An automatic programming approach has been developed for the use of sensor based computers (IBM System/7 and Series/1) for energy management in buildings. The purpose is to aid the facilities engineer who is unfamiliar with programming and who requires a system that can be defined by a sequence of questions and answers. Programmers can add or modify application source programs to extend the system to other user-defined functions.

Automatic programming for energy management using sensor based computers

by M. J. Shah

It has been more than 15 years since digital computers were introduced for process control. Digital control systems have evolved from systems such as the IBM 1710 and 1800 to special controllers that use microprocessors. The hardware cost has dropped considerably, yet the cost of installing computer control systems in sensor based applications has remained relatively high. This is partly because of the rising cost of system components other than the digital computer itself, one such component being the system software.

In the early 1960's, the cost of digital control systems was high enough so that computer vendors provided manpower for one-of-a-kind installations, in cooperation with the manpower provided by the customer. Successful installations for digital computer control required many man-years of effort.

As hardware costs started to decrease in the late 1960's, and as computer vendors began to understand the real-time control environment in the process industries, several multiprogramming real-time executive programs were developed by the computer industry. These general-purpose process control programs were

forerunners of the executive programs subsequently provided on the larger business computers. Control computers, with small available main storage, provided multiprogramming in many partitions with a large number of pre-emptive interrupt and priority levels (386 on the IBM 1800) to satisfy the fast response required for plant alarm conditions.

Application programs such as IBM's PROSPRO and DDC programs[1,2] were subsequently developed with a forms oriented approach to provide information on variables controlled or monitored by the digital computer system. These programs eliminated a substantial portion of the programming effort for system installation, so that plant monitoring and alarm functions, in many cases, could be started within only a few weeks of the hardware installation. The time was reduced if the system executive was generated prior to the computer installation and all the forms for plant variables were filled in to provide data for the supervisory programs such as DDC and PROSPRO before the system arrived. Several weeks still were required, of course, for programmers and engineers to prepare for the installation. And beyond the plant monitoring function, considerable effort was required to implement control functions, often extending into months after the initial installation.

In the early 1970's, further reduction in hardware cost was reflected in systems such as the IBM System/7 for sensor based applications. A host program preparation facility was developed for generating the System/7 real-time operating system, as well as application programs, on large systems such as System/360 for subsequent transmission to a System/7. In addition, the Application Program Generator (APG),[3] a PL/I-like language, was developed to further reduce software effort. In spite of these developments, and the availability of specialized application packages for System/7, the installation effort was not reduced dramatically, in part because many customer installations could not justify a full-time programmer. Thus computer vendors were forced to provide extensive software support.

With the introduction of yet smaller and less expensive process control systems such as the IBM Series/1, it has become mandatory to strive toward a programmerless environment in many applications so that the benefits of digital computer control can be provided with a reasonable software/hardware cost ratio.

Application requirements

As opposed to a batch data processing environment, real-time sensor based applications require some special user interfaces to the digital computer, as outlined below.

- Interfaces must be defined between the user device (control point) and the digital and analog hardware input/output addresses on the computer.
- Decisions must be made regarding alarm scanning frequency, program scheduling for alarm and normal functions, priority assignments, and system resource allocation.
- Programs must be scheduled according to application requirements, such as controlled device cycling in energy management applications.
- A multiprogramming executive interface to the system supervisor must provide all time-dependent inputs with assigned priority levels and specify which programs are to occupy which partitions, in order to optimize memory usage.
- Files of different sizes on bulk memory (disk or diskette) must be allocated for the user's environment, which varies with the number of sensors, control points, and alarm points and the length of time the user wishes to retain data logs.

Basically, the user must be able to tailor an application program, its supporting supervisor, and the associated bulk files by using "fill-in-the-blanks" forms. Questions on the forms should address the user's application; they should not require users to provide program specifications. Suggested default values should be provided in case the user is not sure of the answers at the beginning.

