Copyright © 1977 American Telephone and Telegraph Company
THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 56, No. 2, February 1977
Printed in U.S.A.

1A Processor:

Organization and Objectives

By R. E. STAEHLER

(Manuscript received July 16, 1976)

This paper presents an overview of the 1A Processor in terms of objectives, design philosophy, major features, and applications in the Bell System. It also serves as an introduction to the detailed technical papers that follow.

I. INTRODUCTION

A steady and rapid increase has occurred in urban and toll traffic and service features during the past decade. This has highlighted the need for large, high-traffic capacity electronic switching systems for local, tandem, and toll applications in the Bell System. The 1A Processor[1–3] has been developed to meet this high-processing capacity need. It is a stored program processor having a basic instruction execution speed of 700 ns, a memory access speed of 1400 ns, direct-memory access to bulk memory and data links, and a peripheral bus system and associated controls for interfacing to the switching networks. The technology is based upon beam-leaded integrated circuits that are used for most logic functions. A common processor was designed for use in a variety of applications to minimize the expenditure of both development and manufacturing resources.

The 1A Processor provides a number of improvements over its Bell System predecessors. It executes instructions faster, provides more program and call storage capacity, and permits more rapid program and translation changes. It also employs an improved order structure that uses less storage and leads to more efficient program development. In addition to space, power, and cost reductions, it provides improved trouble detection and repair, shorter installation intervals, and increased dependability. The maintenance programs and fault-recognition programs are an integral part of the processor. It has stand-alone capability that permits use in a variety of applications.

The intent here is to outline the objectives, design philosophy, major features, and applications of the 1A Processor and to provide an introduction to the more detailed technical papers that follow.

II. OBJECTIVES

2.1 Capacity

Capacity objectives for the 1A Processor were set to meet the processing needs of electronic switching systems, which must handle about 500,000 busy-hour calls in a toll environment.

2.2 Flexible network interface

Another objective in the design of the processor was a flexible network interface that permits control of either the space-division network used in No. 1 ESS that requires central pulse distribution enabling or the time-division network used in No. 4 ESS that uses coded enabling.

2.3 Dependability

Bell System electronic switching systems must provide dependable telephone service 24 hours a day during the life of the system. Converting this requirement to a specific design objective, a service interruption which results in a total loss of service, should average less than 3 minutes per year per office.

Since total loss of service can be caused by facilities other than the processor, the processor has been allocated only two-thirds of the downtime; that is, 1A Processor service interruptions that result in a total loss of service should be less than 2 minutes per year per office.

2.4 Self-sufficiency

The processor was designed with the capability of stand-alone operation. In other words, it consists of a self-sufficient hardware-software package that permits self-testing without any auxiliary equipment.

2.5 Program compatibility

Another important goal was the ability to retrofit the processor into capacity-limited operational No. 1 ESS offices. To achieve compatibility, the No. 1 ESS instruction set was included as a subset of the 1A Processor instruction set. This allows well-tested No. 1 ESS call-processing programs to be semiautomatically computer-translated for operation on the 1A Processor.

2.6 Environmental

The processor was designed to operate over a wide temperature and humidity range. Specifically, the worst-case range is from 0°C to 50°C and from 20 percent to 80 percent relative humidity.

2.7 Economy

The processor was designed to be economically competitive with its predecessors, while still adhering to the preceding objectives. To minimize development and manufacturing costs, a low-cost standardized technology based on integrated circuit building blocks was used, and computer aids were concurrently developed to accelerate all aspects of hardware and software development, manufacture, and test.

2.8 Other objectives

Many other objectives were established for the 1A Processor. For example, the processor should require minimum floor space, energy consumption, and installation time, while maintaining enough of a structural similarity to the No. 1 ESS processor and standardized input/output message format to minimize additional training required for experienced ESS craftspersons.

III. DESIGN PHILOSOPHY

3.1 Standardized technology[4]

The ultrahigh performance requirements of the 1A Processor require maximum use of integrated-circuit and thin-film techniques. The packaging methods used in the first generation of electronic switching systems are not adequate for such complex and advanced circuitry. As a result, a fundamental premise in the design of the 1A Processor was recognition of the need for a new and innovative packaging approach. To meet this need, a standardized packaging system called 1A technology was developed.

1A technology consists of a set of standardized high-performance devices and packaging techniques that provide miniaturization, high-speed performance, low-energy consumption, ultrareliability, low-cost manufacture, and ease of installation and maintenance. These elements form the basic building blocks for the processor. Application guidelines stress design standardization and full connectorization of packs, units, and frames. Connectorization permits rapid assembly of complete processors for comprehensive factory testing, rapid disassembly for shipping, and rapid reassembly of a pretested processor on site. It also provides for rapid repair and maintenance. The guidelines also specify a standardized, fixed floor plan, which is applicable to local, tandem, and toll offices and provides graceful store growth. The net result is a reduced engineering and cabling effort and a shortened telephone company order to service interval.

1A technology is designed for use in a broad spectrum of other digital units or processors. It is used in the space-division networks of No. 1 ESS, No. 2 ESS, and No. 3 ESS; the processor of No. 2B and No. 3 ESS; and the time-division network of No. 4 ESS.

3.2 Improved development methods

In the formative stage of the project, it was evident that methods used in the development of the first generation of the electronic switching system family were inadequate for the development of the new generation. This was due to the complexities of integrated circuit technology as well as the increased size of the software package. Hence, a decision was made to designate the attainment of improved development techniques as a specific goal and to form organizations chartered to achieve this goal. The thrust for new techniques was designated as the development of development methods.[5] The main objectives were improved quality, enhanced diagnosability, assured manufacturability, reduced cost, shortened intervals, and more efficient utilization of the members of the development team.

