provides immediate information on drivers' license records, violations, and stolen vehicles for state troopers on patrol.

Colonel Charles Lamb, Superintendent of State Police, said, "The system not only makes the trooper's job easier, it may also save his life, because he knows when he's stopping a stolen car.

"DOVER gives us the potential to develop a centralized law enforcement information system on a computer, eliminating many manpower-consuming clerical tasks, such as those involved in tracing suspects throughout the state," Colonel Lamb said.

Using DOVER, troopers can check a vehicle's status before stopping it by radioing the license number to police headquarters. There, operators query the computer through typewriter-like terminals and relay the response to the officer. The whole process takes less than a minute.

James Deputy, State Highway Controller, whose department is responsible for the computer operation, said the system is running perfectly and is fully utilized by the Motor Vehicle Division of the State Highway Department and the State Police.

The Motor Vehicle Division uses DOVER to maintain three types of information: drivers' license records, including a history of violations; vehicle registration and ownership data, and the stolen vehicle file.

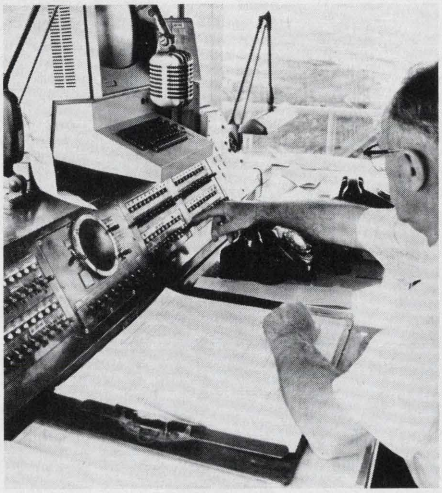
An added advantage of the system is better control over access to the records, Mr. Deputy said. To query the computer, one must know the proper code numbers. Also, certain information, such as the total number of violations on a driver's record, is available only to authorized personnel within the Motor Vehicle Division.

The computer also does accounting chores for the Highway Controller's office, but its speed of operation enables the police to use the system 24 hours a day with virtually no interruption of other functions.

* * *

A computer in a Houston railroad yard automatically directs the switching of some 4,000 freight cars a day while maintaining a perpetual inventory of every car in the classification yard.

The IBM 1800 data acquisition and control system allows Southern Pacific railroad to switch each car into any of 64 tracks in the Englewood Yard, where trains are assembled.



Yardmaster W. L. Timmons, above, directs the operation from his office high above the tracks.

The cars, headed for delivery to hundreds of different destinations across the country, are lined up in front of a switch engine that slowly pushes them to the crest of a man-made hill some 35 feet high. Just as they reach the top, they are uncoupled and start to coast down the other side of the "hump."

As the cars start down, the IBM system consults stored information and takes control of the routing of the car, automatically triggering the switches that send the rolling car to the proper track. On the average, this procedure is repeated 4,000 times each day as some 29 freight trains are assembled.

* * *

A computerized system that can handle the massive power demand of a steel mill or the many needs of a residential customer is controlling the flow of electrical energy into parts of five states.

The installation, anchored by an IBM 1800 data acquisition and control system, regulates electric power for 885,000 Allegheny Power System (APS) users in parts of Pennsylvania, West Virginia, Maryland, Virginia and Ohio.

This is believed to be one of the first digital computer systems in the nation controlling such a highly fluctuating power load for both industrial and residential use.

The computer-based system optimizes control of nearly 4-million kilowatss supplied by 44 generating units. APS subsidiaries serving a 29,500 square-mile-area are West Penn Power Co., with headquarters in Greensburg, Pa.; Monongahela Power Co. in Fairmont, W. Va., and The Potomac Edison Co. in Hagerstown, Md.

Located in the Charleroi Power Control Center, the computer continually matches generating unit outputs with customer needs, to help assure the proper amount of power for large and small users at all times.

For example, when residential customers turn on appliances, the associated increase in power is sent to the Power Control Center and input to the computer. The computer senses the increased power requirement and transmits impulses to the many generating units to raise their output.

The determination of length of impulse and the generating units to which it should be sent for maximum economy are controlled by one of more than 150 programs stored in the computer.

Homer McCarthy, supervisor of system operations engineering, said APS' 1800 system provides greater functional flexibility than previous analog equipment, responding to changes in power demand on a second-to-second basis, around the clock.

"A power control supervisor seated at a specially-designed console now has more information available for the many intricate programs in the computer to continuously control the system more reliably and economically than in the past," he said.

The computer calculates all of the customer load requirements and adds all generator outputs to determine if the totals match. If generation is above or below customer demand at any instant, the IBM system computes the amount of output change needed and automatically signals this information to the 44 generating units over communication lines.

To help insure foolproof operation, the APS computerized system alerts the power control supervisor to error conditions by flashing red alarm lights on the console, sounding a gong, and printing out a message on a typewriter-like terminal. In this manner, malfunctions such as exceeding transmission line limitations, frequency deviations, or loss of individual input from generators or tie-lines are noted.

APS-developed computer control also is being used to more fully integrate the operations of its affiliated companies with neighboring power systems.

The flexibility of the IBM 1800 aids APS in assuring adequate electrical energy on a continuous basis to its many customers, while providing that energy as economically as possible.

* * *

FREE PROGRAM LISTINGS IN SOFTWARE AGE

(Procedures and Practices)

COST . . . . There is no charge for computer programs listed in SOFTWARE AGE.

PUBLISHED . . . . . The listings will be published in the first available monthly issue of the magazine.

All listings will be re-published in semi-annual summary issues of SOFTWARE AGE during the months of the S.J.C.C. and F.J.C.C.

BROKERAGE . . . . SOFTWARE AGE does not broker computer programs nor act as an agent. As an unbiased third party, SOFTWARE AGE is acting, solely, as a pipe-line of communications between buyers and sellers of computer programs.

EVALUATION . . . . SOFTWARE AGE does not compare or evaluate computer programs. The degree of seller on-site support and the extent of buyer capability virtually make the evaluation of any program meaningless.

REQUIREMENTS . . . A prerequisite for any listing is that it be documented or supported to the extent of being operable by the buyer.

ANONYMITY . . . . A cross-coded reference number will be assigned to any program whose originator wishes to remain anonymous. Buyer inquiries will be forwarded to the originator. If the originator still prefers anonymity, buyer will be notified by SOFTWARE AGE.

GROUPING . . . . . Listings will be grouped according to keyword, title, mainframe application and compatability, industry, etc. As listings grow, more categories and definitions will be added.

software age

2211 FORDEM AVENUE
MADISON, WISCONSIN 53701

FREE PROGRAM LISTINGS IN SOFTWARE AGE

(REVERSE MUST BE COMPLETED)

SOFTWARE AGE FREE PROGRAM LISTING FORM

New Listing ☐
Revised Listing ☐
Program Wanted ☐
Broker Wanted ☐

This program is: Utility ☐ Systems ☐ Scientific ☐ Applications ☐ Engineering ☐ Management ☐ Other ☐

Title: | Number of Programs (if more than one)

Description: (Use reverse side if necessary)

Hardware Configuration:

Programming Language: | Documentation Provided:

On-Site Support/Debugging Available? Yes ☐ No ☐ Hours ☐ Guaranteed? Yes ☐ No ☐

Patent No. Copyrighted? Yes ☐ No ☐ | Number of Programs Distributed

Check Here for Cross-Coded Reference ⟶ ☐ ⟵ Check Here for Cross-Coded Reference

(REVERSE MUST BE COMPLETED)

(REVERSE MUST BE COMPLETED)

SOFTWARE AGE FREE PROGRAM LISTING FORM

New Listing ☐
Revised Listing ☐
Program Wanted ☐
Broker Wanted ☐

This program is: Utility ☐ Systems ☐ Scientific ☐ Applications ☐ Engineering ☐ Management ☐ Other ☐

Title: | Number of Programs (if more than one)

Description: (Use reverse side if necessary)

Hardware Configuration:

Programming Language: | Documentation Provided:

On-Site Support/Debugging Available? Yes ☐ No ☐ Hours ☐ Guaranteed? Yes ☐ No ☐

Patent No. Copyrighted? Yes ☐ No ☐ | Number of Programs Distributed

Check Here for Cross-Coded Reference ⟶ ☐ ⟵ Check Here for Cross-Coded Reference

(REVERSE MUST BE COMPLETED)

This program is: For Sale ☐ Lease ☐ Franchise ☐ | Price:

Name and Address of Program Owner ______________________________

Individual Contact ______________________ Title ______________________

Firm ______________________ Industry ______________________

Street Address ______________________________

City, State and Zip ______________________ Tel: (______)______________
(A.C.) (Number)

Additional Description (continued)		Additional Listing Forms Required ______

RETURN TO: PROGRAM LISTING EDITOR • SOFTWARE AGE • 2211 FORDEM AVE. • MADISON, WIS. 53701

This program is: For Sale ☐ Lease ☐ Franchise ☐ | Price:

Name and Address of Program Owner ______________________________

Individual Contact ______________________ Title ______________________

Firm ______________________ Industry ______________________

Street Address ______________________________

City, State and Zip ______________________ Tel: (______)______________
(A.C.) (Number)

Additional Description (continued)		Additional Listing Forms Required ______

RETURN TO: PROGRAM LISTING EDITOR • SOFTWARE AGE • 2211 FORDEM AVE. • MADISON, WIS. 53701

FREE PROGRAM LISTINGS

This marks the beginning of SOFTWARE AGE's Free Program Listings. These listings will be republished in appropriate issues. Each listing will be cross-referenced and will be grouped according to keyword, title, mainframe application and compatability, industry, etc. As listings grow, more categories and definitions will be added. We do request, however, that all forms be typed to ensure proper printing of your listing.