A programming automation approach used in a specific application is described below. The application is energy management in a building, using IBM System/7 and Series/1. The task is to enable a facilities engineer who is unfamiliar with programming to define his system by means of questions and answers. In addition, a facility also should be provided that enables programmers to modify application programs or add new programs to extend the functions of the system.

Facilities energy management

Facilities energy management was chosen for this programming approach because a large number of digital control systems have been installed successfully for this application. Furthermore, because of the continually increasing cost of energy, energy management provided a potential computer application for smaller users who are unsophisticated in programming. The user environment can range from total unfamiliarity with control computers (as in many department stores, for example) to the sophistication of a large industrial facility with a trained programming staff. The application does not require a great deal of control complexity, so the success of this approach can be measured independently of the control strategies for individual facilities.

Essentially, in this application, the digital computer performs the following functions:

- Monitors power consumption from one or more power meters, comparing it with time-dependent target and maximum consumptions. When specified targets are exceeded, selected devices are turned off for periods specified by the user.
- Turns devices on and off according to the time of day, and periodically cycles devices during their *on* period, as during the first shift.
- Monitors large numbers of alarm-condition points and overrides control of one or more devices according to whether the alarm points are on or off. Alarm conditions may arise if values of temperature, pressure, or flow in air conditioning equipment exceed limits, or if the environmental temperature or humidity exceeds limits, or if the security of an area is violated.
- Controls inlet air dampers based on outside air enthalpy; adjusts device *off* times according to outdoor temperature and interfaces to control panels in facility control rooms.

Details of this application are given in References 4 and 5.

Device control functions are performed by digital output from the control computer. Power consumption, alarm signals, and temperature and humidity measurements are read from digital input.

Any problems that involve noise isolation or signal loss generally are handled through a hardware interface between the sensor and the computer system. Interface noise problems are not insignificant, but experience with large numbers of systems has led to standard designs for minimizing them.

Automatic program generation on System/7

In a previous paper,[6] the author described a "fill-in-the-blanks" approach to generating power management application software for the IBM System/7. That work is described here briefly because the Series/1 approach evolved from experience with the System/7 program, and the System/7 program involved some new concepts in program automation for sensor based applications.

The operating system for System/7, referred to as MSP/7 (Modular System Program for System/7),[7] is created by specifying parameters for system macros (coding the macros) and assembling them to prepare a system nucleus. Nucleus preparation involves specifying hardware addresses for data processing I/O, sensor based I/O, and timers, as well as scheduling the frequency with which application programs are to run under the operating system and

Figure 1 Application prototype (in part) before processing with user answers

```
*:      IF    ( :&ADAPTIV EQ 'NO' )*:LLA
        DCL   ADAPTIV (:&METERS) BIT EXT ;
        DCL   ADPTUNT (:&METERS) FIXED BIN (15) EXT ;
*:LLA
        DCL   ALLMSG ( :&METERS ) BIT (1) EXTERNAL ;
        /* POWER CONSUMPTION FOR EACH SWITCH. TO BE INITIALIZED VIA */
        /*  OPERATOR STATION.
        DCL   CONSSWT(:&METERS,:&MAXUNS) FIXED BINARY(15) EXT ;
        DCL   FPERIOD (:&METERS) FIXED BINARY (15) EXT ;
        DCL   IBAND (:&METERS) EXT ;

CMPR_LOOP:
              DO S = ISET TO TOTSW(M)-MAXFAN(M) ;
                 ISETCMP(M) = S + 1  ;
                 IF ICNTLTB(M,S) > ZERO
*:      IF    ( :&LOCKING EQ 'NO' )*:NNE
                      | LOCK(M,S)
*:NNE
*:      IF    ( :&OVERIDE EQ 'NO' )*:NNF
                      | IOVERID(M,S)
```