As a result, Bell Laboratories has developed a complete computer-based data-management facility that permits design specifications to be introduced into a massive data base and enables their subsequent use by complex application programs (see Fig. 1).

The design verification, physical design, documentation, and manufacturing application programs have been essential to the development of the hardware and software of not only the 1A Processor, but also the networks of No. 1 ESS, No. 2 ESS, No. 3 ESS, and No. 4 ESS, and the processor of No. 2B ESS and No. 3 ESS.

The application programs include both automatic and interactive capabilities. The latter permits the designer to interrupt, query, and make changes as an application program is run.

Use of a general-purpose computer with virtual memory simplified the storage of massive quantities of data and made feasible the simulation of large-scale systems containing tens of thousands of gates. For example, simulation of the 50,000-gate central control eliminated many clerical encoding and logic errors at a time when the only existing central

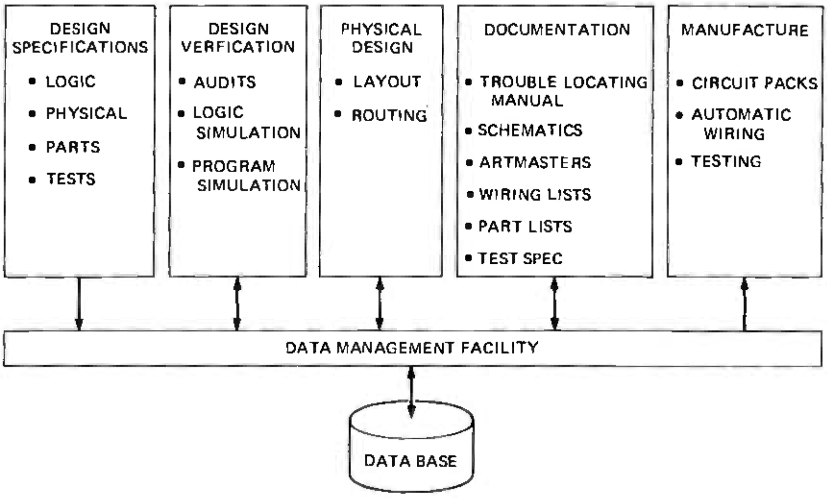


Fig. 1—Computer-aided design.

control was "logically built" in the memory bank of a general-purpose computer.

Interactive simulators were also used to validate new system or program strategies, to design programs, and to debug programs that have been designed. Simulation was found to be an effective debugging tool for all types of programs including fault recovery and diagnostic programs.

The need for a sophisticated laboratory software testing tool triggered the design of a comprehensive utility test system. The hardware consists of over 80,000 gates and 16,000 words of memory. It provides an electrical and time buffer between the processor and a utility minicomputer. The system, the minicomputer in conjunction with its resident utility program, provides direct and conditional control over any processor-resident program as well as monitor, trace, load, and dump capabilities.

3.3 Integrated development, manufacture, and test[6]

The need to automatically convert a staggering volume of design information was recognized as an integrated part of the development methods. To meet this need, application programs were designed to extract manufacturing information from the data base in the general-purpose computer. This information is used by Western Electric to generate artmasters for the plated-interconnection patterns on both circuit packs and printed-wire backplanes, and to produce machine-readable data to control wiring machines for automatic wiring of those connections, which are not contained on the printed-wire backplanes.

Bell Laboratories and Western Electric engineers collaborated on the development of computer-controlled test facilities. The test sequences are verified using the design data base and are used to program minicomputers that control a variety of test stations that test everything including interconnection patterns, circuit packs, backplanes, frames, and complete processors. The processor test sequences from the data base are also used for installation and performance testing in the field.

Still another set of application programs use the design data base to generate documentation such as schematic diagrams and wiring lists.

IV. PROCESSOR ORGANIZATION

4.1 Hardware[7,8]

As shown in Fig. 2, the general architecture of the 1A Processor is similar to the No. 1 ESS processor[9], except for the provision of a direct-memory-access feature for auxiliary units. This is accomplished over an auxiliary unit bus on the central control. Thus, data can be transferred between program and call stores, and file stores, tape units, and external data links with minimal interference with other processor activities. This feature is essential to the achievement of the modern services planned

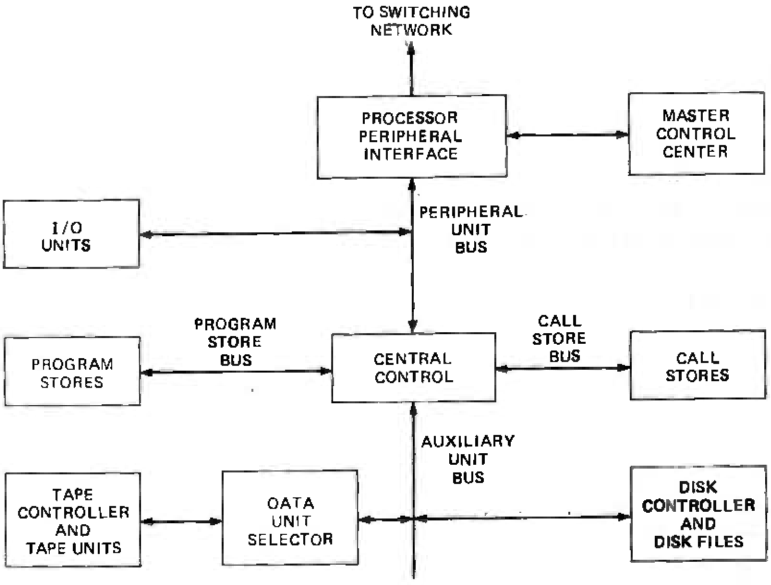


Fig. 2—1A Processor block diagram.

for electronic switching systems without impacting heavily on the call-processing capacity of the system.