SYSTEM SOFTWARE

MACROS

BOCS N/A

Bisynchronous Oriented Communications System is a macro generated program designed to transmit tape or disk files over high speed lines in bisynchronous mode. The macro-generation allows the program to be self-tailoring to the constraints of the network and hardware configuration in which it is to operate. Broad flexibility is allowed in specifying the number of circuits to be controlled, the operating system, block size, etc. The program provides for statisical logging and error analysis. The operator is notified if line errors exceed a threshold value. Line utilization is (optionally) increased by extracting blanks in excess of two and restoring them at the receiving end. The program also has its own restart capability and allows for the retransmission of a single volume of a multi-volume file.

S/360 Mod 30 & up, 32K, OS/DOS, BAL

Technical Services
GT&E Data Services Corporation
502 Gettle Building
Fort Wayne, Ind. 46802

A0101

CONSOLE N/A

To write out on or receive from Console—230 bytes.

360–25 & up, DOS, BAL

Chester Lee
2508 Jackson Drive
Woodridge, Ill. 60515

A0102

PRINTER N/A

To write on syslst—75 bytes.

360–25 & up, DOS, BAL

Chester Lee
2508 Jackson Drive
Woodridge, Ill. 60515

A0103

BASE REGISTER N/A

Initialize base registers and set equates to registers—50 bytes max.

360–25 & up, DOS, BAL

Chester Lee
2508 Jackson Drive
Woodridge, Ill. 60515

A0104

DIGIT CHECK—MACROS N/A

Digit checking for account numbers for credit, debits, etc.—200 bytes.

360–25 & up, DOS, BAL

Chester Lee
2508 Jackson Drive
Woodridge, Ill. 60515

A0105

RELOCATE N/A

Relocate address constants—40 bytes.

S/360–25 & up, DOS, BAL

Chester Lee
2508 Jackson Drive
Woodridge, Ill. 60515

A0106

EQUATE N/A

Sets equates for registers—0 bytes.

S/360–25 & up, DOS, BAL

Chester Lee
2508 Jackson Drive
Woodridge, Ill. 60515

A0107

SUBROUTINES

ZERO-SUPPRESSION ON 1130 FORTRAN PRINTOUTS $50 each, $75 both

Greatly improves appearance of most tabular output by substituting blanks for all identifiable integer and positive or negative real zeros. Usage is via regular FORTRAN formatted WRITE statements. No change is made to any standard IBM-supplied routine. Can be added to any existing program by just one CALL statement. Suppression may be enabled and disabled at will anywhere in user's program. Requires about 80 (hex) words.

1130 with 1132 or 1403 Printer, BAL

Dr. Myron A. Calhoun
Private Consultant
420 Laurel Avenue
Menlo Park, Calif. 94025

A0108

NONE $25

System Macros to allow Assembly language subroutines to be written for PL/I programs, and to allow PL/1 written subroutines to be used by Assembly language programs. The interfaces are simple to master, and have proven very necessary in PL/I installations to maintain system efficiency for those operations not easily coded in PL/I. Conditions with sale.

360 OS, BAL

R. W. Shaw
Consultant in Data Processing
2221 Circle Place
Ottawa 8, Ontario, Can.

A0109

TABLE LOOK-UP SUBROUTINE $200

Very efficient and compact binary table search module. Returns a useable COBOL subscript and uses 272 bytes of core.

360–25 & up, BAL

P. D. Lamasney
Cybernetic Systems, Inc.
2275 Shattuck Avenue
Berkeley, Calif. 94704

A0110

REENTRY ROUTINE $25

This subroutine allows a COBOL program to access information in its own file description entry (either DCB or DTF) via its own Linkage Section.

360 OS or DOS, BAL

Dan Snyder
R. 4, Box 1023A
Christiansburg, Va. 24073

A0111

TRUE-TEST $95

This subprogram tests for true numeric in a PL/I program.

Any Configuration, PL/I

Frank Curcio
27 Saint Marks Avenue
Rockville Center, N.Y. 11570

A0112

DEEP/360 $225

DEEP/360 is a 3K subroutine which is linked into a problem program to inhibit "data exception" type program checks. It was developed for use in testing COBOL and Assembly Language programs where data check program aborts occur often. DEEP/360 is called only twice, once at program initialization and once at program termination. All logic spanned by these calls is protected against data exception program checks. DEEP/360 attempts to "correct" faulty data through the application of a few simple rules. Absolute addresses of instructions involved in any data error trapped are printed on the system console, and problem program execution proceeds uninterrrupted. An auxiliary package, OVRLAY2, permits usage of DEEP/360 in COBOL overlay programs. DEEP/360 and OVRLAY2 are for Disk Operating System use *only.*

IBM 360/30 or up, BAL

Maurice E. Scherer, Jr.,
Director of Applied Research
Macro Services Corporation
131 Tremont Street
Boston, Mass. 02111

A0113

COBOL ARRAY MOVE $150

This subroutine allows programmers to avoid subscripting when moving variable-length strings from arrays (tables) to working storage, or *vice-versa,* or from one array to another. Instead, the programmer declares array sizes once, initially, then specifies for every data transfer the array *address* (symbolic) and the *number* of characters to be moved. This routine is especially valuable where string manipulation is absolutely necessary in a COBOL program.

IBM 360/20 or above, BAL

M. E. Scherer, Jr.,
Director of Applied Research
Macro Services Corporation
131 Tremont Street
Boston, Mass. 02111

A0114

BENNETT DIRECT ACCESS METHOD $1,500

The program is a 677-byte subroutine which uses P.I.O.C.S. to directly access/360 disc records. Current direct access subroutines use at least 3500 bytes of computer memory, an overhead which cannot be afforded by many users. Not only is BDAM 81% shorter than current IBM-supplied direct access, but its P.I.O.C.S. makes it faster than IBM's L.I.O.C.S. approach.

360/DOS, BAL

Richard Bennett
Morphological Cybernetics
125 High Street
Ashland, Mass. 01721

A0115

FORTRAN EXTENSION PACKAGE $10–$95

The extension package is a collection of IBM 360 Assembler language (uses no macros) subroutines callable from FORTRAN or COBOL. They provide access to the useful machine instructions and other functions that are difficult or impossible to perform in high-order languages. Available are Move Characters, Compare Characters, Translate, Translate and test, String manipulation subroutines, and a great many others. Any of the subroutines may be used without loading the entire package. Extremely fast and efficient. May be separately or as a package. The entire package is fully guaranteed. A limited version of this package is available in FORTRAN IV.

360, BAL

SYCOM Computer Systems
344 South Division
Ann Arbor, Mich. 48108

A0116

XPAK/360 $100

XPAK/360 is a collection of IBM S/360 Assembler subroutines (no macros are used) callable from FORTRAN IV or COBOL. They provide access to the useful machine instructions and other functions that are expensive or difficult to perform in high-order languages. Available are Move Characters, Compare Characters, String manipulation subroutines, Translate, and many others. The code is extremely short and efficient, and is free of length limitations, etc. Detailed description on request.

360, BAL

PRS Associates
344 South Division, #420
Ann Arbor, Mich. 48108

A0117

JAMPAK/360 $50

JAMPAK/360 allows the user, with a single subroutine call, to compress data for output and restore it on input from any storage device, thereby effecting a very large savings in storage space. Savings in space of up to 60% for numeric data and 40% for alphanumeric data. The subroutines also allow the user to translate bit strings to byte strings and vice-versa. Detailed description on request.

360, BAL

SYCOM Computer Systems
344 South Division
Ann Arbor, Mich. 48108

A0118

SOFTWARE SYSTEMS

MASTER SHADOW I (A Remote Access, Message-Switching System) N/A

SHADOW is a highly-advanced communications software system for handling remote terminals under the MASTER Operating System on CDC[R] 3500 and 3300 Computer Systems. It serves primarily as a switching hub for communications on the CYBERNET[TM] network. This extension of operating software will handle a wide variety of remote job data strings. The SHADOW System will form an on-line communications link between 3300 and 6600 Centers or other 3300 Centers enabling a customer's MARC[TM] Terminal to have access to 3300 processing or 6600 processing as well as to the SHADE file management system on the 3300. SHADOW will provide truly automatic network processing by allowing jobs entering the system to be processed at other sited.

CDC 3300 (96K), COMPASS

A0119

SYSAID/360 $500

It functions in an OS environment, is called by application programs, and it provides the following functions: 1) logs time on and time off for job step (on console); 2) fetches parameters from JCL card (PARM) for use by job step; 3) fetches today's date from OS for use by job step; 4) passes completion codes to OS to be tested by JCL cards of subsequent steps (COND); 5) logs job step result on console ("good run"/ "bad run").

360, BAL

Gerald B. Bernstein
SBL Management Services
2082 East 58 Street
Brooklyn, N.Y. 11234

A0120

STORAGE TO STORAGE BIT SHIFT ROUTINE $25

This subroutine will shift up to 256 bytes from 1 to 7 bits left or right with one call. It can be supplied in a Macro format if desired.

