Thomas J. Harrison
Bruce W. Landeck
Hal K. St. Clair

Evolution of Small Real-Time IBM Computer Systems

In parallel with the development of data processing applications for computers, effort was directed to other areas in which computers might provide benefits for the user. One early effort was the application of computers to the monitoring and control of industrial processes such as those used in oil refinery units, steel plants, and paper machines. Over time, these early efforts were generalized to a broader class of applications in which the computer was connected directly into an external process *which placed time response requirements on the computer system. These systems have become known as* real-time systems. *In this paper, the evolution of IBM small real-time systems is traced from the late 1950s to the present. Emphasis is placed on a few features and requirements which characterize these systems.*

Introduction

A significant portion of IBM's effort in the 1950s was devoted to computers which eventually merged into the single architecture of the System/360. Concurrently, however, other groups were exploring new opportunities for computers that resulted in several other computer sequences. Although general-purpose, these systems were optimized for characteristics different from those emphasized in the System/360.

One such sequence was aimed at moving data processing systems closer to data sources associated with the operational aspects of the users' enterprises. The initial application was the monitoring and control of continuous industrial processes, but, over time, it has broadened to encompass diverse operational applications such as energy management, discrete manufacturing control, laboratory automation, traffic control systems, and telephone toll ticketing. The small real-time IBM computers used in these applications are represented by the 1720, 1710, 1800, System/7, and, currently, the Series/1.

Both "real-time" and "small" are relative modifiers whose quantitative meanings are application dependent. In general, "real-time" means that the information processing, to be useful, must be completed before some outside event occurs. Depending on the application, the time available may range from a fraction of a second to hours. Similarly, a real-time computer need not be small. In particular, many of the early machines (*e.g.*, SAGE, Whirlwind) were physically large and powerful computers. Nevertheless, most real-time applications today utilize physically small computers that are in the lower range of computing power.

Requirements for small real-time computer systems

Although the initial trend was to use existing architectures in early small real-time machines, unique process control requirements were major forces in the evolution of today's small real-time computers.

Hostile environment Early data processing computers typically were installed in dedicated air-conditioned and filtered rooms which resembled a "clean room." For industrial process use, however, the computer must operate over a wide range of temperature and humidity in the presence of particulate and gaseous contaminants.

Reliability and unattended operation Many industrial plants operate 24 hours per day, 7 days a week, and so must the computer. This placed emphasis on reliability and ease of repair. Bid requests requiring availability in excess of 99.9% were not uncommon in the early 1960s. In addition, the computer had to be usable by plant operators who had little knowledge of computers. This placed great emphasis on ease-of-use and human factors long before these became the topics of interest that they are today.

Sensor input/output subsystems In order to directly connect the computer to process equipment, a number of special subsystems were required [1]. Many process instruments generate analog signals derived from sensors such as thermocouples and strain gauges. These signals must be converted into discrete digital representations to be used internally in the digital computer.

Feature configurations Even in a given industry and a given process category, there is considerable variation in the design of process equipment. Thus, the designer of an industrial computer system must provide the means for incorporating virtually any combination of features into a system.

Time response A key requirement is predictable time response; this requires architectural features that assist in the timely completion of a task and in switching from one task to another. The concept of interrupts to control the input/output equipment of a general-purpose computer was extended to include interrupt sources outside the computer.

Environment chronology

The computer business is driven by user needs and the capabilities of available technologies. As the designer or user perceives new uses, the evolution of related designs is affected. Initially, small real-time systems were intended primarily for industrial process control and this strongly influenced their characteristics. Over the past twenty years, however, our view of the application set has changed to the point that the Series/1 satisfies the needs of many different applications. It is useful, therefore, to consider a chronology of the environment over the past twenty or thirty years and how this affected the products which evolved.

- ***The 1950s: Years of experimentation***

By the mid-1950s, work had been started on using computers for the control of industrial processes [2, 3]. In early IBM studies, data-collection hardware was installed to gather actual process data from several plants. From these data, a mathematical model of the plant was created and utilized to optimize plant performance [4].

On the basis of initial studies, a development team was formed to build a "process control computer." The first decision was to choose between a 1400-series machine, then in development, and a small computer known as CADET for use as the central processing unit. CADET, which became the IBM 1620, was selected primarily because it was computationally oriented and the 1400 was a character machine. This was despite the fact that the early 1620 did not have a hardware adder (it used table look-up arithmetic) or built-in multiply [5]. Its variable-field-length data and its low cost, coupled with adequate performance, were winning factors. The resulting system, the IBM 1720 Process Control Computer System, had as its first users AMOCO in Whiting, Indiana, SOCAL in El Segundo, California, and E. I. du Pont in Wilmington, Delaware. These three pilot systems were installed in 1961, and the AMOCO and SOCAL systems were operational for many years.

With the experience gained on the 1720, IBM's first commercially available offering, the 1710, was defined to provide a lower entry price at less function than the 1720 to broaden the application set. This machine was announced in early 1961, and several hundred machines were installed during its sales life; a number are still in use after nearly twenty years. The 1720 and 1710 were technically successful machines that provided rapid learning, by both IBM and the user, as to the needs of process control. Coupled with changes in technology, this set the stage in the early 1960s for a machine which more nearly satisfied the original goals of the study teams and designers of the 1720.