The direct-memory-access feature also permits a varied spectrum of maintenance programs to be retained on lower-cost file stores with programs paged into high-speed random-access stores only as needed. Substantial reduction in high-speed memory can thus be realized. Furthermore, many maintenance features, which were not included in earlier generation electronic switching systems because of a high-cost penalty, are now economically provided in the 1A Processor.

High-speed, low-power DTTL logic integrated circuit devices are used for all logic circuits. The beam leads of the chips are thermocompression bonded to a ceramic substrate. The complete ceramic is mounted on a supporting aluminum plate and bonded to a male-connector assembly.

Program and call stores contain two magnetic core modules and are capable of storing 65,536 words of 26 bits (24 data bits plus 2 parity bits). File stores contain four magnetic disks and are capable of storing a total of 64 megabits. A sufficient number of each type of store is provided to meet the needs of each of the contemplated processor applications.

In the 1A Processor, all subsystems have redundant units and are connected to the basic system via redundant bus systems. Thus, it is possible to operate at full capacity with faults in some units.

The central control units are fully duplicated; they normally operate in step and compare results.

The program store community consists of a prime set plus two additional (or "roving spare") stores. If one of the prime stores fails, the program that is contained in that store is transferred from the duplicated copy of the system program contained in the file store to one of the "rover spares."

The call-store community consists of a prime set plus a full on-line duplicate set for only those stores that contain transient call data. For those stores that contain translation data, one of the duplicated stores containing transient call data is preempted and loaded with the necessary translation data from the duplicated copy in the file store.

Remaining critical units in the processor are duplicated while less critical units are not duplicated.

4.2 Maintenance software[10]

Error-detection is the basis for automatic detection and resolution of problems without service interruption. This is achieved through matching of duplicated units, parity checks on communications (both address and data) over all buses, interval self-checking in hardware units (control pulsing, timing, internal data transfer), and system sanity timers.

A failure reported by any one of these error-detection schemes triggers a program interrupt, and control is transferred to a fault-recovery program. This program verifies that a fault is present, identifies the most probable subsystem or unit, and then removes the faulty unit from service by reconfiguring the system to maintain call-processing capability. The intent is also to retain as much redundancy as possible in the remaining hardware units in the working configuration.

Special programs, called restart programs, set a diagnostic-request flag and save any data that will aid in identifying trouble symptoms. Upon completion of these actions, system operation is continued by returning control to the interrupted program.

Maintenance experience with earlier generation electronic switching system processors clearly highlighted a need for rapid repair through the use of a comprehensive set of diagnostic tests. To meet this need, a number of diagnostic practices, which differed significantly from those employed in the sixties, were introduced. These practices included organization of the development team to emphasize early and continuing diagnostic planning and execution, use of a common standard macro-oriented test design language, and application of a complementary fault-simulation process involving the merging of logical simulation in a general-purpose computer with physical simulation in an operational system laboratory.

The resulting diagnostic programs differ from those used in previous electronic switching systems in that they are also used, as noted earlier, for computer-controlled hardware testing in the manufacturing plant and during installation on the telephone company site.

Hardware or software errors can cause mutilation of the program or data base. Special programs for mutilation detection provide for a progressive verification of integrity. If the mutilation is minor, audit programs analyze the affected data base and correct any errors on a time-shared basis with normal call-processing programs. If the mutilation is severe, the call-processing program is stopped until the mutilation programs are regenerated by transferring the backup information from file memory.

4.3 Administrative software

To provide a number of the required 1A Processor capabilities, an extensive set of administrative programs was developed and is included in the processor software package. (These perform many of the functions normally included in an operating system, but since program control is not included, it would not be correct to use that term here.)

These programs serve many functions, such as: transferring data between the call and program stores and the auxiliary units such as the file store, tape store, and input/output terminals; transferring programs into the program store from the file store; controlling (on a predetermined priority basis) the execution of deferred programs in a time-shared mode with call-processing programs, gathering all system-detected trouble symptoms and recording them in the file store for later analysis (these data include audit errors, interrupt reports, fault recovery actions, and diagnostic failure results); executing utility features permitting on-site analysis; temporarily allocating and loading stores for special programs; and providing an on-line complete and partial update capability.

V. APPLICATIONS

5.1 Large toll offices—No. 4 ESS[12]

On January 17, 1976, the first 1A Processor for toll applications was cut into service as the common control for the No. 4 ESS toll switching office in Chicago (see Fig. 3). No. 4 ESS uses a solid-state time-division network and can switch up to 550,000 peak busy-hour calls from 107,000 trunks. It is described in a future special issue on No. 4 ESS. The second No. 4 ESS was cut into service in Kansas City on July 3, 1976.

Two more No. 4 ESS offices are expected to become operational in 1976.

5.2 Metropolitan local offices—No. 1A ESS[13]

On October 15, 1976, the first 1A Processor for local applications cut into service as the common control for the No. 1A ESS local switching office in Chicago (see Fig. 4).