360 OS or DOS, BAL

Dan Snyder
R. 4, Box 1023A
Christiansburg, Va. 24073

A0121

MASTER SHADE 1 N/A

SHADE (File Manager) provides a capability for interactive user-file manipulation, where the customer can create, maintain, and report his data file. A user may develop applications programs which utilize SHADE's file manipulating features. Under SHADOWS, SHADE is also a logical "terminal" allowing jobs and other files to originate from SHADE and be sent to any destination in the CYBERNET system. Similarly, user-program output may be received by SHADE from: 3300 MASTER, 6600 SCOPE, and other SHADE systems. SHADE lets the user create files off-line (on disk packs and tapes) or on-line (disk files and drums). This flexibility allows the user to update his files on a deferred or immediate basis and significantly control his costs.

CDC 3300 (96K Core Storage under MASTER), COMPASS

A0122

PLUS $1,900

Abstract or Narrative:
PLUS is a Program Library Update System for the secure and compact storage, easy maintenance, and efficient compilation of source language programs for data processing facilities. Programmers are relieved of a time consuming and error-prone clerical activity, and computer time is saved since several programs can be entered, changed, removed, and/or prepared for compilation at once, and at computer (not programmer) speed. The program also will generate a job stream file and job control setup for compiling or assembly, and handles test files and job control cards without initiating computer control operations. Unusual is the generation of a Library Index report, a table of contents of the program library. Program description, version number, modification number, date of last revision, language, author, number of statements, and number of changed statements are reported. A Report of Changes itemizes changed statements, providing a valuable historical record of all maintenance activity on a program. A Job Schedule Report, listing job control setups, is produced when this facility of the program is used. PLUS also reproduces all or portions of any source programs on the printer listing or on tape or punch files.

IBM 360/25 up, 32K, OS, DOS, TOS, RCA Spectra 70, COC 3100, COBOL

Cullinane Corporation
60 State Street
Boston, Mass. 02109

A0123

INDUSTRIAL INFORMATION'S RECORD MANAGEMENT SYSTEM $45,000

Proprietary Software Systems. The Record Management System (IIRMS) is an application-oriented file processing system for use by many diverse organizations. It provides the non-programmer with an interpretive language to carry on a dialogue, in English, for such file processing activities as file creation, file maintenance, file interlocation and generation of reports. To the programmer, IIRMS is a powerful subroutine for use in compiled programs to directly access mass storage devices, and other peripherals. References to 300 data files are possible from within a single program. IIRMS is fully tested, documented and backed by support services throughout the life of an installation.

IBM 360/50, 360/65; Univac 1106, 1108; RCA Spectra 70/45, 70/46; Burroughs 3500

John J. Rooney
Industrial Information, Inc.
500 Office Bldg., Executive Drive
Fort Washington, Pa. 19034

A0124

MAGIC COBOL SHORTHAND $8,000

Because of the number and variety of services provided by the product, the name chosen was MAGIC: Multipurpose and Generalized Interface to COBOL. The input to IMI/MAGIC is an abridged program consisting of intermixed COBOL program adhering to COBOL rules. Some of the functions of IMI/MAGIC are as follows: rigorous syntax checking—the syntax of each statement is checked and errors are diagnosed; optional recognition of COBOL word abbreviations; substitution of one or more syntactic elements to replace a user-declared synonym; allowance for common errors of syntax or punctuation; monitoring or inhibiting of user-designated words; and assistance in evolution to a higher COBOL dialect.

IBM S/360, 32K, COBOL

Mr. Robert P. Bell,
Director, Applications Development
Information Management Incorporated
447 Battery Street
San Francisco, Calif. 94114

A0125

CRAM/360 $6,200

Data can now be stored on ordinary file media in far less space than was normally required, by using CRAM/360 Compression, Retrieval and Maintenance. CRAM/360 is capable of reducing the disk storage requirements for most 360's by an average of 40%. It is easily adaptable for other manufacturers' machines. It requires no maintenance at any time.

The system stores data in a special direct access library. The data is fragmented to optimize retrieval, as well as being compressed on storage. When the source data is scanned, all blanks are deleted and only valid data is stored. Data is automatically decompressed upon retrieval. Once data has been accessed, under one of the 1,125 *reusable* index pointers, it is stored on specific tracks assigned to each member of the fragment. When data is deleted from the library, its index pointers and associated tracks are freed for re-use.

Various options are available for the user. He can add new source data, retrieve old data unchanged, update existing file data, delete existing members, list the source members selectively, and list the contents of the indexes. Specific applications suggested for this system include: program source language files, mailing lists, inventory control files, and other large-record indexed files where data storage becomes a significant factor.

All Systems Compatible with OS or DOS BAL

Charles Salvaggio or Jack Hirschfield
Computer Interactions, Inc.
425 Northern Boulevard
Great Neck, N.Y. 11021

A0126

BASIC TO FORTRAN TRANSLATOR $3,000

Translates Basic (all levels) to FORTRAN (G, H and WATFOR). Translation is 100% on commonly used basics. Offered as service on time-sharing environment for sir charge only.

360, 128K up, APL/I

Christopher D. Iles
International Conversion Systems
370 Lexington Avenue
New York, N.Y. 10017

A0127

ARTIC (Associometrics Remote Terminal Inquiry Control System) $14,500

ARTIC provides users of DOS with an efficient approach to satisfying the software requirements of a teleprocessing system, while allowing the programming of inquiry applications to be written in a high level language such as COBOL or PLI. ARTIC supports teletype units audio response, cathode ray tubes, and the IBM 2740; facilitating the multi-tasking capabilities of DOS. ARTIC makes teleprocessing programming so simple, that it can be handled by a junior programmer. Includes ADAM (Associometrics Data Management System).

Minimum: 360/30–32K, ALC

R. L. LaRose, Vice President
Associometrics Incorporated
5531 Dyer Street
Dallas, Texas

A0128

ADAM (Associometrics Data Management System) $4,500

ADAM, the data management sub-system of ARTIC (Associometrics Remote Terminal Inquiry Control System), supports index sequential and direct access file organization; facilitating the multi-tasking capabilities of DOS. ADAM, when used in either a teleprocessing or non-teleprocessing environment, conserves core usage through common work areas, input/output areas, randomizing routine, etc. ADAM is of special benefit to the user who is in an overlay situation, on the verge of having to upgrade his central processor because of memory requirements or programming in COBOL.

Minimum: 360/30–32K, ALC

R. L. LaRose, Vice President
Associometrics Incorporated
5531 Dyer Street
Dallas, Texas

A0129

TOL (Transactions on Line) $50,000

TOL is a hardword software system that front ends System 360. Its a non-programmer-oriented time sharing system. TOL handles a large number of interactive terminal within a reentrant multi-programmed environment. TOL provides a low overhead and high level of efficiency in use of System 360. Applications for TOL include data entry and validation, order entry, file update, message routine.

360–30 and up, BAL

Ronald Buren
Software Methods Inc.
10 East 40 Street
New York, N.Y.

A0130

TELE-COBOL $15,000

A system consisting of access methods, line handlers, and utility programs which allow an OS/360 user to receive and transmit data to remote terminals while staying within COBOL (PL/I and FORTRAN support are also available). The user writing in COBOL includes a "CALL INPUT" statement to process data received from a remote terminal and/or a "CALL OUTPUT" statement when he desires to transmit data. The handlers will queue input from the terminals on a disk and notify the operator when transmissions are received and which user programs they are being delivered to. The appropriate COBOL program will be called into another partition to process all the data accumulated for it. Also, there are three utility programs in the system which create, purge and dump the files, respectively.

Requires a 27XX, and 30K of an OS/360–MFT System, BAL

Stephen B. Hesse
Complex Systems Inc.
122 East 42nd Street
New York, N.Y. 10017

A0131

PSI–VALET LIBRARY SYSTEM $2,880

PSI–VALET is a totally integrated library storage, maintenance and management security system. Operating under OS or DOS on any 360 model 25 or larger, the three phase system requires 22K of memory and will run in any 22K batch partition. Any program (source or object), job control stream or card image data file may be stored, retrieved and maintained. PSI–VALET, by creating a central direct access disk or data cell library, will eliminate card deck storage, will increase programmer and operator efficiency, project control and management security. Job stream generation with job control will better utilize CPU time.

PSI–VALET is designed for direct access storage devices. Storage is random and access is direct. File reorganization is never required. Space is conserved by compressing all additions to the library. Retrievals are expanded to their original condition. All additions to the library can be sequence checked. Any library file can be transferred from one type storage device to another and be re-entered with the proper respective format. Special PSI–VALET features will aid the teleprocesser.

The PSI–VALET library system will generate a job stream with job control to compile, assemble, link edit, etc. All or any portion of the library may be reproduced on tape, printer, punch or disk. Production status programs or files cannot be modified or deleted. Modifications can be made of copied versions of production programs. Library file deletion requires management intervention. PSI–VALET systemizes the activities of all programmers. On line-real time programming maintenance is possible. PSI–VALET gives management control by producing a library directory listing of all modifications and library activity, the date of activity and the status and modification number of each program and file. For management security, special back-up tapes are produced which can be restored in minutes in the event of a disaster.

IBM 360 Model 25 or larger 22K, DOS & OS

Edward F. Loritz,
Vice President–Director of Marketing
Pansophic Systems Incorporated
230 East Roosevelt Road
Lombard, Ill. 60148

A0132

SOFTWARE GENERAL

EXEC 1 $3,500

EXEC 1 is a real-time, time-sharing monitor for the D.E.C. PDP-12. It consists of three basic function modules: A keyboard Monitor, a File Handler, and an I/O Handler for all standard PDP-12 peripheral devices. EXEC 1 is intended to be used as the base for a multi-job production time-sharing system where a relatively small number of well-behaved users require simultaneous access to the PDP-12.