Figure 2 Application source (in part) after processing the prototype

```
        DCL   ADAPTIV (02) BIT EXT ;
        DCL   ADPTUNT (02) FIXED BIN (15) EXT ;
        DCL   ALLMSG ( 02 ) BIT (1) EXTERNAL ;
        /* POWER CONSUMPTION FOR EACH SWITCH. TO BE INITIALIZED VIA */
        /*  OPERATOR STATION.
        DCL   CONSSWT(02,010) FIXED BINARY(15) EXT ;
        DCL   FPERIOD (02) FIXED BINARY (15) EXT ;
        DCL   IBAND (02) EXT ;

CMPR_LOOP:
              DO S = ISET TO TOTSW(M)-MAXFAN(M) ;
                 ISETCMP(M) = S + 1  ;
                 IF ICNTLTB(M,S) > ZERO
                    | LOCK(M,S)
                    | IOVERID(M,S)
                    | CHILLSW(M,S)
                      THEN GO TO NEXT_SWITCH ;
```

specifying the initial startup program at program load time. To avoid difficulties for the facilities engineer who is unfamiliar with programming, it is desirable to automate this process.

The fill-in-the-blanks facility for APG/7[3] provides some degree of automation for preparing the MSP/7 nucleus, with a reasonable degree of validity checking of answers. This facility was extended to tailor a complete application source program, step by step as follows:

- The application source, written in the high-level language of APG/7, is made into a "prototype" which is processed according to the user's answers to form a compilable source. Portions of the prototype are shown in Figure 1, and the program source after processing is illustrated in Figure 2. Dimensions are filled in, source code is eliminated, repeated, or retained, and variables are defined, all based on the user's answers.
- A file containing the APG fill-in-the-blanks answers for generating the System/7 operating system macro source was itself made into a prototype, which was processed according to a minimum of user answers. This in effect allowed the creation of system parameters such as the number of multiprogram partitions, and it allowed specification of the storage location of programs based on user answers regarding machine mem-

Figure 3 Question forms for application requirements

```
                IBM SYSTEM/7 APPLICATION PROGRAM GENERATOR
                        METER FORM (LEVEL 02)
                     *** METER SPECIFICATION ***

                                                          METER - -
                                                       CC 1     7
          USE THIS FORM TO DESCRIBE EACH POWER METER
          IN YOUR SYSTEM.

          COMPLETE ONE FORM FOR EACH METER. THIS IS
          A 3 PAGE FORM; SO BE SURE TO COMPLETE ALL PAGES.

          SPECIFY A UNIQUE METER NUMBER (01-32) IN
          COLUMNS 7 AND 8.

CC
12      21

UNITS   - - - 1     HOW MANY POWER CONSUMING UNITS ARE MONITORED
                    BY THIS METER? INCLUDE FANS (SECONDARY UNITS),
                    IF ANY.
                    SPECIFY FROM 001 TO 704
                    (DEFAULT=008)

FANS    - - -       HOW MANY FANS (SECONDARY UNITS) ARE MONITORED
                    BY THIS METER?
                    SPECIFY FROM 000 TO 704
                    REFER TO THE 'FANS' QUESTION ON THE 'PMAPCONF'
                    FORM (LEVEL 01).
                    (DEFAULT=002)

CHILLSW -           IS A CHILL SWITCH ATTACHED TO THIS
                    METER?
                    SPECIFY 0 FOR NO, 1 FOR YES
                    REFER TO THE 'CHILLSW' QUESTION ON THE
                    'PMAPCONF' FORM (LEVEL 01).

ADPTCON - - -       IS 'FLOATING TARGET' USED FOR THIS METER?
                    SPECIFY YES OR NO
                    IF YES, ANSWER THE NEXT 3 QUESTIONS.
                    REFER TO THE 'ADAPTIV' QUESTION ON THE
                    'PMAPCONF' FORM (LEVEL 01).
                    (DEFAULT=NO)

ADPTUNT - - 7       WHAT IS THE PERCENTAGE OF THE TOTAL NUMBER OF
                    UNITS ON THIS METER THAT MUST BE SHUT DOWN
                    BEFORE ADAPTIVE ADJUSTMENTS WILL BE MADE?
                    SPECIFY FROM 01 TO 50
                    (DEFAULT=50)

ADPTPRC - - 8       BY WHAT PERCENTAGE IS THE FLOATING TARGET
                    TO BE ADJUSTED UP OR DOWN?
                    SPECIFY FROM 01 TO 50
                    (DEFAULT=10)

ADPTFAC - - 9       BY WHAT FACTOR WILL THE FLOATING TARGET (KWSX)
                    BE MULTIPLIED WHEN IT IS CHECKED AGAINST THE
                    CURRENT POWER USED (KWUSED) TO DETERMINE IF THE
                    FLOATING TARGET SHOULD BE RESET TO THE STANDARD
                    KW (KWS) VALUE? THIS PERMITS RAPID UPWARD AD-
                    JUSTMENT OF THE FLOATING TARGET WHEN LARGE LOAD
                    INCREASES OCCUR BETWEEN TIME ZONES, SUCH AS THE
                    CHANGE FROM NIGHT SHIFT TO DAY SHIFT.
                    SPECIFY FROM 02 TO 10
                    (DEFAULT=03)
```