No. 1A ESS uses a remreed space-division network and can handle up to 240,000 peak busy-hour calls and about 100,000 line terminations. No. 1 ESS call-processing programs are semiautomatically computer-translated to operate on the 1A Processor. The 1A Processor occupies less than one-half the space of the No. 1 ESS processor.

5.3 Retrofit into No. 1 ESS offices

Plans call for the 1A Processor to replace the No. 1 ESS processor in capacity-limited working offices (the first retrofit is scheduled for 1978). The new processor will provide more than twice the present call-processing capacity of a No. 1 ESS.

VI. PERFORMANCE

6.1 Laboratory

Prior to field service, extensive laboratory testing was conducted during which over 100,000 processor hours were logged. This testing

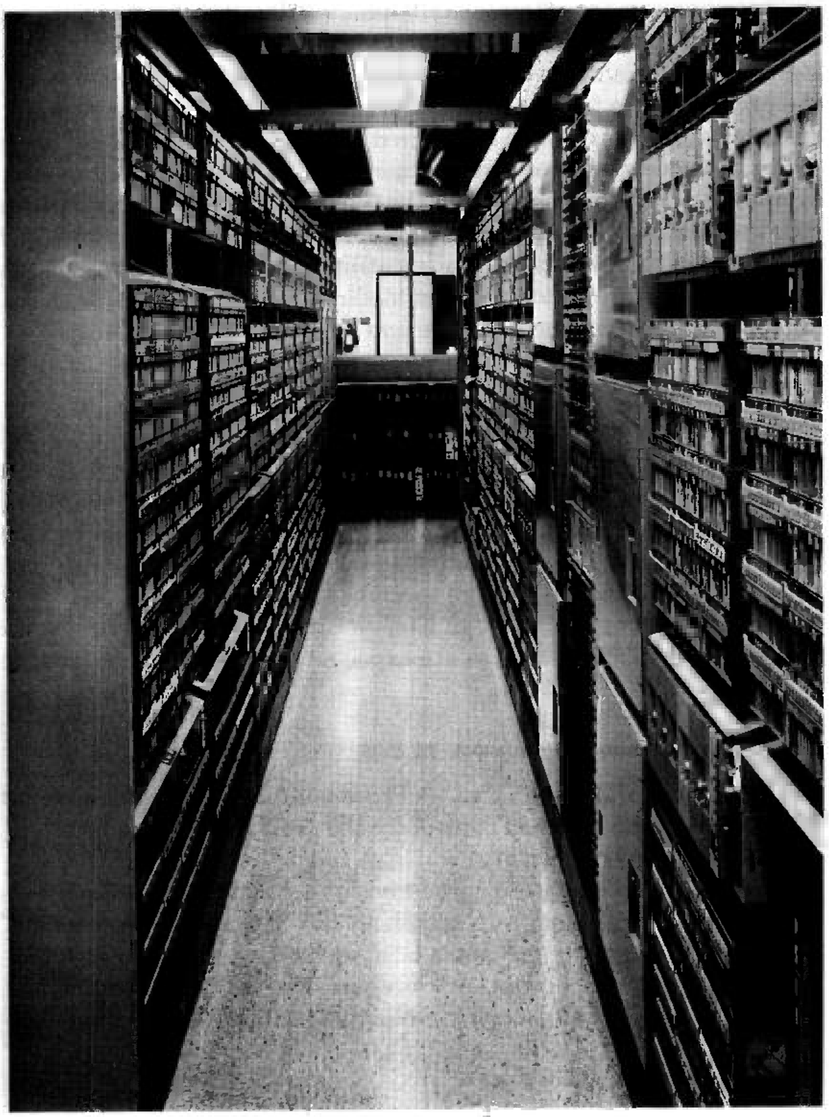


Fig. 3—1A processor in No. 4 ESS (Chicago toll office).

indicates that the design objective of hardware component reliability, automatic isolation of faulty units without service interruption, rapid repair, and automatic recovery from program and data mutilation are being met. More detail is presented in the associated papers.