- *The 1960s: Industrial computers come of age*

Numerous industrial control projects were attempted in the early 1960s. Some were highly successful and some failed. It was recognized that the mathematical-model approach initially pursued was complex and often required vast resources. Alternatives were sought by university and industrial researchers in an effort to ease the task of applying computer control, since demonstrated advantages clearly existed.

Of the approaches, Direct Digital Control (DDC), championed by Dr. T. J. Williams of Monsanto (currently with Purdue University), probably was the most influential in guiding the technical evolution of process control computers [6]. The then-current (and still used) analog control method utilized a specialized analog computer, known as a controller, which continuously solved the linear equation

$$Output = K_1 e + K_2 de/dt + K_3 \int e dt,$$

where e, the error, is the difference between the actual process parameter value and the desired value (the set-point), and the Ks are constants. A dedicated analog controller was used for each process "loop" consisting of a sensor (possibly more than one), which provided the current value, and a process actuator (*e.g.*, a valve) driven by the controller output.

The limitations of using the linear analog controller to control a nonlinear process were recognized by Williams and others. In the DDC concept, the computer samples the loop input, calculates the digital equivalent of the analog controller equation, and drives the process actuator directly—thus the name "Direct Digital Control." At computer speeds, a single computer can handle hundreds of loops on a time-shared basis. In addition, the computational capabilities of the computer simplified the implementation of advanced control algorithms, such as nonlinear and adaptive control, without extensive process hardware changes. It initially was estimated that DDC could be justified if the amortized computer cost was less than $1000 per loop.

This target price provided an elusive goal for the designers of industrial control computers for most of the decade. The technology was not yet ripe for this low cost and the broad acceptance of DDC has only recently been realized.

Although DDC was a persuasive force, other factors were important in shaping the systems of the 1960s. Prices were continuously decreasing but process control computers still were expensive and proved economically attractive only on rather large processes. But these large processes were complex and economic justification often required that the total process, or even several small processes, be controlled by the same computer. This led to the early development of executive programming systems which incorporated multiprogramming concepts, and to the early use of high-level languages and application packages such as FORTRAN and PROSPRO for process use.

It was in this environment that the IBM 1800 was developed—a new, high-function process control computer. It also was recognized, however, that related application areas such as high-speed data acquisition for wind tunnels and laboratory automation could be satisfied simultaneously by such a machine. Here, the emphasis was on data input rather than on feedback control. The 1800 was designed to expand applications into new areas.

The CPU used as the base for the 1800 was a small, scientifically oriented machine, announced as the IBM 1130 [7]. It was a 16-bit binary machine, to which the 1800 added parity, storage-protection bits, and additional features such as the preemptive priority interrupt structure [8].

The 1800 was announced in November 1964, and was a very successful machine: several thousand were produced and many remain installed today. The installation of the last new 1800 is scheduled for 1981, almost seventeen years after its initial announcement!

- *The late 1960s: A period of transition*

Attention in 1966 was directed to an 1800 follow-on, dubbed the 18XX. A new factor was emerging in that a number of very simple, very low-cost computers were becoming available. The low cost often allowed the computer to be dedicated to a single task, thereby greatly simplifying the needed programming system and the programmer's work. These low-cost machines were known as "minicomputers."

Two factors became obvious in the initial planning. One was that the 1800 architecture and its programming support could not be subset to the point at which significant cost benefits could be derived. Secondly, process control applications were becoming a proportionally smaller and slower-growing segment of the new emerging opportunities. As a result, the 18XX as a standalone system was abandoned, and attention was turned to alternative system configurations.

The new system was to be merely a front-end controller, without local storage or significant computational capability. It would maintain, however, the flexibility of the 1800 in terms of feature configuration, sensor input/output subsystems, and modular industrial-style packaging. The controller architecture was relatively primitive and great emphasis was placed on speed.

As the design progressed, performance requirements dictated that the system have a small amount of local storage, and the evolution to a standalone system had begun. It continued until the system became capable of standalone operation and had all the attributes of a small real-time computer system. The decision was made to incorporate the fastest semiconductor memory available (400 ns), despite lengthy debates and concern about the acceptability of volatile storage in industrial control applications.

The initial host-dependence concept also was reflected in the software. All program preparation was to be performed in a host using cross compilers and assemblers. Since many new applications were anticipated, the

programming system consisted of a set of facilities which could be combined to build an optimized control program for each application.

The system was announced as the System/7 in late 1970 [9]. Initial reaction to the machine was poor due to the lack of native programming support, the requirement for host support, and modularity increments which resulted in expensive small configurations. Over the next few years, native program preparation was provided, and considerable resource was devoted to application development. The system ultimately was successful, but in areas not entirely anticipated by its designers, and, as intended, mostly in areas other than process control. In retrospect, it was an expensive machine, ahead of its time in its orientation to host support and hierarchical interconnection, which often was at a disadvantage when compared to less sophisticated, standalone minicomputers.

- *The 1970s: The "minicomputer" era*

By the early 1970s, it was clear that the minicomputer would be the pervasive small machine of the decade. It was intended that the System/7 evolve to meet these changing conditions by extending its function and reducing cost. Analysis showed, however, that the controller orientation of the System/7 was not easily extendable to the higher-function instructions becoming common in minicomputers. It finally was decided to develop a new architecture which was of much higher function, implementable as a family of models, and extendable, while still retaining some important features of the System/7.