PDP-12A with Real-Time Clock and 8K, PLAP Compatible Assembly Language

Gene Sengstock, Vice-President
Agrippa–Ord Corporation
394 Massachusetts Avenue
South Acton, Mass. 01720

A0133

PL/I (A) CARD COMPILER $36/mo

A compiler for a PL/I scientific subset on the System/3 of IBM.

System/3 8K with RPG/II, PL/I String Subset

Jay Woods
Woods Associates
Box 5894
Stanford, Calif. 94305

A0134

TESTGEN—The COBOL Test Data Generator $3,000

TESTGEN is the test data generator for use with all COBOL computer programs.
TESTGEN generates the right amount of data, giving sums of numbers to be checked against the program's answers. Most or all of TESTGEN's information comes from the source program itself. Alphabetic, alphanumeric data are generated. You have the use of optional control cards to expand and restrict character sets, to have sequenced rather than random data, and to give totals and sub-totals of numeric fields based on control breaks in other fields. Subtotals and totals may be computed for the product of two fields even though the product itself does not appear in the test data. Using TESTGEN is as simple as making an ordinary compilation. Just place TESTGEN and the COBOL source program together in the input stream.
TESTGEN will . . . scan each File Description analyzing the data record formats and printing any source errors recognized.
TESTGEN will . . . write onto cards, tape or disk records formatted according to the specifications of the File Description in the source program.
TESTGEN will . . . print listings and summary totals of the test data produced.

OS/360 and DOS/360, BAL

James E. Serwer
Interface Systems, Inc.
330 Municipal Court Building
Ann Arbor, Mich. 48108

A0135

FASTBALL $2,500–$3,000

FASTBALL creates a complete and efficient COBOL program ready for compilation. All source statements are self-descriptive and easy to modify.

FASTBALL serves as the starting point of all COBOL program writing.

Report Generation, File Copies, File Creates, Indexed Sequential File Reorganization, Spools Records, Reblocking, Change Record Sizes.

File Updates, that is moves, adds, subtracts, and blanking is accomplished by use of the same one sheet transmittal. Two sheets are required if more than fifteen input or output data-names are necessary. Not including fillers.

FASTBALL is a stacked job program. Many programs may be consecutively generated. The input and output sides of the transmittal can be used independently for Data Division generation.

FASTBALL with file updates is available for 64K computers as one model. For 32K computers it is available in two modules.

Four levels of totals are supported. Five elements may be totaled.

360 32K, COBOL

Richard C. Brown
Brown Bros. Enterprises
509 Fuller N.E.
Grand Rapids, Mich. 49502

A0136

CULPRIT REPORT GENERATOR—UTILITY TYPE $10,000/13 yr.

"CULPRIT" is a powerful generalized report generator which retrieves and manipulates data and produces reports from existing data files as directed by parameter cards. No compiling is necessary. The program is highly efficient and many reports can be produced with a single file pass. Thus CULPRIT not only handles one-time reports with 1/10 to 1/40 of the programming time normally required, it can replace the report-producing part of all your present and future programs. Of course, the versatility of the program enhances its use for such purposes. Easy-to-use specification forms direct the selection of records, any computations to be performed, sequencing, and formatting of the output. Constants can be introduced and data manipulated to produce new fields for sorting and reporting. A valuable feature is the program's multi-line output capability, useful for address labels, notices, etc. Other options include header variables, multiple-lines in headers, detail, and totals, creation of new fields, calculations and counts, multiple-field sorting with specified break totals, separately-specifiable total lines with calculation ability on the totals, use of mnemonics for working fields, use of multiple input files, and many others. Many default options permit CULPRIT to be used by non-programmers. Output may be printer, punched cards, tape, or disk.

IBM 360–30 65K, 8 RCA Spectra 70, BAL

Cullinane Corporation
60 State Street
Boston, Mass. 02109

A0137

FILE MAINTENANCE

REAL-TIME FILE MAINTENANCE N/A

Provides user with a real-time file system which can update the file while still on-line with the control functions.

Real-Time with On-Line Disk File, BAL

Patrick J. Williams
ADC Systems
33 East McMillan
Cincinnati, Ohio 45219

A0138

MISCELLANEOUS

ARMsim(TM) $3,500

1401 Simulator for use with DEC PDP/8. Can run as background commercial processing function in Real-Time dedicated systems.

All PDP/8 Series, IBM 1401

Stanley G. King
1510 Merkley Avenue
West Sacramento, Calif. 95691

A0139

DYNAMIC MULTI-TASKING SYSTEM $7,800

DMTS is a generalized, flexible and efficient multi-tasking framework that allows users to run independent production programs concurrently as subtasks in any DOS partition. Throughput is thereby increased, both for cpu and peripherals. The user never becomes concerned with DOS multi-tasking macros since these are all programmed into the DMTS package in a generalized form. User programs may be written in COBOL, FORTRAN, or BAL. A minimum of modification is required to run user programs under DMTS.

360/DOS, BAL

R. S. Carter, President
Western Systems, Inc.
467 East 3rd South, Suite 205
Salt Lake City, Utah 84103

A0140

DYNAMIC LOADER $100

Because of the built-in dynamic load, this program saves relink-edit of programs calling subroutines which have been altered. This feature also insures that all calling programs will link to the most current version.
This program enables subroutine calls in input and output procedures of COBOL programs using the Sort verb.
It can be used with ALC, COBOL, PL/I.

360–30 and up, BAL

Robert Woodward
Comp–Services
1106–8 East Livingston
Columbus, Ohio 43205

A0141

MULTI-TASKING MONITOR CONTROL PROGRAM N/A

Monitor Program will load and run up to 9 self-relocating programs (core and I/O equipment allowing) in a batched job partition. Interrupt by operator allows programs to be initiated by operator through console. Can terminate any job running, assign devices through console. Program itself is self-relocating and is approximately 4K. For DOS.

S/360 with multi-tasking feature on supervisor—Rel 19 or later, BAL

Chester Lee
2508 Jackson Drive
Woodridge, Ill. 60515

A0142

UTILITY PROGRAMS

UTILITY FILE MAINTENANCE $650

The file maintenance program is designed to correct or delete specified records or blocks on a tape or disk file. No compilation required. A series of parameter cards are available to find and correct or delete any record or records on a tape or disk file. Version I operation since June, 1967.

360–40 Card, Tape & Disk, BAL, DOS/TOS

David H. Thornton
931 Hillview Drive
Cary, N.C. 27511

A0143

"AWAY" **$500**

A source library system designed to eliminate the need to maintain source programs on punched cards. Works with COBOL, ASSEMBLY, & RPG. Provides complete documentation of all changes made to programs. Saves programmer and computer time. Provides user selected options for varying functions to be performed. Automatically generates Job Control cards. Manual and complete operating instructions included.

IBM 360/25 and up, COBOL

Irwin Taranto
Taranto & Associates
P.O. Box 6073
San Rafael, Calif. 94903

A0144

MULTI-MIX (General Purpose Utility Programs) **$2,000**

A multiprogrammer, stand-alone system consisting of a complete set of general purpose utility programs. Supports magnetic tape, paper tape, card ready, punch and up to two printers. All programs run independently and asynchronous to each other. No control cards required.

IBM 360 16K and up, BAL

Stephen B. Hesse
Complex Systems Inc.
122 East 42nd Street
New York, N.Y. 10017

A0145

JAMPAK **$250**

JAMPAK allows the user, with a single subroutine call, to compress data for output to a storage device and to decompress and retrieve the data on input from a storage device, thereby effecting a very large savings in storage space. Savings in space up to 90%. Also allows user to translate from bit strings to byte or word strings and vice-versa.

Any Configuration for FORTRAN IV Version, S/360 for BAS Version, FORTRAN IV or BAL

PRS Associates
344 South Division, #420
Ann Arbor, Mich. 48108

A0146

OVERLAY PROGRAMMING **$800**

Complete documentation including tested program examples. DOS/TOS job control set-up for testing your overlay structure, cataloging the structure and executing the cataloged structure.

360, ALC and COBOL

Joel Shapiro
1001 Spring Street, Apt. 120
Silver Spring, Md. 20910

A0147

SUPERIOR INTERPRETING DISPLAY DUMP FOR THE SYSTEM/360 **$160**

This is a core memory dump program which interprets the data in memory. Instructions, CCW's, and strings of alphanumeric characters are identified and clearly displayed. Floating point numbers are displayed in hexadecimal and converted to decimal in "E" or "F" format depending on the magnitude of the exponent. Integers are displayed in both hexadecimal and decimal. This program was specifically designed to replace the Fortran FDUMP routine but it can also be called by programs written in other languages.

360/25 and up, BAL

W. P. Fisher
Superior Computer Systems
P.O. Box 2971
New York, N.Y. 10001

A0148

COBOL CROSS REFERENCE PROGRAMS **$197**

The COBOL Cross Reference Programs provide a listing of the statements in a COBOL source deck and a cross reference listing of the data names. In addition, any deck may be reproduced and/or listed. A COBOL source deck may be reproduced, resequenced and/or given an identification code.