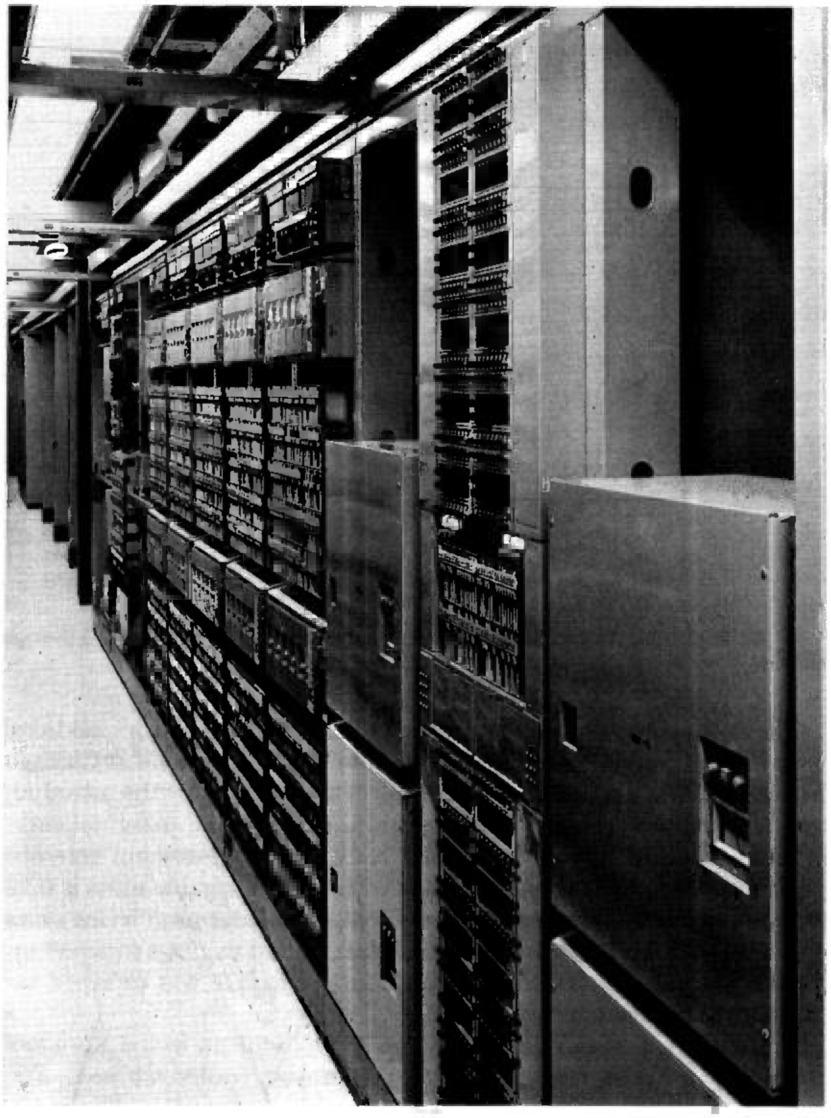
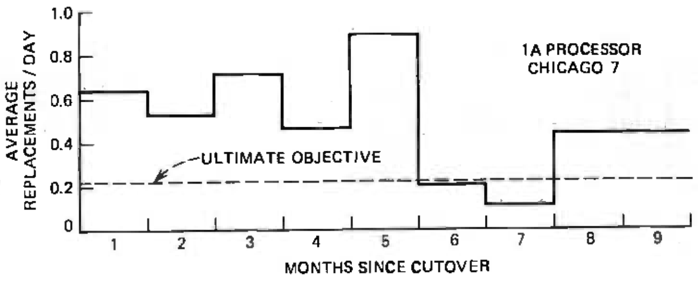


Fig. 4—1A Processor in No. 1A ESS (Chicago local office).

6.2 Field

Field performance of the 1A Processor in both No. 4 ESS and No. 1A ESS has been excellent. This is best illustrated by key operational data gathered from the first two offices put into service.



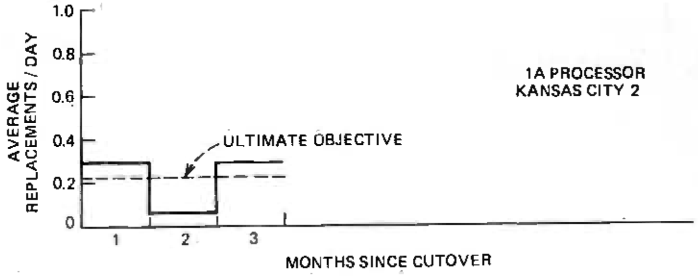


Fig. 5—Replacement rate for pluggable units in the 1A processors in No. 4 ESS through October 1, 1976.

The replacement rate for pluggable units (circuit packs, core modules, power modules) in both offices is shown in Fig. 5. The trend at Chicago 7 shows a marked reduction in the replacement rate after the introduction of some design improvements and the removal of marginal early production units. The ultimate or steady-state replacement rate objective for the 1A Processor complex with 7000 pluggable units is 0.22 unit per day. The data from Kansas City 2 (with later production units and where design improvements were introduced prior to cutover) indicate that the ultimate replacement rate objective was attained essentially at cutover.

Hardware, software, or memory mutilation problems in the No. 4 ESS can result in the need for system reinitialization (called a phase). Fig.

TYPE OF PHASE	DURATION (SECONDS)	EFFECT OF PHASE
1	1	NO SERVICE EFFECT
2	27	TRANSIENT CALLS LOST
3	40	TRANSIENT CALLS LOST
4	40	ALL CALLS LOST

Fig. 6—No. 4 ESS reinitialization phase structure.

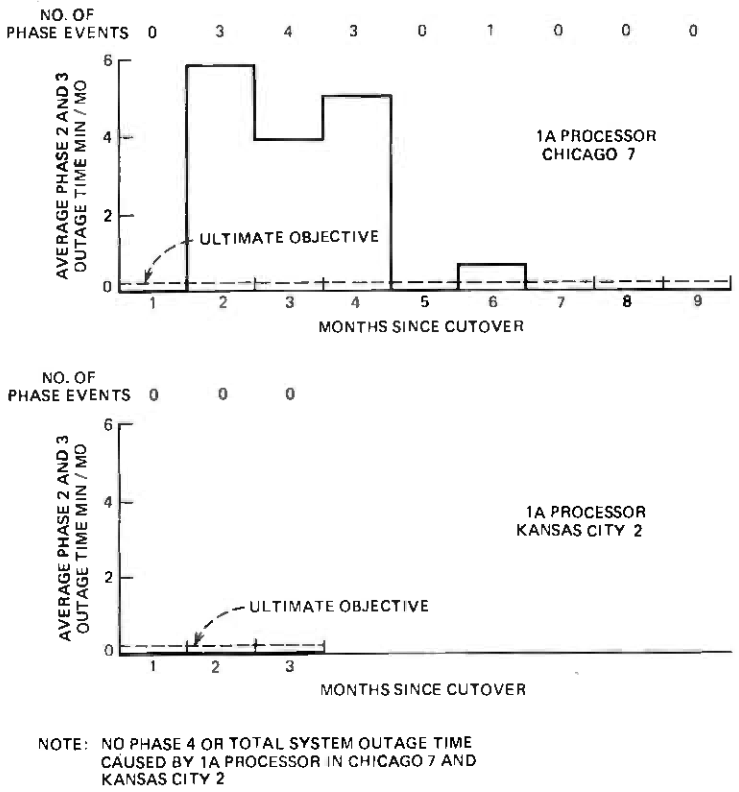


Fig. 7—Phase events and outage time in No. 4 ESS caused by 1A Processor through October 1, 1976.