The resulting Series/1 family was announced in 1976 [10], and was hailed in the press as "IBM's entry into the minicomputer business." Looking back, it might more accurately be seen as a "reentry," since the IBM 1620 and 1130 clearly were minicomputers before the name was coined in the late 1960s. Since 1976, numerous Series/1 enhancements have been announced, including new processors, additional peripherals, and two significant operating systems.

Evolution of engineering features

The architecture and most features of small real-time computer systems are not unique or remarkable when compared to those of large data processing machines. In many cases, new functions or features on small machines are well known on larger computers. The design innovation often is in providing such a function on the small machine within severe cost and space constraints. Three areas, however, always have been of particular concern to the designers and tend to characterize small real-time systems. These are the sensor input/output subsystems, features for time responsiveness, and physical packaging.

- *Analog input subsystem*

The sensor I/O subsystems necessary to make measurements of, and provide control signals to, the physical world are lumped into four main classifications: analog input, digital input, analog output, and digital output. Of these, the analog input (AI) subsystem always has represented the most challenging design task, primarily due to the difficulty of accurately handling low-level analog signals in the presence of high-level electrical noise.

Two common analog process variables are temperatures measured with thermocouples and forces measured with strain gauges. The AI subsystem must be able to read this information without adding expensive equipment for each input. Unfortunately, the signals from these devices are differential signals in the low millivolt range that may be imposed on a common voltage to ground (common mode voltage, or CMV) of many volts. To detect temperature changes of a few degrees with a thermocouple, for instance, it may be necessary to measure microvolt signal differences in the presence of 5-volt CMV and random process noise. In analog control systems, heavy filtering is used to eliminate noise effects. To utilize the speed of a computer, however, individual readings must be taken in milliseconds or less. Thus, the shared equipment must have a wide bandwidth, and consequently is subject to errors from the high-frequency and transient noise common in an industrial plant. To achieve a total subsystem error of less than 0.1%, each of the units in the AI subsystem must contribute less than 0.01 to 0.05% error.

The first unit in the subsystem is a multiplexer required to switch the relatively expensive analog-to-digital converter (ADC) between hundreds or thousands of low-level analog inputs. For the 1710 and 1720, IBM pioneered with the C. P. Clare Company in the development of miniature card-mounted mercury-wetted relays that matched computer packaging and drive requirements [11]. These relays provided the basis for a low-level signal multiplexer with excellent signal switching characteristics.

The 1800 multiplexer utilized similar card-mounted dpdt mercury-wetted-contact relays in a "flying capacitor" configuration [12]. In this approach, a 300- to 600-μF nonpolarized capacitor is connected between the armatures of a dpdt multiplexer relay. With the relay not actuated, the capacitor is connected to the differential signal input through two resistors, forming a balanced single-pole RC filter. When actuated, the relay switches the charged capacitor to the high-impedance input of the ADC. During the measurement period, therefore, the

subsystem physically is disconnected from the process, providing excellent isolation from external CMV and noise sources.

The 1800 system also was designed for the high-speed data-acquisition application requiring sampling rates of thousands of readings per second. For this, the "Bright Switch," a back-to-back inverted matched bipolar transistor pair, provided linear switching with a reasonably low offset voltage [13]. In the System/7 and Series/1, the inherently low offset voltage of FET switching technology provides for high-speed electronic multiplexing down to the 50-millivolt full-scale range.

Unbalanced leakage paths also cause errors due to the conversion of CMV into a normal mode signal. To minimize this, leakage resistances to ground must be maintained at more than 1000 megohms in a 95% relative humidity ambient environment. This finally was achieved in early systems through empirically determined card layouts with maximized leakage paths, by special card coatings, and by the use of desiccants in the multiplexer enclosure.

In addition to an accurate multiplexer, the AI subsystem requires an ADC to digitize the low-level signals in the same difficult environment. The conventional approach was to amplify the low-level signal to a value compatible with digital technology, usually ±5 volts full scale. However, high-speed, high-precision dc amplifiers were difficult and expensive devices to build in the 1960s, so the 1710/1720 utilized a unique low-level ADC that performed digitization at the low millivolt level in an electrically isolated front end [14]. This reduced error sources by converting to a digitized value as early in the circuit flow as possible and obviated the need for a separate dc amplifier.

To provide high performance at lowest cost, IBM offered the "Dual-Ramp" ADC in 1966 [15]. This design uses a "ramp-up/ramp-down" technique that converts the analog signal into a digital count in a manner that causes most component inaccuracies and drifts to cancel. It produces a result that is dependent primarily on the long-term stability of the precision reference voltage and the short-term (millisecond) stability of all other components. This technique has been used in the vast majority of low-cost ADCs and digital voltmeters since the early 1970s. Since the technique is limited in speed, however, the System/7 developed the "Triple-Ramp ADC." This technique utilizes the same low-cost error-compensation concepts of the dual-ramp ADC, but incorporates a high-speed "slew" and low-speed "trim" approach to provide a much higher conversion speed [16].

- *Time response features*

The primary architectural characteristic that distinguishes small real-time systems from many other machines is the interrupt structure. The real-time system must be responsive to process events. Typically, the most important CPU responses are to exceptional conditions which appear very infrequently, such as an imminent unsafe condition.