Figure 4 Application answer prototype (in part) for system configuration

```
CONFIG      STGSZ     :&STORAGE
CONFIG      ASM       YES
CONFIG      PROGS     10
*:        IF     ( :&DISK EQ 'YES' )*:CCC
*:        IF     ( :&DUMP EQ 'YES' )*:BB2
*:        IF     ( :&PATCH EQ 'YES' )*:BB2
*:        IF     ( :&PRINT EQ 'NO' )*:CCC
*:BB2
SCHED       SCHLEVEL 2
SCHED       SCHED    1
SCHED       USERSCHD YES
SCHED       MAXSCHD  12
CYCLIC 01   PROGRAM  POWRCHK
CYCLIC 01   COUNTS   :&INTERVAL<PMAPCONF>
CYCLIC 01   OFFSET   607
CYCLIC 02   PROGRAM  POWR
CYCLIC 02   COUNTS   :&SCHEDSS<PMAPCONF>
CYCLIC 02   OFFSET   63
CYCLIC 03   PROGRAM  POWRMSG
CYCLIC 03   COUNTS   300
CYCLIC 03   OFFSET   205
*:        IF     ( :&DISK EQ 'NO' )*:FFF
CYCLIC 04   PROGRAM  POWRCPW
CYCLIC 04   COUNTS   3600
CYCLIC 04   OFFSET   61
CYCLIC 05   PROGRAM  PMAPSCH
CYCLIC 05   COUNTS   :&INTERVAL<PMAPCONF>
CYCLIC 05   OFFSET   602
*:FFF
```

ory size and whether the system incorporates disk storage. Some typical questions are shown in Figure 3. Figure 4 illustrates part of the answer prototype as processed using answers to the questions in Figure 3.

- The job control language required to assemble the system supervisor and compile the application programs created in the first two steps was also made into a prototype file. This file was processed according to user answers, creating a system tailored to fit the application requirements based on the user's machine configuration.

In essence, the user answers questions on forms prepared for the application, enters the answers by means of a keyboard or punched cards, and follows a prescribed ten-step procedure to generate his system. The ten steps themselves could have been automated, but an inaccurate answer, recognized at the end of system generation, could require total system regeneration. In addition, it was decided to provide the user with system information between procedural steps. Further, a programmer can intercede between steps to add or modify application programs or modify the system nucleus. Depending on the complexity of the user's application, total system generation, after specifying the user answers, took 5 to 12 hours of System/7 time.

In almost all System/7 energy management installations, systems engineering assistance was required for system generation. When the stepwise procedure was followed, program generation and installation went smoothly. Difficulties arose when program modifications and functional extensions to the application were required, mainly because it was necessary to have operational knowledge of nucleus generation, system macro definitions, the job control procedure for the compiler, the assembler, the linkage editor, and the source language.