6 shows the No. 4 ESS reinitialization phase structure. Note that a phase 1 has no system effect. Phases 2 and 3 result in the retention of established calls, but the loss of transient calls. Phase 4 results in a total system outage with a loss of all calls.

Fig. 7 shows both the number of phase 2 and 3 events as well as the average phase outage time by month caused by the 1A Processor for both No. 4 ESS offices. The outage time represents the duration of the phase, during which established calls were retained, but no new calls were established. The trend at Chicago 7 shows a rapid reduction in the frequency and outage time for phase 2 and 3 events as the causes of the phases were determined and corrected. The ultimate or steady-state total 1A Processor phase outage time objective is 0.16 minute per month or 2 minutes per year. Note that at Kansas City 2 there have been no 1A Processor phase events or outages.

As noted in Fig. 7, no phase 4 or total system outage has been caused

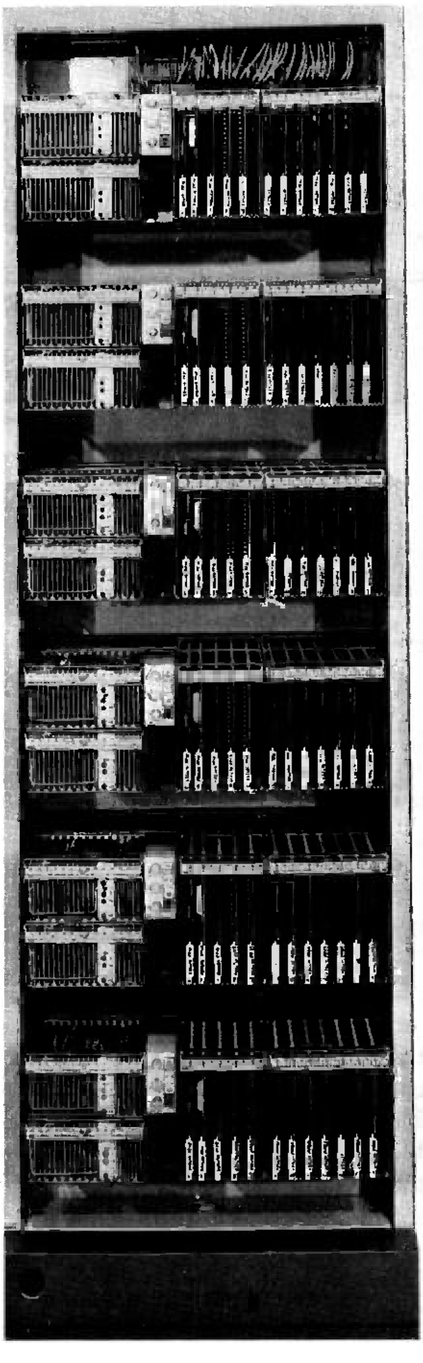


Fig. 8—Program and call stores using 4 K MOS integrated-circuit devices.

by the 1A Processor in either Chicago 7 or Kansas City 2.

VII. CONCLUSION

After logging more than a total of 8000 in-service hours, 150,000 laboratory and field hours, 4 million unit hours, 350 million ceramic integrated-circuit pack hours, and 10 billion semiconductor integrated-circuit hours on the 1A Processor, overall performance indicates that the design objectives are being realized.

Nevertheless, constant attention is being given to possible areas of improvement. 1A technology was designed so that larger scales of integration could be smoothly introduced into the 1A Processor. Furthermore, as with the No. 1 ESS processor, as technological advances occur, it is expected that subsystems can be redesigned and smoothly introduced into the 1A Processor. A specific example is a program and call store using 4 K MOS integrated-circuit chips in place of magnetic cores. These stores require less space and power than the magnetic core stores currently used in the 1A Processor. As shown in Fig. 8, six stores (over 10 megabits) can be mounted in one 2-foot, 2-inch bay. The integrated circuit program and call store is currently being manufactured by Western Electric with shipment expected in early 1977. These continuing refinements of the 1A Processor should provide additional improvements in performance as well as additional economic advantages.

VIII. ACKNOWLEDGMENT

The design of the 1A Processor required the cooperative efforts of hundreds of people in Bell Laboratories, Western Electric, and A.T. & T. The author wishes to acknowledge the contributions of all the team members whose work is summarized herein.

REFERENCES

1. R. E. Staehler, "1A Processor—A High Speed Processor for Switching Applications," International Switching Symposium, Boston, Massachusetts, June 1972.
2. R. E. Staehler and T. S. Greenwood, "1A Processor—Development and Status," International Switching Symposium, Munich, Germany, September 1974.
3. R. E. Staehler and R. J. Watters, "1A Processor—An Ultra-Dependable Common Control," International Switching Symposium, Kyoto, Japan, October 1976.
4. J. O. Becker, J. G. Chevalier, R. K. Eisenhart, J. H. Forster, A. W. Fulton, and W. L. Harrod, "Technology and Physical Design," B.S.T.J., this issue, pp. 207–236.
5. R. W. Ketchledge, "Development of Development Methods," International Switching Symposium, Munich, Germany, September 1974.
6. H. A. Hilsinger, K. D. Mozingo, C. F. Starnes, and G. A. Van Dine, "Testing and Integration," B.S.T.J., this issue, pp. 289–312.