The characteristics of preemptive priority interrupt structures have evolved over the years to achieve a twofold purpose: First, they ensure that the highest-priority jobs are initiated promptly, without requiring completion of a lower-priority task currently executing. Secondly, they ensure that the CPU is spending more time on responding to events, rather than on analyzing just what the event is. Without any interrupt structure, the system must constantly poll under program control to look for these infrequent occurrences and, with hundreds of potential interrupt sources, this overhead could cause significant performance degradation. In contrast, the early general-purpose systems only had to contend with a dozen or so interrupt sources, usually associated with input/output equipment.

The IBM 1720 provided an interrupt system by adding hardware, outboard from the 1620 mainframe, to continuously scan nineteen internal and up to fifty process-interrupt signals [17]. The interrupt system could be masked and unmasked as a whole, and it was a preemptive system in that interrupts could be nested (a second interrupt could interrupt the servicing of a first interrupt). Hardware vectoring of the interrupt was provided to cause an automatic program branch to a service routine uniquely associated with each interrupt source. However, there was no multilevel hardware priority, so that each service routine had to decide whether an interrupt should be handled immediately or deferred.

The 1710 provided a less complex, nonpreemptive system in which many events could capture the attention of the CPU software and an interrupt routine had to be completed before any new interrupt could be recognized. Identification of the interrupt source and the storage location of its service routine was determined by customer-written software. This approach did not require the outboard hardware of the 1720 and was much lower in cost—a prime consideration in designing the 1710. This simple interrupt system was comparable to "priority processing" in contemporary data processing systems.

Based on the experience with the 1720 and 1710, the IBM 1800 system in 1964 provided a true preemptive interrupt structure with up to 24 priority levels, each with

16 sublevels. While servicing an interrupt, the system could be interrupted only by a request on a higher level, and interrupts could be nested indefinitely. Interrupts on lower levels were queued until all higher-level interrupts had been serviced. Each interrupt level could be masked individually by software to block interrupts from that level. When accepted, an interrupt caused a hardware branch to a service routine associated with the level.

In these systems, the basic interrupt action was to cause a forced branch in the program and to retain the address of the next instruction so that the interrupted program could be resumed after servicing the interrupt. In most cases, considerable "housekeeping" was needed to save the intermediate results of the interrupted program before servicing the new interrupt. Typically, all registers, accumulators, and indicators would have to be stored and return linkages established. This housekeeping delayed the response to the interrupt.

In 1970, on the System/7, IBM first utilized the powerful concept of duplicating the complete register set for each interrupt level. A separate instruction address register, eight general-purpose registers, and registers for status information were provided for each of four priority levels. This was equivalent to providing multiple processors with a common set of controls and main storage. Thus, no intermediate results were saved explicitly; an interrupt simply activated a new set of registers. Consequently, within 800 ns (two storage cycles) of the occurrence of an interrupt, the processor could be servicing the interrupt. The four System/7 interrupt levels each had 16 sublevels. The sublevels provided direct hardware vectoring for up to 16 interrupt sources on that level, avoiding software analysis to determine the source of the interrupt.

Another very powerful concept introduced in the System/7 was the ability to change the priority of interrupting devices under program control. In previous systems, the priority was hardwired at installation. In many real-time applications, however, the true importance of an interrupt depends on the process state at that instant. In the System/7, the software, through a PREPARE instruction, could modify the priority, setting both the level and sublevel for each interrupting device. The Series/1 system provides essentially the same interrupt structure and functions.

- *Industrial packaging*

The cost of the physical package is significant in small systems. Beyond its primary purpose of providing means to mount and enclose the electronic components, it must satisfy a range of requirements from withstanding vibration to aesthetic appeal. In addition, there is a strong economic incentive to use the mass-produced packaging technology of larger IBM systems.

Modularity An important packaging requirement is modularity—so that the broad range of applications can be satisfied with a single system. For example, some applications require hundreds of analog inputs, some only a few, and some none at all. Since the beginning, the package design has been oriented such that space, power, and cost need not always be provided for the maximum system, only to be left unused by the smaller typical system. The 1710/1720 systems followed the conventional computer physical design in that almost every optional feature had a fixed reserved location. The sensor I/O termination blocks, matching cards, and input-signal multiplexing were the only areas where a modular, building-block design was incorporated. The 1800 system also utilized IBM standard boards and gates, but was given added flexibility by use of "floating features." In this approach, certain gate locations (the same space and power) could house one of several different sensor I/O features. The floating features, however, caused appreciable difficulties in specifying cable lengths, providing build and test instructions, servicing the system, and determining the validity of a customer order.

The System/7 introduced the first truly modular system in the evolution. Rack-style enclosures with common power supplies were offered, providing two, three, six, nine, or twelve modular subframe positions, with the dc power and system internal interface bussed to each position. The CPU and features were housed in individual subframes which, with few restrictions, could be used in any position. The approach proved to have its limitations, primarily in penalizing small configurations with relatively high cost. The common power supplies, designed to power two, three, or six modules, resulted in extra costs when all module positions were not utilized.

Advances in technology allowed the Series/1 to carry the modularity/flexibility concept much further. The use of high-frequency switching techniques allowed small low-cost power supplies to be incorporated in most individual modules, thereby closely matching power capacity to actual requirements. The use of higher-density, lower-unit-power integrated circuits provided the opportunity for housing more function in a module, thereby amortizing package cost over several features. A feature which required a separate subframe in System/7 often requires only a single card in Series/1.