Major objections to this procedure on System/7 can be summarized as follows:

- System generation took several hours and, in most cases, required a System/7 larger than the energy-management System/7.
- Changes in some answers to the system questions required regeneration of the system.
- Program modification or extension required not only knowledge of APG/7, but availability of the APG/7 compiler and macro assembler at the system generation site, and familiarity with the System/7 program preparation facility job control language.

Clearly, automation must go further for systems such as the IBM Series/1, the user environment for which is less sophisticated

than that for System/7, and program preparation is to be minimized if not entirely eliminated.

Energy management with Series/1

The IBM Series/1 is a sensor based digital computer system with bulk memory on a fixed disk as well as on removable diskettes. With certain access frequency restrictions, the diskettes can be used for programs as well as data.

From the standpoint of cost justification, the application has to be diskette based rather than disk based. All interdependencies among the real-time executive, the language compiler, and the application are to be avoided as much as possible so the user will need no additional training or background to modify the application program. The application system had to be self contained and easy to modify with a simple, high-level language, both for writing new application programming modules and for modifying the standard programs included with the application. These requirements were addressed as described in the following paragraphs.

The architecture of the energy management real-time supervisor for Series/1 is based on the System/7 LABS/7 and EDX/7 programs.[8,9] Support in the supervisor was restricted to the I/O devices used in the application.

A subset of EDX/7 was chosen as the high-level language for the application program, with certain additions and modifications in the input/output commands. Data queuing features were added for communication between asynchronously executing modules. The command set provides a PL/I-type language via structured macros, but it is simpler and more primitive, and it is closer to assembler language.

The system generation step is eliminated by providing all executable programs in object form on a diskette. As is shown below, the program architecture made the execution programs independent of variations in most user requirements.

A forms oriented, step-by-step personalization procedure was included for entering all data about the user's facility. This information was stored in data tables on a diskette. All the tables were constructed with reasonable default values for as many parameters as possible when user answers were omitted. Diskette file space was allocated for a maximum number of control and monitor points.

The program was designed with functional modules, so that the user could start up his installation within minutes by responding,

Figure 5 Facsimile of power management application initialization form and prompted questions at initial-program-load time

Initialization Form

		Answer*	Entered
1.	Meter factor (000.001-999.999)		☐
2.	Pulse available (Y/N)		☐
3.	Demand calculation type (1=sliding window, 2=integrated)		☐
4.	Demand increment size (15-120 seconds)		☐
5.	Demand interval size (5-120 minutes)		☐
11.	*Operations monitor used on DO point 2* (Y/N)		☐
12.	Automatic restart clock available (Y/N)		☐
13.	FC/PM 3 features used (Y/N)		☐
14.	Digital device addressing used (Y/N)		☐
15.	Total number of devices (1-95; 1-256 if digital device addressing of FC/PM 3 used)		☐
16.	Total number of monitoring points (0-95)		☐
17.	Number of integrated DI/DO cards used (1-5)		☐

*Entry must be made

```
>ENTER METER FACTOR:
>PULSE AVAILABLE? (Y/N):
>ENTER DEMAND CALCULATION TYPE:
>ENTER DEMAND INCREMENT SIZE:
>ENTER DEMAND INTERVAL SIZE:
>OPS MONITOR USED ON DO POINT 2 (Y/N):
>AUTO RESTART CLOCK AVAILABLE? (Y/N):
>FC/PM 3 FEATURES USED? (Y/N):
>DIGITAL DEVICE ADDRESSING USED? (Y/N):
>ENTER TOTAL NUMBER OF DEVICES:
>ENTER TOTAL NUMBER OF MONITORING POINTS:
>ENTER NUMBER OF INTEGRATED DI/DO CARDS USED (1-5):
```

at initial-program-load time, to the prompts shown in Figure 5. The user could then progress by providing on-line data about his control and monitoring points and his facilities control strategies. The application also was divided into operational modes, so when the user provides data for an individual function, the corresponding functional mode can be activated. Table 1 shows the various application modes and their functional descriptions.