7. A. H. Budlong, B. G. De Lugish, S. M. Neville, J. S. Nowak, J. L. Quinn, and F. W. Wendland, "Control System," B.S.T.J., this issue, pp. 135–179.
8. C. F. Ault, J. H. Brewster, T. S. Greenwood, R. E. Haglund, W. A. Read, and M. W. Rolund, "Memory Systems," B.S.T.J., this issue, pp. 181–205.
9. "No. 1 Electronic Switching System," B.S.T.J., *43,* Parts 1 and 2, September 1964.
10. P. W. Bowman, M. R. Dubman, F. M. Goetz, R. F. Kranzmann, E. H. Stredde, and R. J. Watters, "Maintenance Software," B.S.T.J., this issue, pp. 255–287.
11. G. F. Clement, P. S. Fuss, R. J. Griffith, R. C. Lee, and R. D. Royer, "Control, Administrative, and Utility Software," B.S.T.J., this issue, pp. 237–254.
12. A. E. Spencer, Jr. and H. E. Vaughan, "No. 4 ESS—A Full-Fledged Toll Switching Node," International Switching Symposium, Kyoto, Japan, October 1976.
13. J. S. Nowak, "No. 1A ESS—A New High Capacity Switching System," International Switching Symposium, Kyoto, Japan, October 1976.

Copyright © 1977 American Telephone and Telegraph Company
THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 56, No. 2, February 1977
Printed in U.S.A.

1A Processor:

Control System

By A. H. BUDLONG, B. G. DE LUGISH, S. M. NEVILLE, J. S. NOWAK, J. L. QUINN, and F. W. WENDLAND

(Manuscript received July 16, 1976)

This article contains a description of the central control, input/output subsystem, processor-peripheral interface, master control console, and the interunit communication bus system. The units are discussed in a manner which highlights comparison of features with No. 1 ESS and stresses those features which are most important in meeting the stringent reliability objectives.

I. INTRODUCTION

The 1A Processor control system, coupled with program, call, and auxiliary memory, is an independent, stand-alone processor with full operational and maintenance capability. It provides for execution of the processor and system program, automatic system maintenance actions, manual control and monitoring, and interfaces to memory, network, and external systems.

This processor is an outgrowth of previous Bell System switching processors. Although it contains many similar or modified features, the new processor offers significant improvements over its predecessors. In comparison with the No. 1 ESS processor,[1] several major new features and improvements are provided—for example, four to eight times faster instruction execution, writable program memory, greatly expanded program and call memory size capability, and auxiliary memory with autonomous transfer to and from program and call memory (direct memory access). These advancements are achieved while maintaining compatibility with the No. 1 ESS network and programs.

The control system consists of the central controls (CCs), communication buses, and the man-machine interface and control. The man-machine interface and control includes the master control console (MCC),

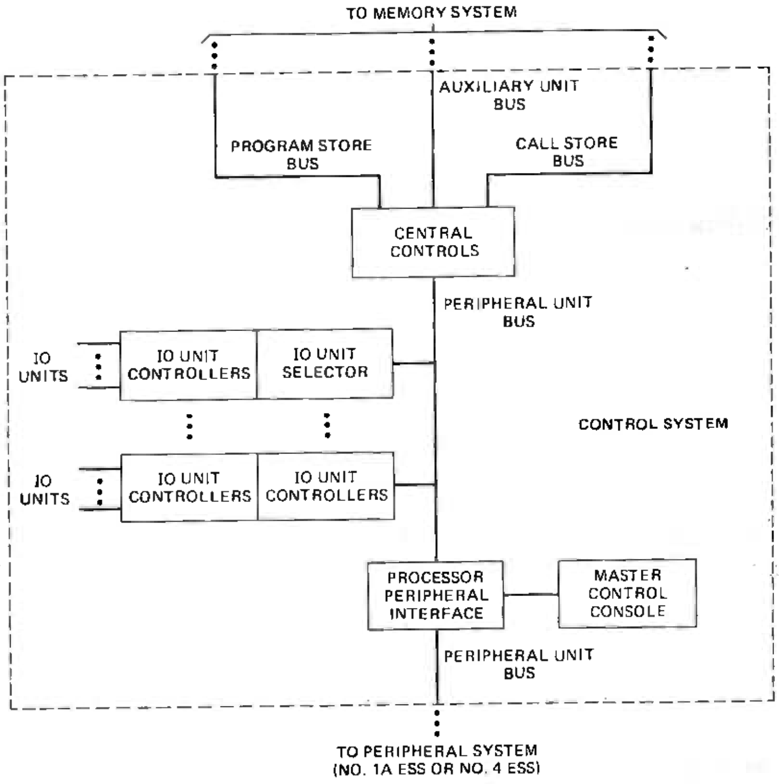


Fig. 1—Control system.

processor-peripheral interface (PPI), and the input/output unit selectors and controllers (IO). These are illustrated in Fig. 1.

This article presents an overview of the control system features and then details the operation and maintenance features of each subsystem.