IBM 360, DOS, COBOL D

Richard K. Schmitt, II
National Water Lift Company
2220 Palmer Avenue
Kalamazoo, Mich. 49001

A0149

FORTRAN XREF GENERATOR **$10**

As an introduction to SYCOM and to aid in developing a customer list, we are offering a full FORTRAN IV cross-reference map generator for a nominal $10 handling charge. The program is complete and fully guaranteed. It is written entirely in FORTRAN IV and produces an ordered list of each variable name, and its type, length, where defined and a list of all statement numbers in which the variable was used.

Any Configuration, FORTRAN IV (Entirely)

SYCOM Computer Systems
344 South Division
Ann Arbor, Mich. 48108

A0150

FORTRAN FORMATR **$150**

FORTRAN FORMATR—A Fortran Format Conversion Program allows the user to perform FORTRAN-type format conversion without I/O and with full error recovery. With a single subroutine call the user can specify EBCDIC-to-binary or binary-to-EBCDIC format conversion on an entire list by passing a standard

For more information, circle No. 11 on the Reader Service Card

FORTRAN-IV format statement. FORTRAN FORMATR accepts A,I,F,E,X,H, and T format terms in a standard FORTRAN format statement. Any number of levels of embedded parentheses and mixed terms allowed, and any conversion subroutine may be used separately if desired. IBM S/360 users may request FORMATR-360 which has all of the above features but operates faster (using 360 assembler language) and allows "Z" hexadecimal and literal field conversions.

Any Configuration, FORTRAN IV (Entirely)

SYCOM Computer Systems
344 South Division
Ann Arbor, Mich. 48108

A0151

COBOL TDG TEST DATA GENERATOR $9,000

Test Data Generator automatically generates all the input files required to check out COBOL programs. Eliminates laborious and repetitive manual preparation of data for testing programs. Input data is created using the Data Division of the COBOL source program and English-like control cards. This data is then used as the foundation for testing the logic of the COBOL program. Test data inputs can be retained in the form generated by TDG for repetitive use in testing the procedural steps of the programs. Programmers can specify as many single and/or multiunit input files, in any combination, as may be required to thoroughly and properly check out a program. Different record types can be defined for a file and the number and order of records of each type can be specified, either explicitly or implicitly. Distribution of records and fields within records can be controlled. Data mixes, analogous to real-life data, can be incorporated to test and retest the logic of a new or recently modified program. TDG permits users to cluster values of an item around a specific point in a range, roll or shift values within a field, and specify random values, fixed literal values, or values that are algebraically dependent on the value of another field. The user can also specify ascending or descending sequences of values for a field with the ability to vary the increment or the sign, as well as repeat the sequence.

Any Computer, 64K, COBOL

Mr. Robert P. Bell,
Director, Applications Development
Information Management Incorporated
447 Battery Street
San Francisco, Calif. 94111

A0152

THE AUSTIN S/360 LIOCS SIMULATION SYSTEM $5,875

The Austin Simulation System allows third generation S/360 programs under DOS to access 1400 compatibility disk files. The System allows the user to immediately write S/360 programs which will not have to be altered once the file is converted to a native S/360 file. Since both the S/360 programs and the unconverted 1400 programs can access the same 1400 compatibility disk file, the 1400 to S/360 conversion can take place orderly, one program at a time, and once all the 1400 programs have been converted, the system will also allow the user to physically convert the disk file in a matter of minutes.

The system is easy to use, using keyword macros to describe the 1400 file and the S/360 file to which the user will eventually convert. The Simulation System can make a 1400 Consecutive or Control Sequential file act like a S/360 Sequential or Index-Sequential file in any combination, plus allow 1400 random files to act like S/360 direct access files. Because the file definition is in the Austin module and not the problem program, the problem program will not have to be recomplied when the compatibility file is converted to the S/360 native file.

S/360, DOS, BAL

David J. Hayes
Austin Systems Company, Inc.
87 Terrace Hall Avenue
Burlington, Mass. 01803

A0153

BIT-MANIPULATION FUNCTIONS $5 each

Gives FORTRAN programmers full range of logical bit-manipulation normally reserved for assembly language users. 16-bit logical AND, OR, EXCLUSIVE OR, SHIFT RIGHT LOGICAL & ARITHMETIC, SHIFT LEFT LOGICAL, and CONSOLE SWITCH input are available. It is no longer necessary to multiply/divide by powers of 2 to perform shifting. Packing/unpacking of internal tables may be done with ease. Money returned if sample program supplied does not run correctly. Requires about 40 (hex) words total.

IBM-1130, BAL

Dr. Myron A. Calhoun
Private Consultant
420 Laurel Avenue
Menlo Park, Calif. 94025

A0154

ALLEGRO (Linear Programming System) V.4.2 N/A

ALLEGRO is an extremely fast linear programming (LP) code, capable of solving problems of up to 11,000 constraints, written for the CDC 3600 computer. A 900 x 3000 problem with 11,000 non-zero element can be run in only 30 minutes using ALLEGRO. A selection of solution options is available including Primal, which is standard solution algorithm, as well as dual solution. ALLEGRO has flexible input options to use: the SHARE card format; the more compact ALLEGRO format; or to load and revise all or parts of an existing Matrix from storage. Several output report options are available from ALLEGRO including: constraint summary; variable summary; sensitivity analysis; matrix display and statistical analysis.

CDC 3600 (65K), FORTRAN/COMPASS

Robert O. Young
CDC Data Service Division
4550 West 77th Street
Minneapolis, Minn. 55435

A0155

TESTING AND DEBUGGING

GENERAL PURPOSE SIMULATION SYSTEM (R/GPSS) N/A

Remote/General Purpose Simulation System, this system allows the user to describe his "system" with a simplified block diagram, enter the description via a TS terminal, according to the language rules of R/GPSS, and schedule a processing run on his model. The user then selectively reviews the "performance" of the system, makes changes or additions to the model and repeats the simulation run as desired.

360/65, GPSS

Joel Leichter
Princeton Time Sharing Services, Inc.
U.S. Highway No. 1
Princeton, N.J. 08540

A0156

NOVASM N/A

NOVASM simulates all NOVA computer operations and provides comprehensive debugging and simulation control commands to aid in the rapid development of NOVA programs. An I/O device simulates software interface to correlate any NOVASM input/output device with any PDP-10 device.

PDP-10, MACRO 10, FORTRAN

Mr. Henry P. Semmelhark, Pres.
Comptek Research, Inc.
4548 Main Street
Buffalo, N.Y. 14226

A0157

DISK/TAPE DUMP $100

Hexadecimal and character dump of any data set on any storage device under OS/360. Utility will dump indexed sequential and sequential data sets from DASD or tape storage devices. Also prints fixed or variable length records, blocked or unblockod. No special control cards necessary. Any number of records may be printed, either from the beginning or end of the data set. Every nth record may be 'parm' field of the 'exec' card. Program prints approx. 3000 characters per second, operating in a 30K partition. The programs can be compiled under COBOL 'E' or COBOL 'F' under MFT or MVT.

360—40, BAL and COBOL

Allen Sigal
Hennepin County Data Processing
1116 Flour Exchange Building
Minneapolis, Minn. 55416

A0158

ITRACE $160

ITRACE is a routine which assists in the debugging of System/360 program by following a program instruction by instruction as it is being executed, printing for each instruction a record of any changes in memory or in registers. Integer numbers are shown in hexadecimal and in decimal. Floating point numbers are shown in hexadecimal and in "E" or "F" format depending on the magnitude of the exponent. Alphanumeric fields are shown as such.

360/25 and up, BAL

W. P. Fisher
Superior Computer Systems
P.O. Box 2971
New York, N.Y. 10001

A0159

CIP (Console Interface Program) $1,000

CIP is a debugging program that allows automatic debugging of multiple errors during program execution. Just as a COBOL compiler allows a user to catch multiple compilation errors, CIP allows users to catch multiple errors during execution. During program testing, the system intercepts any operator, program check, interval timers or unrecoverable error interrupt and automatically displays the module name and the interrupt location in both absolute and relative values. It can be used in unit testing of program modules, system testing, and recovering from software errors in the initial stages of productive system operation.

360-30 and up, RCA Spectra 70, BAL

Jerald Greenberg
Worldwide Computer Services Inc.
280 North Central Park Avenue
Hartsdale, N.Y. 10530

A0160

BALFIX $2,000

BALFIX system provides on-line debugging facility for IBM/360 Assembler programs.

360/30 DOS and up, 32K, BAL

Charles B. Wang
PDA Systems, Inc.
12 East 86th Street
New York, N.Y. 10028

A0161

360 BAL PREPROCESSOR $400

Program uses a Honeywell 200 Series computer to diagnose assembler/keypunch errors in a 360 BAL program, including multiple tags, undefined operands, illegal parameters and mispunched op codes. Use in single or batch processing environment; about 300-cdpm can be processed. Ideal for use in educational environment where a less costly H200 supplants the 360 in diagnosing typical student errors before running on a 360.

H200 w/ Advanced Program instructs; 2 tapes, 16K, EASYCODER

Warren Kendrick
Brown Institute
3123 East Lake Street
Minneapolis, Minn. 55406

A0162

DATA HANDLING

GRAD $7,000

GRAD (Generalized Remote Access Data Base) is a selective retrieval and update system written to function in a batch or time sharing environment. The system is interactive with formatting and totaling capabilities available. Up to 3 different data bases can be handled by the system.

360, B5500, RCA, COBOL

Ronald Buren
Software Methods Inc.
10 East 40th
New York, N.Y.