Interestingly enough, continuingly increasing integrated circuit densities are tending to resurrect the System/7

problem of high packaging costs for small configurations. When a total CPU, 128K bytes of storage (K = 1024), and 1 input-output adapter all fit on two or three cards in a module which can accommodate about twenty cards, the package cost represents a significant portion of the total cost of a small configuration. Although it is likely that industry standard rack enclosures will continue to be used, it is obvious that continued evolution and innovation in packaging can be expected.

Industrial environment The 1720 system was designed principally for the heavy industries of petroleum refining, steel plants, power generation, and the like. It was the first complete IBM system designed for such an industrial environment. The enclosure was built of 10-gauge steel, designed to survive an inadvertent impact from a fork lift truck. The operating limits were specified at 40° to 122° F and 0 to 95% relative humidity (85° F maximum wet bulb temperature). The entire system had a vibration specification of ±0.25 G, which required testing 2500-pound units at this vibration level. In addition the covers were gasketed and closed with screw locks to allow them to be maintained at a slight positive pressure to exclude hazardous or contaminating gases.

The 1800 system was installed in a number of locations that had very corrosive atmospheres, particularly in paper pulp mills. Initially, a few 1800 systems required mechanical replacement in less than a year due to corrosion which would destroy the copper interconnections. A special impervious coating was developed for all circuit cards and boards to protect them from rapid deterioration.

As a result of the 1800 experience, extensive testing was done to establish the quantitative corrosive effects of several common industrial gases and airborne particulates on standard IBM printed-circuit cards, boards, and mechanical devices. Several severity categories were identified for specification purposes based on the long-term effect on IBM equipment.

The System/7 was designed to function in these environments without the need for special coatings on individual cards and boards. An optional feature, the "Internal Air Isolation" (IAI) feature, offered a nonrefrigerating heat exchanger mounted on top of the enclosures. Recirculated air inside the machine enclosure passes through a finned air-to-air heat exchanger to provide cooling without an interchange of inside and outside air.

Several environmental monitoring devices were developed in cooperation with the Field Engineering Division. In a proposed installation suspected of having high contamination levels, the Installation Planning Representative installed the device at the site prior to system installation. If the device showed unacceptable levels of gaseous contaminants, the use of the IAI feature was required to validate the rental or maintenance agreement.

Except for standard card coatings, no special provisions for gaseous and particulate contaminants were taken for the Series/1. Several factors entered into this decision: The IBM semiconductor component packages had been improved and were less susceptible to corrosive gases. The fans in each module produced sufficient air velocity that particulates would not collect on the cards, even though the filters had been eliminated. Finally, an analysis of applications showed that extra expense for additional protection was not warranted since relatively few systems would be subjected to hostile environments. The air analyzer is still used to provide guidance to the user in identifying severe environments so that appropriate measures, such as special air conditioning, can be employed.

Evolution of IBM real-time operating systems

The IBM 1720 did not have an operating system, as such; the operating system functions were integrated directly into the application program. This increased the complexity of the application program, limited flexibility, and increased maintenance problems. The lack of a separate operating system on the 1720, however, clearly established the need for operating systems on future real-time hardware products. As a result, the Basic Executive System was developed for use on the IBM 1710 in 1962. This was a minimum-function dedicated operating system which provided an interrupt handler and I/O driver support for the real-time input/output channel. Its use was minimal, however, because a new operating system, Executive II, was developed for the 1710 in 1963.

Executive II was a major extension in function and design and was the first IBM system to provide disk residence for user and system programs. It provided a set of error recovery routines for the I/O devices, as well as facilities for automatically exercising and testing I/O devices on line so as to enhance system availability. Further, Executive II provided for real-time scheduling of user application programs based on time interval, time-of-day clock, and external events. The concept of using external events (interrupts) as the scheduling stimulus was a significant advance over the batch orientation of then-current business-oriented computers. Application program preparation for Executive II was done off line using SPS-II, a symbolic assembly language. Executive II was a widely used and successful operating system.

The third operating system for the IBM 1710 was released in 1964. It was called the FORTRAN Executive and was a significant milestone in the evolution of real-time systems, since, for the first time, the high-level language FORTRAN was available for use by the real-time application programmer. While FORTRAN Executive had limited acceptance because it came late in the life of the IBM 1710, it pointed the direction for the use of high-level languages in real-time applications. All subsequent real-time systems have provided FORTRAN or other high-level languages for application development.

In November 1964, the Time-Sharing Executive (TSX) system was announced in conjunction with the IBM 1800. When delivered in early 1966, TSX was the first IBM operating system to provide real-time support with concurrent background batch capability. This allowed the user to prepare, compile, and link into real-time application programs without taking the system off line. The background batch capability further allowed commercial and engineering programs to be compiled and executed concurrently with real-time programs. FORTRAN was the primary user real-time language under TSX. The use of assembler language could be relegated to those areas where the performance or size of the generated code from FORTRAN was unacceptable.