All communication between program modules regarding the user's facility is via data tables which, with default values, are on the application program diskette. Diskette tables and files for logs are pre-allocated for the maximum building facility supported by the program. The user is thus relieved of having to allocate files based on his particular requirements. The tables, based on user

Table 1 Application modes and their corresponding functions

Mode	*Function*	*Actions*
1	Power monitoring	Consumption pulses read. Demand interval report printed.
2	Alarm monitoring	Monitor points read. Alarm sounded, if required. Standby function available. DVON, DVOF, DVRS commands usable.
3	Time-of-day start/stop control	All devices placed and maintained in time-of-day state.
4	Cyclic start/stop control	All cyclic control devices (defined on device parameter forms) are cycled during time-of-day *on* time.
5	Demand control	All demand control devices (defined on device parameter forms) are controlled.
6	Extended condition switch control	Extended condition switch feature becomes active.
7	Enthalpy control	Outside air inlet damper under computer control.
8	Outside air strategy control	Certain device off-times (defined on temperature-dependent basic parameter form and temperature-dependent device off-time form) are modified when there is an environmental temperature change.

answers to personalization parameters, are built in main storage for each functional mode when the mode is activated by the user. The same table storage is released when related functional modes are deactivated. The GETMAIN and FREEMAIN commands in the application language allow this function. The storage table addresses were always maintained in a system vector table, which was used by programs to address data in the various tables. The personalization parameters are developed from forms which are described in Reference 10.

Utilities are provided for users who want to modify programs and extend the application functions by programming their own modules. The utilities include a function for translating a module written in the EDX subset language into an executable program, as well as library facilities for allocating, copying, listing, and modifying files on a diskette, and a source library editor with TSO-like commands to edit, insert, and modify source text for program modules. Because the system is diskette based and has limited bulk storage, certain size restrictions are required, such as a maximum of 425 program source statements and 30 new user modules.

For users who have no need to modify programs, as well as for those with a programming background who want to modify or add program modules, the system is totally self contained. It does not depend on any other operating system or program preparation facility, and no knowledge of compiler or assembler operation is required.

The need for a job control language is eliminated by using simple commands for library utility functions, for source editing, and in the programming language itself. Wherever possible, utility functions are designed to operate in a conversational format so as to minimize operator errors. Details of the commands are given in Reference 10.

Although program preparation utilities can be used only off line, requiring the application to be stopped, the program source can be modified and translated into a load module in 10 to 20 minutes, excluding time required for printing during translation and operator entries for the source editor. This is possible because the program architecture allows each application module to function separately, communicating via common tables, with the supervisor performing relocatable loading. As long as program modification is preplanned and source changes involve no more than 10 to 15 lines in a module, the entire application can be brought back on line within an hour.

Application generation on Series/1

This section shows the steps a user follows to modularize, or personalize, the program for his building facility. The personalization programs execute in a transient area while the application is running, so the user can change parameters in his facility after his initial specifications, or he can activate additional control and monitoring points after specifying their characteristics by personalization.

The user is given forms to fill out, providing information about his facility. Several types of forms are used, depending on the functions chosen. The forms are described in Reference 10. An initialization form provides the basic system parameters required to start up the application, including the number of control and monitoring points, the user's power meter factor, the method used by the utility company in calculating peak demand, and the number of digital input/output cards used.

After the program is initialized, the system can start the power meter monitoring function and can generate consumption reports (mode 1 in Table 1). The system can start monitoring alarms and time of day functions (modes 2 and 3) as soon as the user has per-

sonalized data for at least one device and one monitoring point. As personalization progresses for more devices and monitoring points, they are brought under system control. Device control specifications can be changed on line by first deactivating system control of the device, then restoring it after the changes have been made. Other functions, such as device cycling (mode 4), demand control (mode 5), enthalpy control (mode 7), temperature conditioning of device cycling (mode 8), and device control override with logical combination of multiple monitoring point status, can be activated as their personalization data is completed, using operator commands and requests. See References 4 and 10 for details of these forms and for prompts and operator procedures.