A0163

DATA RETRIEVAL $1,500

This data retrieval program has the unique feature of generating its own optimized machine instructions from data in the parameter cards prior to execution of the search and retrieval phase. Execution time is thereby substantially reduced by the elimination of control word testing and searching for data field locations in the tape record. The generated coding is "tailor made" for the search being performed. Parameter cards are simple in layout and easy to use. Parameters specify the following:

1. Record length and blocking factor of tape record.
2. Data field locations and lengths and/or literals to be used in the compares, whether compare is to be numeric or alphanumeric, the conditional result to test for, and the "AND" and "OR" connectors needed to establish the logical linkage of the search parameters.
3. Data field locations and lengths to print for records that are selected during the search and retrieval.

The search parameters can be specified by use of any number of data fields, literals, compares, and connectors. Each compare result is specified by mnemonic names for conditional branches, as follows: EQ, NE, GT, LT, GE, LE.

IBM 1401, 12 or 16K, 1 Tape (or Emulator), 1401 Autocoder

Gordon A. Walker
P.O. Box 5244
Irving, Texas 75060

A0164

UPGRADE 1400 TO COBOL TRANSLATOR $24,000

A system to translate IBM 1401-1440-1460 source or object code to COBOL for IBM-360 OS or DOS, RCA SPECTRA-70, or BURROUGHS B5500, to clean compile for 60¢ per 1400 source card, to clean execution on a bid basis.

A series of routine manual procedures used before and after the machine translation assures a 97% to 100% translation when a clean compile is first obtained. In addition to being used as a service, the system is for sale or lease, and available for franchise.

1401/16K/5 Tapes, 1401 AUTOCODER

Mr. Robert P. Bell,
Director, Applications Development
Information Management Incorporated
447 Battery Street
San Francisco, Calif. 94111

A0165

SOLIS (Symbionics On Line Information System) $25,000

This program will run On Line Information Systems using CRT or typewriter terminals. By the completion of 5 SOLIS forms almost any information system can be running On Line within a few days of systems analysis work being complete, no programming being necessary. Useful for general ledger, accounts receivable, sales analysis, etc. Also for information retrieval, hospital information systems, educational information systems, reservation (theatre, etc.) systems, computer assisted instruction. On Line inventory.

Any Computer with Disc, Terminal Capability and 64K Byte Memory, FORTRAN IV

B. A. Hodson, President
Symbionics Systems LTD
550 Berry Street
Winnipeg 21, Canada

A0166

UTILITY EDIT $925

The edit program is designed to accomplish two primary objectives. First, to edit a variety of card, tape, or disk formats and designated fields therein for correctness of data. Second to write to tape or disk fixed or variable length records blocked as required, and formatted as desired. Ten accumulators are available for hash totals. Thirty-five macros are available to perform a wide variety of edits. The edit was designed to create files for production runs but programmers will also use the edit to create test data master files. Version I became operational in December, 1966. No compilation is required. Macros entered on parameter cards.

360/40 Card, Tape & Disk, 360 BAL for DOS/TOS

David H. Thornton
931 Hillview Drive
Cary, N.C. 27511

A0167

AUTOCODER TO COBOL (R/Autocope) N/A

A version of ACCAP, pre and post processors which allows users to convert AUTOCODER to COBOL programs through the facilities of a typewriter-like time sharing terminal.

IBM 360/65, COBOL

Joel Leichter, Vice President
Princeton Time Sharing Services, Inc.
U.S. Highway No. 1
Princeton, N.J. 08540

A0168

PRO/TEST: THE PROFESSIONAL TEST DATA GENERATOR N/A

Can generate any file on tape, disk, or card with fixed or variable length records, blocked or unblocked, of ascending or descending sequence. Field values can be random, constant, or sequential. Fields can be summarized. Field values can be generated depending on the values of other fields. Files can be generated with up to nine levels of control breaks. The PRO/TEST forms utilize free form operand coding with extensive parameter defaulting. The PRO/TEST reports reflect all parameter errors, provide an analysis of the file to be generated, and print the complete or selected portions of the file.

360, 32K DOS or OS; H200 16K

Leslie B. Conklin,
Vice President—Proprietary Systems
Synergetics Corporation
Second Avenue
Burlington, Mass. 01803

A0169

UPDATE & RESEQUENCE COBOL SOURCE DECKS ON TAPE $1,200

This utility program can do any combination of the following:

1. Resequence by 10 or 100.
2. Change the field ID.
3. Modify, delete or add to the existing source deck.

DOS/TOS 360, ALC

Mr. Joel Sphairo
1001 Spring Street, Apt. 120
Silver Spring, Md. 20910

A0170

SIMULATOR, MULTI-FUNCTION UNIT-RECORD $100

SIMUR was designed to eliminate the need for a unit-record reproducer, various stand alone utilities, and user written specility programs in a Third Generation, multiprogramming environment. The functions include selectively reproducing and gang-punching, resequencing and reidentifying all 360 source program decks, 80/80 listing, reproducing and counting under under DOS program control. The program runs in 6K and is self-relocating.

360, 25 and up, BAL

Robert C. Periatt
Computer Analysts, Inc.
4447 Talmadge Road
Toledo, Ohio 43623

A0171

BIT/BYTE CONVERSION SUBROUTINE $50

Particularly useful for data compression, allows high-level language programs to inspect bits, etc., 202 bytes.

360/25 and up, BAL

Mr. R. D. Lamasney
Cybernetic Systems, Inc.
2275 Shattuck Avenue
Berkeley, Calif. 94704

A0172

COST ACCOUNTING

CONSTRUCTION COMPANY COST SYSTEM N/A

The system is primarily designed for electrical contractors but may be adapted to any building trades. Budget information, purchase orders, invoices and labor data are inputed. Outputs include labor and material status, comparisons to budget and projected completion costs by as many as 50 phases per job. In addition this data is reported in summary by job and shows profit earned, receivables and billing data and all information needed to satisfy government standards for percentage of completion method of reporting profits.

Honeywell 120 16K, 4 Tapes, COBOL D

Morton Tuckman, Vice President
Software Service Division,
International Computer Corporation
6806 Trexler Road
Lanham, Md. 20081

A0173

COST ALLOCATION SYSTEM $2,000

This program reads and stores a permanent allocation table and uses this table to allocate input direct cost and volume items to any number of line entries for the generated report. Each of the allocated entries in turn is allocated to other line entries by repeated scans of the table until all allocations have been completed for each input item. After a sort/merge, these direct and allocated cost and volume records are then summarized and a report printed of this month, last month, and year-to-date cost and volume data, along with cost center and line entry identification (carried on Master File record). File maintenance (add, change, delete), as well as file updating, is performed concurrently with report printing. Flexibility is achieved in report printing by use of Master File item codes to perform sign reversal, accumulation to a specified level, selected accumulator print and roll totals, and calculation of cost/quality ratios. This system was designed for cement plant cost allocation, but can be readily used without program change for any operation where "cost centers" can be logically defined.

IBM 1401—16K, 4 tapes (or IBM 360 with Emulator), 1401 Autocoder

Gordon A. Walker
P.O. Box 5244
Irving, Texas 75060

A0174

DOCUMENTATION

MADFLOW Free

Automatically draws a flowchart of a program written in MAD.

IBM 7094 and UNIVAC 1108, MAD

Dr. M. D. Abrams
Dept. of Electrical Engineering
University of Maryland
College Park, Md. 20742

A0175

DETAP (Decision Table to COBOL Processor) $14,500

DETAP converts COBOL Decision Tables to COBOL source code acceptable to the COBOL compiler. It produces a totally-documented COBOL program, with every Decision Table included as a COBOL NOTE to ease the difficult task of program debugging. DETAP processes limited entry and extended entry; optimizes for computer running time or core; allows as many as 4 stub continuations and 4 rule continuations; optimizes common sequences of actions; enables transactions to be traced through a decision table; checks all tables for completeness, contradictions and redundancies; and provides formatted decision tables for easier reading.

IBM 360 with 32K, COBOL

Mr. Robert P. Bell,
Director, Applications Development
Information Management Incorporated
447 Battery Street
San Francisco, Calif. 94111

A0176

ENGINEERING

LEHIGH ANALOG SIMULATOR (LEANS) $100

This program simulates the analog computer on a digital machine. LEANS supplies a full array of summers, multipliers, integrators and nonlinear elements.

1130, 16K, Disk, FORTRAN IV

J. P. Schwar
Lafayette College
Alumni Hall of Engineering
Easton, Pa. 18042

A0177

CRITICAL VALUE SEARCH PROGRAM—EXTVAL $250

This program will perform a random or steepest descent search on arbitrary functions of many variables. Output provides sensitivity calculations and worst case values. The user provides the routine for evaluating and defining his particular function.

FORTRAN

Robert E. March
Software Products Co.
3409 St. Joseph Avenue
Berrien Springs, Mich. 49103

A0178

NON-SYMMETRICALLY LOADED AXISYMMETRIC AND PLANE STRESS/STRAIN FINITE ELEMENTS $500

This program uses exactly integrated, linear strain, trapezoidal elements for either non-symmetrically loaded axisymmetric or plane stress/strain problems. The program contains the equations needed in order to automatically compute all possible accelerations of a rotating system accelerating through space. Hence, "gyroscopic" stresses and deflections may be easily determined. The program also features simplified and semi-automated input.

Any 32 K (or greater) Machine, FORTRAN

Dr. Arthur Wilson
American British Consultants
2857 East Beryl Avenue
Phoenix, Ariz. 85028

A0179

REAL-TIME MULTIPROCESSING TACTICAL EXECUTIVE SYSTEM N/A

Executive module for the scheduling and monitoring of real-time tasks related to shipboard command and control systems, which utilize single and/or multiprocessor computer complexes. Engineering designs available for single processor, multiprocessor and multiprogramming versions.