A specialized control program facility, called PROSPRO, was developed in 1966 to reside on top of TSX. It was intended primarily for use in the control of continuous processes such as those found in an oil refinery. PROSPRO provided built-in functions familiar to the process engineer, such as the controller equation described earlier, which could be invoked using terminology known to control engineers. In addition, it featured a "fill-in-the-blanks" programming technique whereby data from forms prepared by the control engineer were used to create tables that determined program sequences and computer responses needed to control the process. It allowed the control engineer to utilize the system with very little specialized knowledge of computers [18]. In its original release, it effectively supported the supervisory (setpoint) control philosophy, but a later release (PROSPRO II) for MPX provided DDC.

In 1967, IBM announced a new operating system for the IBM 1800, called the Multiprogramming Executive System (MPX). It was the first IBM system to provide multiple fixed partitions into which programs could be scheduled on the basis of external events, time-of-day clock, time interval, operator command, and the batch job control language [19]. MPX was capable of controlling multiple independent real-time processes, with batch program preparation and business and engineering applications running concurrently in the background.

MPX introduced a number of new facilities to the real-time environment. For example, it was the first IBM real-time system that effectively could support communications to other systems and terminals using either BiSync or Stop/Start protocols. MPX also supported file sharing between systems. This allowed two MPX/1800 systems to be connected to an IBM 2311 Disk File and concurrently use it for communications between the systems and for shared-data-set residence. In terms of support facilities, MPX provided the first Macro Assembly Program for use on an IBM real-time system.

MPX made significant advances in the area of error recovery, an important consideration for continuously running processes. First, while MPX was a disk-resident system, it was not disk dependent and the system continued to function if the disk failed. All of MPX's error-recovery facilities still were operational with the disk down, as well as any user routines that were main-storage resident. The system also was designed so that a customer engineer could take a device off line, work on it, exercise it, and then bring it back on line after repair without having to take the system down. Multiple levels of automatic backup were provided for I/O devices upon failure. Also, when power returned to the system after a power failure, MPX would automatically restart the system and the application.

MPX was the culmination of all the development knowledge and experience obtained starting with the Basic Executive System back in 1962. In 1968, the mission was moved from San Jose, California to Boca Raton, Florida, resulting in an almost completely new development group. That, coupled with the initial controller orientation of the next system, resulted in a new series of real-time operating systems which evolved during the 1970s.

In October of 1970, the IBM System/7 was announced with the Modular System Program/7 (MSP/7) as its operating system. As noted earlier, System/7 was designed to rely on the System/370 for host support functions, such as program preparation. As such, the first release of MSP/7 was designed to be only a kernel for a real-time operating system. The initial level of function was similar to, but less than, that of Executive II on the IBM 1710. The first release primarily was a collection of modules which the user combined, with some additional programming, to create a control program tailored to the application. Due to its assumed host dependency, MSP/7 did not support user or system program residency on a disk, but did support communications back to a System/370 host. Program preparation was provided by the System/370, with only minimal capability off line on the System/7.

Version 5 of MSP/7 was released in 1972 with two major new facilities. The first was Symbolic File Support (SFS) and the second was the Disk Support System (DSS/7). These new facilities provided for program transients resident on disk and a monitor for off-line batch operation. Functionally, Version 5 was similar to the FORTRAN Executive system on the IBM 1710. Further, the primary applications for MSP/7 were now in the communications systems area. In 1974, Version 9 of MSP/7 was released and provided for a multipartition monitor similar to MPX, essentially completing the evolution of MSP/7.

During the 1974–1975 time period, three other operating systems were developed for the System/7 by groups other than the MSP/7 development organization. These were the Event-Driven Executive (EDX/7), the Application Program Generator (APG), and the Application Monitor. EDX/7 was developed at the IBM San Jose Research facility. It was originally designed as a laboratory-automation real-time system, but then was generalized into IBM's first real-time interactive system. APG was developed at the Application Development Center in Palo Alto. It was primarily oriented toward the continuous process control environment. Its significance is that it was the first IBM real-time system to provide PL/I as the primary user interface to the real-time facilities [20]. The Application Monitor was developed in Boca Raton, and introduced several new concepts. Primary among these was the late binding of resources to an application program [21].

The IBM Series/1 was announced in November 1976 with the Control Program Support System (CPS). This system was similar in concept to Release 1 of MSP/7. In April 1977, the second system announced for the Series/1 was the Real-time Programming System (RPS) [22]. RPS is a full-function, real-time operating system supporting dedicated, host, and interactive environments. The design of RPS was heavily influenced by the Application Monitor System, MSP/7 Release 9, and MPX/1800.

RPS is strong in the communications area with support of Start/Stop, BiSync, and SDLC communications protocols. It also provides a multiple terminal management facility to aid the user in communication and interactive applications. It provides great depth of function in the areas of multiprogramming, multitasking, data management, and program preparation. The program preparation facility under RPS is interactive in design and supports FORTRAN, PL/I, COBOL, BASIC, and Macro Assembly languages. Further, the system provides for both late and early binding of system resources to application programs. RPS is being utilized in commercial, communications, and real-time process applications.

The Event-Driven Executive (EDX) was announced on the Series/1 in September 1977. This was the first time an IBM control program had bridged totally different hardware architectures. The first release of EDX was compatible with EDX/7 and has been enhanced to support FORTRAN, COBOL, and PL/I through its user interface [23]. For the user who does not require the power, flexibility, and full function of RPS, EDX provides an alternative with the emphasis on ease of use and performance.