Once a user has preplanned the control strategies for his facility and transferred the information to the personalization forms, he generally can start his application on the same day the Series/1 is installed and wiring is completed to the Series/1 through a manual control panel.

Description of results

The results of the System/7 approach, discussed above, point out the pitfalls of long program generation times and of dependence, for program modification, on knowledge of compilers, supervisor control programs, and a job control language. In most cases, up to two weeks were required for program preparation and installation. Almost all the System/7 installations involved program modification or application extensions, so the success of our approach on System/7 cannot be measured objectively.

For Series/1, the application has been installed at several building facilities, including some within IBM. In the first few installations, a hardware interface for the building and the computer (that is, a manual control panel) was already available since a Series/1 was replacing a System/7. The program was personalized and the system brought on line in one to two days in these cases. In one location, additional program functions were activated after the required temperature and humidity sensors were linked to the Series/1. Meanwhile the system was on line, performing normal facilities control functions.

Judging by data available from various installations to date, the reception of this approach to program automation has been enthusiastic. Compared with previously experienced delays in program installation, the reduction in system installation time has had a remarkable effect on the acceptance of computers by facilities management personnel.

A limited number of users have employed the program preparation facility to alter programs to match existing control panel in-

terfaces and add other program modules. In almost all cases, the alteration was accomplished easily. Some programmers have objected to the size restriction imposed on user programs because of limited diskette space as well as the fixed directory allocation. Once a programmer is convinced of the advantages of modular architecture, however, the objection to size restrictions tends to become less vigorous.

Oddly, in new Series/1 energy management installations, the responsibility for delays in installation startup has shifted back to hardware, specifically the interface panels and wiring that provide the control and monitoring path from the computer to the user's equipment. With proper planning and analysis of the building facility to provide personalization data for the program, it is not unusual for energy savings to be achieved the first week after hardware installation is complete.

CITED REFERENCES

1. *1800 Process Supervisory Program (PROSPRO/1800) Application Description*, IBM Systems Library, order number GH20-0261, available through IBM branch offices.
2. *IBM 1800 Time-Sharing Executive System Concepts and Techniques*, IBM Systems Library, order number GC26-3703, available through IBM branch offices.
3. *IBM System/7 Application Program Generator (APG/7) General Information Manual*, IBM Systems Library, order number GH20-1162, available through IBM branch offices.
4. *System/7 APG/7 Power Management Handbook*, IBM Systems Library, order number SB30-0617, available through IBM branch offices.
5. *Introducing IBM Facility Control/Power Management 2, 2M, and 3 General Information Manual*, IBM Systems Library, order number GH30-0094, available through IBM branch offices.
6. M. J. Shah, "A fill-in-the-blanks approach to generating a real time control system," *Proceedings of the IBM Automatic Programming Symposium, Yorktown Heights, 1975*, IBM Thomas J. Watson Research Center, Yorktown Heights, New York 10598.
7. *Understanding MSP/7*, order number GC34-0027, and *MSP/7 Installation Guide*, order number GC34-0031, IBM Systems Library, available through IBM branch offices.
8. D. L. Raimondi, H. M. Gladney, G. Hochweller, R. W. Martin, and L. L. Spencer, "LABS/7—a distributed real-time operating system," *IBM Systems Journal* **15,** No. 1, 81–101 (1976).
9. *System/7 Event Driven Executive Program Description/Operations Manual*, IBM Systems Library, order number SB30-0812, available through IBM branch offices.
10. *IBM Series/1 Facility Control/Power Management 2, 2M, and 3 User's Guide*, order number SH30-0119, and *Facility Control/Power Management 2 and 3 Application Program Utilities Reference Manual*, order number SH30-0177, IBM Systems Library, available through IBM branch offices.

The author is a senior industry analyst at the IBM Application Development Center, P.O. Box 2328, Menlo Park, CA 94025.

Reprint Order No. G321-5104.