Shipboard Tactical Configurations (i.e., NTDS, DX, ASMS), BAL or Higher Level

Mr. William G. Phillips, Jr.
Steiner and Associates
3320 Kemper Street
San Diego, Calif. 92110

A0180

SHIP CROSS CURVES OF STABILITY $250/run

From an input of stations and transverse radii, this program produces output every 15 degrees of transverse inclination. Volume, tons, GZ, VCB, and righting moment are calculated.

IBM 1130, 8K, FORTRAN IV

N. Harper
Scotian Design Services
Box 726
Port Neches, Texas 77651

A0181

STRES1 N/A

STRES1 is a general purpose computer program which analyzes rectangular multi-story rigid, plane frame structures, with or without set-backs and/or overhangs, subjected to static loadings.

STRES1 can accommodate any plane frame composed of horizontal and vertical members up to 20 stories high and up to 10 bays in width of a GE-420 time-sharing system. Much larger structures can be accommodated with the Burroughs-5500 version of STRES1.

STRES1 is completely conversational in its time-sharing versions.

STRES1 computes the end shears and moments by a modified Hardy Cross relaxation process for each member in the structure.

GE-420, Burroughs-5500 Time-Sharing Systems (other hardware versions available), FORTRAN IV

Frank E. Eby
National Engineering Software Co., Inc.
P.O. Box 9404, Rosslyn Station
Arlington, Va. 22209

A0182

STRES2 N/A

STRES2 is a general purpose computer program which analyzes plane truss and frame structures composed of prismatic elastic members and subjected to static loading conditions.

The program can accommodate plane structures that are composed of a combination of pinned and fixed ended members. STRES2 computes the axial force, shear, and bending moment at each end of each member in the structure. The program will give the user a complete listing of all of the joint displacements in the structure upon request. The user may mix the types of materials used for the various structural members and loads may be placed anywhere on the structure. Such loads may be either concentrated, uniformly distributed, or moments.

STRES2 utilizes the stiffness method of solution and can accommodate up to 60 joint displacements on a GE-420 time-sharing system. Much larger size problems can be accommodated on the Burroughs-5500 time-sharing system.

The time-sharing versions of STRES2 are completely conversational.

STRES2 will accommodate a plane structure of any geometry, i.e., sloping as well as horizontal and vertical members.

GE-420, Burroughs-5500 Time-Sharing Systems (other hardware versions available), FORTRAN IV

Frank E. Eby
National Engineering Software Co., Inc.
P.O. Box 9404, Rosslyn Station
Arlington, Va. 22209

A0183

SHIP HYDROSTATIC CURVES $300/run

This program produces all data necessary to plot a complete set of hydrostatic curves: volume, tons, LCB, LCF, VCB, TPI, BMT, BML, KMT, KML, block, prismatic, waterplane, midship section coefficients, Bonjean areas.

IBM 1130, 8K, FORTRAN IV

N. Harper
Scotian Design Services
Box 726
Port Neches, Texas 77651

A0184

SHIP LINES FAIRING $350/run

This program produces mold loft type offsets from a scale lines drawing. The hull is divided into flat and curved regions by control lines. The control lines are faired and used for reference. Program fairs offsets on waterlines and stations until successive iterations vary by less than 0.01 foot. Fairing matrix will accept 12 stations by 12 waterlines.

IBM 1130, 8K, FORTRAN IV

N. Harper
Scotian Design Services
Box 726
Port Neches, Texas 77651

A0185

SHIP STIIL WATER BENDING MOMENT $300/run

This program balances the load curve for weight = buoyancy and LCB = LCG. The shear curve is balanced for equal positive and negative areas. The hydrostatic curve program automatically prepares the buoyancy input.

IBM 1130, 8K, FORTRAN IV

N. Harper
Scotian Design Services
Box 726
Port Neches, Texas 77651

A0186

REFORMING PROCESS SIMULATION (Gasoline or BTX Operations) $15,000

The models are designed to simulate the reactor system of a refinery gasoline or BTX reforming operation. Primary calculations embodied in the model are for kinetics of dehydrogenation, cyclization, isomerization and cracking; heat balances; yields and economic results.

The models are used to study the effects on reformer performance of changing operating variables, feed types and catalysts.

Card Input, Printer Output, 8K, FORTRAN

Roger O. Pelham
Profimatics, Inc.
6355 Topanga Canyon Blvd., Suite 433
Woodland Hills, Calif. 91364

A0187

FLASH VAPORIZATION COMPUTATION $2,500

This program calculates a hydrocarbon mixture vapor/liquid separation given specified temperature and pressure. Depending on source data available, the program can be run in one of several modes. Either K values, liquid and vapor mole fractions (X and Y), X and relative volatilities, or Y and relative volatilities must be supplied. This flexibility extends the program to a wide range of design and operating conditions.

Card Input, Printer Output, 4K, FORTRAN

Roger O. Pelham
Profimatics, Inc.
6355 Topanga Canyon Blvd., Suite 433
Woodland Hills, Calif. 91364

A0188

FLUID CATALYTIC CRACKING (FCC) PROCESS SIMULATION $18,000

The model is designed to simulate the reactor-regenerator section of a fluid catalytic cracking unit. Primary calculations include those for heat balances, and kinetics of reaction, regeneration and afterburning. Output includes yields, operating conditions and economic values.
The model is used to study the effect of FCC performance of changing feeds, catalysts, and operating conditions.

Card Input, Printer Output, 12K, FORTRAN

Roger O. Pelham
Profimatics, Inc.
6355 Topanga Canyon Blvd., Suite 433
Woodland Hills, Calif. 91364

A0189

ALKYLATION MODEL $12,500

The model is designed to simulate the olefin alkylation process for production of high octane gasoline. Primary calculations include the alkylation reaction kinetics, distillation and isobutane recycle rates, and refrigeration loads.
The model is used to study the effect on refinery economics of changing alkylation feeds and operating conditions.

Card Input, Printer Output, 8K, FORTRAN

Roger O. Pelham
Profimatics, Inc.
6355 Topanga Canyon Blvd., Suite 433
Woodland Hills, Calif. 91364

A0190

GCAP N/A

Generalized Circuit Analysis Program for electrical circuit AC and DC analysis.

PDP-10, FORTRAN

Dr. Eamonn McQuade,
Mgr., Systems Programming, Computer
Graphic Controls Corporation
189 Van Rensselaer Street
Buffalo, N.Y. 14210

A0191

ELECTRONIC CIRCUIT ANALYSIS PROGRAM (R/ECAP) N/A

This package enables circuit designers to model an electronic circuit using familiar equivalent circuit elements and to perform AC, DC, and/or transient analyses through the facilities of a typewriter-like time sharing terminal.

IBM 360/65, FORTRAN

Joel Leichter, Vice President
Princeton Time Sharing Services, Inc.
U.S. Highway No. 1
Princeton, N.J. 08540

A0192

LOGSIM—A GENERAL LOGIC SIMULATION PROGRAM $1,495

Simulates general digital systems. Extremely easy to use with free format input.

IBM 360/50, Univac 1108
95% FORTRAN IV, 5% BAL

Robert E. Marsh
Software Products Co.
3409 St. Joseph Avenue
Berrien Springs, Mich. 49103

A0193

(Continued on page 44)

S/A market place

where you can reach 125,000 programmers, mathematicians, analysts and EDP managers

RATES

Classified—75¢ per word, minimum order $7.50. Blind ads are an additional $10.00. Payment must accompany copy.

Display Classified—One inch by one column, $75.00. Column width is 2¼". Help wanted and employment services ads will be accepted in the classified section. Ads of ⅛ page or more will be listed in the Advertisers' Index, and will qualify for the free resume service.

Closing Date—the 10th of the month preceding (i.e. May issue closes April 10th). Send order and remittance to:

software age
Classified Department
P. O. Box 2076, Madison, Wisconsin 53701

CLASSIFIED ADVERTISING ORDER FORM

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25
26	27	28	29	30
31	32	33	34	35

________Words = $________

☐ Blind Ad $10.00 Insert________time(s) Total Enclosed $________

NAME________________________

ADDRESS________________________

CITY____________ STATE____________ ZIP________

Signature________________________

WORD COUNT: Includes name and address (unless blind ad). The name of the city (Des Moines), state (New York) and zip code are counted as a total of two (2) words. Each abbreviation, initial, single figure or group of figures or letters count as a word. Symbols such as 35mm, COD, AC, etc. count as a single word. Hyphenated words count as two (2) words.

HELP WANTED

SCIENTIFIC SYSTEMS ANALYST. Opportunity to implement and take charge of a new real-time, medium size, scientific computer installation. Requires person with experience in real-time system to instrumentation. Prefer candidate to have a background in chemistry and mathematics. Envision multiple instrumentation handled on an interrupt driven priority basis (mass spectrometer, gas chromatographs, autoanalyzer, IR, etc.). System will also involve extensive background batch processing. Candidate would work with vendor and R & D personnel prior to computer delivery in 3rd Quarter of 1970. Excellent opportunity and working conditions, company paid benefits including profit sharing and relocation expenses. Write to: Administrative Manager, PHILIP MORRIS RESEARCH CENTER, P. O. Box 3-D, Richmond, Va. 23206. An Equal Opportunity Employer.