Contributions to real-time applications

Throughout the evolution of these IBM real-time systems, there has been considerable effort toward understanding and supporting particular computer applications in industry. Literally hundreds of these applications have been developed by IBM singly or in cooperation with customers and many have contributed to advances in fields other than computers. It is not possible to exhaustively examine these contributions in this paper but several will be noted on the basis of their significance or other unusual aspects.

Shortly after the decision to build the 1720, several technical support and research groups were established in San Jose. The groups participated in several early studies and made contributions in the area of mathematical modeling and adaptive control [24]. As part of the activity, a small distillation column, controlled by an 1800, was installed at the plant site in San Jose. This was used to develop and test control algorithms [25]. It was also used for human-factors studies relating to consoles for use in the process industries.

In addition to many process control applications, the 1800 was used in oceanographic research aboard seagoing vessels. It provided the capability of collecting and analyzing data from towed transducers. Previous methods had involved the recording of data on magnetic tape which was returned to port for reduction and analysis.

Following the System/7 announcement, a major application development effort was undertaken. Included was the development of the IBM Bridge System [26]. The system included a ruggedized System/7 interconnected with the ship radar, autopilot, and navigational receivers, and a special operators' console. The primary application of the system, and its main justification, is to assess the possibility of collision between the ship and other ships in the vicinity. The system digitizes the radar signal, automatically detects targets, and determines their courses and speeds. Using this information and a collision-assessment algorithm, the ship's officer is provided with a prioritized display of potential collisions. In response, the officer can enter a tentative course or speed correction

and, through simulation, determine if this action will avoid the collision without creating another potential collision. If so, the officer initiates the action to alter the ship's motion. In addition to collision assessment, other applications include position fixing, route planning subject to longitudinal and other constraints, route tracking, and control of the autopilot utilizing adaptive control algorithms that take into account the state and characteristics of the ship (*e.g.*, loaded or unloaded, minimum turning radius, etc.).

Soon after announcement of the System/7, it was recognized that it had potential in the communications industry, primarily because of its high speed and unique interrupt structure. As a result, a dedicated development group was established to explore applications, particularly in connection with telephone central offices. A number of applications were developed and sold, including the use of the system to record initiation and termination times of toll calls to use as the basis for billing. Specialized operating systems and application programs were developed for this and related communications uses [27]. This work has been continued and now utilizes the Series/1.

Of particular significance in this application development is that it was of benefit to both the user and IBM. Particularly in the early days, the potential of the computer was not understood by all engineers in the process industries. Through studies and application development, industry became aware of the potential and how the computer could be used to improve the control of processes and thereby improve the economic return for the customer. For IBM, it offered an opportunity to sell systems and, perhaps more importantly, to understand the needs of the industries so that future computers could more efficiently or easily satisfy applications.

Today, many users have the necessary expertise to apply computers with little or no assistance from the vendor. Nevertheless, application development continues as an aid to multiple-unit marketing and to make the use of computers easier for businesses and for individuals who are not professional programmers or engineers.

Projections for the future

The evolution of IBM real-time systems has not been a simple straight-line extrapolation of the past. It has been influenced by a large number of economic and technical factors. It is likely that the future will have the same characteristics, complete with a few dead ends and false starts. There are few, if any of us, who in 1958 when we started on this evolution could have foreseen where we would be today; the rate of change of technology has astounded us all. Thus, even assuming a linear or logarithmic extrapolation, the computer world twenty-five years from now on the fiftieth anniversary of the *Journal* cannot be predicted by these authors.

Some observations can be made which point a direction to the future. The "minicomputer" was spawned some fifteen years ago as a minimum-capability stored-program machine, driven by the desire for low cost and constrained by minimizing circuit counts. It was devoid of any significant software support and it flourished in an environment of experimentation as entrepreneurs sought to exploit it. It has evolved to a sophisticated machine with significant software support and computational power. It is pervasive throughout industry and is produced by the tens of thousands every year.

Now the "microprocessor" has appeared, as a result of the capability of semiconductor technology. It appears that the evolutionary cycle of the minicomputer is being repeated, only at an accelerated pace. The first microprocessor on a chip was rudimentary compared to its minicomputer predecessor. It had 4-bit words and only a few instructions and was very slow. Like the original minicomputer, the desire was low cost and the constraint was the number of circuits that could be fabricated on a single silicon chip with economic yields. But in a few years, more circuits became available as a result of semiconductor technology advances and the 8-bit microprocessor appeared. Soon after, the 16-bit microprocessor was announced and is now in common use, with 32-bit designs close behind. In the 1950s, computers were built one at a time, in the 1960s it was by the thousands, and in the 1970s by the tens of thousands. The microprocessor has already reached the hundreds of thousands and is expected to reach millions in a few years.

The software is lagging but, again, the rate of progress is much faster than in the case of the minicomputer. High-level languages are available for the microcomputer and operating systems have been developed. Although the level of sophistication is less than that of minicomputer software, there is no reason to believe that it will remain so for long.

What, then, is the future of the small real-time computer? It already is obvious that many of the real-time functions are being assigned to microprocessors buried in instrumentation, terminals, automobiles, television sets, and other equipment. This will continue as the general-purpose programmable nature of the computer and the low cost of the microprocessor make possible endless applications. The minicomputer will continue, but often in the role played in the past by the larger data processing host computers. Furthermore, the microcomputer has

become a component available for incorporation in larger computers as a replacement for random logic. Its existence as a component in a bigger system may or may not be visible to the programmer or user of the system. When visible, it could provide an opportunity for parallel processing which will enhance the computing power of the system. But it also will require greater understanding on the part of the programmer concerning the control of concurrent processes. It is likely that new programming techniques and tools will be developed to assist the programmer in this control.

Although people in general are accustomed to bills, letters, and mailing lists produced by computers, they are not computer programmers in the traditional sense, and never will be. The challenge of the future for computer companies is in making computers available to the general public without requiring that they become experts in programming. The computer must be as easy to use as the telephone and must have the same transparency that masks the complex equipment, the call-routing algorithms, and the like. The challenge of the future, therefore, is human factors applied to the use of computers. With a friendly user interface, the power of the computer as an information processor will be available to everyone as a means of increasing productivity and enhancing the quality of life.

Acknowledgments

The three authors have been involved in most of the evolution described in this paper. Rather than trust their limited view and memory, however, they interviewed several dozen key participants in this evolution. The contributions of these interviewees are acknowledged, along with those of the hundreds of other people whose innovative thinking and hard work made these systems a reality.

References

1. T. J. Harrison, *Minicomputers in Industrial Control*, Instrument Society of America, Pittsburgh, PA, 1978.
2. T. M. Stout, "Computer Control of Butane Isomerization," *ISA Journal* **6**, 98-103 (1959).
3. J. R. Middleton, "Chocolate Bayou Instrument and Computer Systems," *Proceedings Texas A & M 18th Annual Symposium on Instrumentation for the Process Industries*, January 1963, College Station, TX, pp. 83-90.
4. R. H. Crowther, J. E. Pitrak, and E. N. Ply, "Computer Control at American Oil," *Chem. Eng. Progress* **57**, 39-43 (1961).
5. *IBM 1620 Central Processing Unit, Model 1*, Order No. A26-5706, available through IBM branch offices.
6. T. J. Williams, "Direct Digital Control Computers—A Coming Revolution in Process Control," *Proceedings Texas A & M 19th Annual Symposium on Instrumentation for the Process Industries*, College Station, TX, January 1964, pp. 70-81.
7. *IBM 1130 Computing System Functional Characteristics*, Order No. A26-5881-6, available through IBM branch offices.
8. *IBM 1800 Data Acquisition and Control System Functional Characteristics*, Order No. GA26-5918-9, available through IBM branch offices.
9. *IBM System/7 Functional Characteristics*, Order No. GA34-0003-7, available through IBM branch offices.
10. J. D. Schoeffler, *IBM Series/1, The Small Computer Concept*, IBM Corporation, Atlanta, GA, 1978.
11. A. J. Koda, "Switching Assembly," U.S. Patent 3,076,878, 1960.
12. T. J. Harrison, *Handbook of Industrial Control Computers*, John Wiley & Sons, Inc., New York, 1972, pp. 223-241.
13. R. L. Bright, "Junction Transistors Used as Switches," *AIEE Transactions on Communication & Electronics* **74**, 111-121 (1955).
14. H. L. Funk, T. J. Harrison, and J. Jursik, "Converter Digitizes Low Level Signals for Control Computers," *Automatic Control* **18**, 21-23 (1963).
15. C. H. Propster, Jr., "Analog to Digital Converter," *IBM Tech. Disclosure Bull.* **5**, 51-52 (1963).
16. H. B. Aasnaes and T. J. Harrison, "The Triple Integrating Ramp ADC," *Electronics* **41**, 69-72 (1968).
17. *IBM Reference Manual, 1720 Control System*, Order No. A26-5512, available through IBM branch offices.
18. *PROSPRO II (TSX/1800) Process Systems Program Functional Description*, Order No. GH20-4420, available through IBM branch offices.
19. *IBM 1800 Multiprogramming Executive Operating System Introduction*, Order No. GC26-3718, available through IBM branch offices.
20. *Understanding APG/7*, Order No. SH20-9513, available through IBM branch offices.
21. J. G. Sams, "Low Cost Systems Using Multiple Processors," *IBM Tech. Disclosure Bull.* **19**, 2879-2883 (1977).
22. *IBM Series/1 Real-time Programming System Version 4 Design Guide*, Order No. SC34-0242-1, available through IBM branch offices.
23. *IBM Series/1 Event-Driven Executive System Guide*, Order No. SC34-0312-2, available through IBM branch offices.
24. R. M. Bakke, "Adaptive Gain Tuning Applied to Process Control," *Proceedings Instrument Society of America 19th Annual Conference and Exhibit*, New York, October 12-15, 1964.
25. P. E. A. Cowley, "Pilot Plant Puts DDC to the Test," *Control Engineering* **13**, 53-56 (1966).
26. *IBM System/7 Maritime Applications/Bridge System General Information*, Order No. GA34-0010-1, available through IBM branch offices.
27. *IBM System/7 Centralized Automatic Message Accounting Computerized (CAMA-C)*, Order No. SA34-1526, available through IBM branch offices.

Received October 23, 1980; revised January 6, 1981

Thomas J. Harrison and Hal K. St. Clair are located at the IBM Information Systems Division laboratory, P.O. Box 1328, Boca Raton, Florida 33432. Bruce W. Landeck is located at the IBM General Systems Division headquarters, P.O. Box 2150, Atlanta, Georgia 30301.