COMPUTER PROGRAMS WANTED

WE WILL PAY CASH or high royalties for your computer programs. Mail details to Armor EDP Consultants, 2 Pennsylvania Plaza, New York, N. Y. 10001.

PROFESSIONAL SERVICES

PROGRAMMING WITH ANALYSIS. Complete senior services in Scientific or Commercial FORTRAN, BAL, COBOL. Reasonable rates. Software available. Call collect 212–944–3464, Eastern Research Group, N. Y. C.

PRODUCTS

A COMPREHENSIVE MANUAL on how to write an effective EDP resume, plus 6 successful samples. Send $4.25 to K & H ASSOCIATES, Resume Specialists, P. O. Box 265, Horseheads, N. Y. 14845.

PROGRAMMING AIDS for 3rd generation analysts: Improved pocket HEXADDER $15, HEX-a-rule $5, free catalog. HEXCO, Dept. SA, Box 55588, Houston, Texas 77055.

CONVERT NOW contact Atlantis Enterprises, P. O. Box 1501, Sunnyvale, Calif. 94086.

continued from page 20

I IS EVEN
I IS EVEN
I IS EVEN

The following systems belong in this group:

CDC-6000; CDC-3800
B-5500
PDP-10
Com-Share's FORTRAN II

Winners of problem 19 ($10.00 each)

F. Powers, B-5 Kamiak Apts., Pullman, Wash. 99163

C. W. Cooper, Johns Hopkins University, Silver Spring, Ma. 20910

H. L. Longacre, 401½ 21st Street, Manhattan Beach, Cal. 90266

R. E. Lehman, 13855 Superior Road #1510, Cleveland, Ohio 44118

L. R. Grosenbaugh, Box 54164, Atlanta, Ga. 30308

XTRAN

(Continued from page 41)

FINANCIAL

INVESTMENT AND BORROWING

STOCK SCREEN FOR INSTITUTIONAL ACTIVITY $450

Calculates a derivative for any listed stock of price action for ten days back. Sorts and presents selections for analysis.
Uses statistical theory to set limits for volume analysis using daily quoted data to show extraordinary activity for investment choices.
This program uses the computer to detect institutional activity which makes up so much of the present day stock market. It acts on short term, to allow trading to closely follow the "best informed."
No limit is set on number of issues processed, hence the "screen" effect is one of its most valuable characteristics. Modified statistical formulas are employed to flag deviations over a set of gates or parameters. All assumptions are documented and calculations have been tested.
Input could be adapted for on-line use, or simply be taken from the newspaper quotations if not more than a day old.

Card, Printer, Any FORTRAN IV Compiler, FORTRAN IV

John B. Wineman—
Eugenia Wineman, Ph.D.
7508 West 90th Street
Los Angeles, Calif. 90045

A0194

MUNICIPAL BOND PRICING PROGRAM N/A

By computer analysis of a broad selection of known market offerings from sources such as the Blue List, Dealer Offering Lists, and new issue sales, the current relationship of yield, coupon, maturity and quality of bond is established. The derivatives of this analysis are then utilized to construct a municipal market model, which fully reflects municipal market conditions as of the date of analysis. Three primary considerations of determining the yield of a municipal bond are: 1. Quality of bond; 2. Coupon; 3. Maturity. The quality of the bond (credit characteristics) varies within the popular ratings AAA, AA etc. The municipal bond pricing program allows for a wide disparity in ratings through a numbering system which provides a more refined definition of relative trading characteristics, while credit characteristics are the starting point for estimating the worth at which a bond will trade. In addition to the three primary considerations for determining the yield of a bond, other considerations must be examined, namely: 1. Present of a call feature; 2. Whether the bid or offered side of the market is required; 3. Influence of odd lot. Accuracy of prices then is a function of the following: 1. Type of bond; 2. Maturity; 3. Coupon; 4. Quality; 5. Level of Market 6. Call feature; 7. Bid or offered side of market; 8. Odd-lot influence; 9. Special traditional considerations.

360/30, 32K, 1—2311 Disc, 2-Tape Drives, FORTRAN IV

Arthur Dorfman
Arthur Dorfman Associates, Inc.
2790 West 5th Street
Brooklyn, N.Y. 11224

A0195

BOND HANDLING PACKAGE $10,000

This package consists of two groups of programs; Bond Accounting and Bond Pricing.
Bond Accounting is designed to produce a bank's securities subsidiary ledger and, at the same time, provide useful portfolio management information. The accounting program describes each issue, accrues interest income, amortizes premium, accrues discount, prints the current amortized and/or accrued book value, prints net income for the current period, prints actual cash that should have been received, summarizes information by category by bank. Tax program shows taxable gains and/or losses. Coupon program prints list due to be clipped for payment. Bond pricing programs produce a current market appraisal of a bank's government and municipal portfolios. A statistical profile is prepared which reflects average maturity, average yield at cost, average coupon, etc. A summary is produced which provides book value, market value, total gains and losses, etc., for each category of security. Summaries by year of maturity or geographical location, individual pricing which appraises each item individually, and an escrow report are also prepared.

360 65K, COBOL DOS

David E. Weisberg
URS Data Sciences Company
1700 South El Camino Real
San Mateo, Calif. 94402

A0196

CAPITAL INVESTMENT RISK ANALYSIS MODEL $5,000

Time Sharing Computer Model.

Comshare Time Sharing System, FORTRAN IV

George Malanga
Business Strategies, Inc.
101 Park Avenue
New York, N.Y. 10017

A0197

CERTIFICATES OF DEPOSIT—MULTIPLE BANKS $5,000

Produces weekly report of new, redeemed, and changed certificates plus alphabetical trial balance. Monthy processing produces a 12 month accrual and cash flow forecast. Interest journal and checks/disposition notices are also produced each month. Interest compounded daily and paid monthly, quarterly, semi-annually, or annually. 1099's are printed during last processing run in December.

360/30 32K, 3—2311, 1442 or 2540, BAL

Jim Smith, Sales Representative
Union National Bank
Public Square
Springfield, Mo. 65805

A0198

TIME DEPOSIT ACCOUNTING SYSTEM N/A

Scientific Resources Corporation offers a multi-bank system with bank control file for flexibility in individual processing of savings bonds, certificates of deposit and special pass-books. Provides interfact to central file through separate name/address file. Allows any compounding frequency and rates. Includes automatic renewals with notices to customers, interest checks audit confirmations and complete line (32) reports. May be processed at varied frequencies depending on transaction volume.

360 64K DOS 2311 Disk and Tape, COBOL

D. N. Panaccio
Scientific Resources Corporation
Commerce and Enterprise Drives
Montgomeryville, Pa. 18936

A0199

NOTE: SOFTWARE AGE does not broker computer programs nor act as an agent. As an unbiased third party, SOFTWARE AGE is acting solely as a pipeline of communications between buyers and sellers of computer programs.

A prerequisite for any listing is that it be documented or supported to the extent of being operable by the buyer.

software age

CONFIDENTIAL INQUIRY

Your original copy of this form will be retained at the offices of SOFTWARE AGE and will be used for no other purpose than to notify the specific firms which you have checked (on the reverse side) of your interest.

TYPE OR PRINT CLEARLY FOR PHOTO REPRODUCTION

JOB DESIRED: ______________________

List computer hardware knowledge (names of systems, tape, disk, terminals, etc.): ______________________

Programming specialties and years of experience (commercial, scientific, theoretical, experimental, analog, etc.): ______________________

Systems programming on which you have had development experience (compilers, assemblers, executives, monitors, O.S., etc. Indicate for what computer): ______________________

Programming languages used and extent of experience (COBOL, FORTRAN, etc.): ______________________

Applications programmed (aerospace, banking, insurance, math subroutines, compilers, etc.): ______________________

Systems analysis experience (card design, flow charting, operation analysis, etc.): ______________________

EDP management experience (include years and number of people reporting to you): ______________________

SALARY: __________ (current) __________ (desired) **DATE OF AVAILABILITY:** __________

EDUCATION: Indicate major as well as degree unless self-explanatory.

Degrees			
Years			
Schools			

EMPLOYMENT: Indicate present employment and previous jobs below.

Employer			
City			
Years	to	to	to
Title or Function			

Name __________ Signature __________

Home Address __________ (city) __________ Home Phone __________

__________ (state) __________ (ZIP code) Age __________ U. S. Citizen? __________

Security Clearance __________ Location Preference __________

Marital Status __________

Military Status __________

BE SURE YOU HAVE CHECKED ON REVERSE SIDE THE COMPANIES YOU WANT TO SEE THIS INQUIRY. PUT FORM IN STAMPED ENVELOPE AND MAIL TO:

software age
MAGAZINE
P. O. BOX 2076
2211 FORDEM AVE., MADISON, WIS. 53701

check your interests here

Fill in the confidential inquiry form on the other side of this sheet. This form provides all the information advertisers require to screen applicants. If further information is desired, you will hear from the advertiser direct. Then, check below the boxes of those companies to which you want copies of your form sent. Mail to SOFTWARE AGE, P.O. Box 2076, 2211 Fordem Avenue, Madison, Wisconsin 53701. (**Please do not send us your own resume. We will only process this form. A new form must be filled out for each issue in which you are answering ads.**)

EMPLOYMENT AND SEARCH AGENCIES

PRODUCTS AND SERVICES

☐ **I do not now receive S/A. Please send me information on how I may receive a FREE subscription.**

....
Name

....
Street Address

....
City State Zip Code

software age

P. O. Box 2076
2211 Fordem Avenue
Madison, Wisconsin 53701