II. CONTROL SYSTEM FEATURES

2.1 General

The control system coupled with the memory system constitutes a stored program processor. For switching system control, the instruction capability of the central processing unit (CC) is tailored for control and logical operations rather than mathematical operations such as multiplication, division, and function evaluation. Internal CC operations include fixed-point addition and subtraction, word and item manipulation (such as rotation and masking), sequencing to perform multiple autonomous tasks associated with each communications bus, sophisticated instruction address manipulation, and sequencing and logic operations dedicated to maintenance functions and interrupt handling. The instruction set reflects these features and is oriented to provide efficient transfer of control, item manipulation, index register modification, and condition testing. Special instructions and instruction sequences are provided to optimize control of the switching network, and other special instructions are oriented toward maintenance actions such as interrupt handling, configuration control, and subsystem fault diagnosis.

In addition to the improvements previously mentioned, the 1A Processor surpasses the No. 1 ESS processor in other areas. Operational improvements include an expanded instruction set, more general internal CC logic, a coded enable unit selection capability on the peripheral bus, expanded IO capability for input and monitoring, and a centralized peripheral interface. Maintenance features have been improved by including more access and control, so that virtually all memory elements can be initialized and monitored. In addition, checking circuits can be exercised, all bit combinations can be transmitted over buses, and critical control signals can be sent to each unit over two completely separate paths.

2.2 Central control

The internal structure of the 1A Processor CC is similar to the No. 1 ESS CC, in part to achieve a high degree of forward compatibility. Within this constraint, a great many improvements have been made to its instruction set.

Internal registers and arithmetic logic functions have been expanded and enhanced. Processing features include generalized adder access, generalized registers, "shadow" registers which save the contents of each general register on request, and 24-bit internal data paths and registers. These changes are provided to improve the development of application and maintenance programs.

Instruction set features include generalized No. 1 ESS options, instructions with multiple functions, new instructions to utilize new internal processing features, memory protection features, and an instruction to search memory for a word matching selected register contents. Instructions are one or two 24-bit words, where appropriate, to save memory when only one word is necessary. In many cases, an instruction is available in either one- or two-word format, and the format selection is based on the length of fields or options specified by the programmer. Transfer addressing capability is improved by the use of an address stack utilizing a pointer in CC. Program and call store interfaces and internal CC logic are designed to work with either a 700-ns or 1400-ns CC-store cycle. Maintenance instructions allow communication paths and subsystem interfaces to be fully checked.

The peripheral control has been expanded beyond the central pulse distribution (CPD) scheme used in No. 1 ESS to provide coded enabling with bus transmission checks. The coded-enable scheme accommodates a long bus without an excessive time penalty. This is possible because the CC recognizes the completion of an operation and resumes processing immediately, rather than allowing a fixed time for the full bus length. The reply bus has also been expanded from 16 to 24 bits, and nonperipheral instructions that do not interfere with peripheral registers are processed simultaneously with the completion of peripheral instructions.

The new auxiliary unit control provides for mass memory units with autonomous transfer of data between program or call memory and the auxiliary memory. This saves substantial processing time, since the transfers are not directly controlled by instructions executed in the CC. An interface controller in the CC administers the data transfer to or from auxiliary memory units on a priority basis. The priority can be altered by program control.

Enhanced maintenance features in the CC include selective inhibits of clock pulses, read and write access for nearly all memory elements including sequencer circuits, sophisticated start-stop control, and two 24-bit matchers in each CC. The CCs have access to each other over fast, dc interconnecting buses which provide control and route the CC data for matching. They also have the capability to generate maintenance and configuration signals for each unit and a circuit to check for proper operation.

2.3 Communication buses

Communication buses are used to transfer control, timing, and data signals between processor subsystems. The buses are designed to provide physical and electrical isolation, to provide for simple addition of units on an in-service basis, and to provide a pluggable connection for efficient factory testing, installation, and maintenance.

Peripheral buses are directly compatible with the No. 1 ESS peripheral system and, therefore, the timing and signal levels are constrained to be the same as No. 1 ESS. Other buses utilize advanced circuits, packaging, and cable techniques to achieve faster, lower-power operation consistent with the 700-ns CC cycle.

Each bus system is duplicated and has configuration control circuits, as in any major subsystem.

2.4 Man-machine interface and control

Normally, communication between personnel and machine is via terminals, such as teletypewriters, that interface through input/output unit selectors and controllers. Up to 96 channels are available with 110-

or 1200-word-per-minute data rates. Terminals may be local or remote, with data sets used for remote terminals.

The master control console (MCC) provides direct manual control and monitoring of key indicators. In addition to this console for the 1A Processor, a companion console is associated with each system using the 1A Processor, such as No. 1A ESS[2] or No. 4 ESS.[3]

The processor peripheral interface (PPI) provides the logic and access circuitry for the MCC. It also contains circuitry to monitor and control individual unit power switches and other manual controls.

A major function of the PPI is to provide a common unit for interface to peripheral systems such as the No. 1A ESS or No. 4 ESS network. The ability to loop peripheral bus transmissions back to the CC is provided. This allows intraprocessor bus cables and circuits to be checked independently from the peripheral units.

III. CENTRAL CONTROL OPERATIONAL HARDWARE FEATURES

Central control is a synchronous stored-program central processing unit designed with a multiplicity of hard-wired sequencers to facilitate the level of concurrent operation required. It is worst-case designed to operate on a tightly synchronized basis with main and auxiliary memory units within the 1A Processor complex. The worst-case design is based on a model of 1A technology interconnection and devices and is supported by both extensive measurements and analytical methods. Within central control, over 100 representative critical timing chains were simulated using simulation models and tools that evolved with the hardware design. With specifications on the devices and hardware, which also evolved with the design, the average worst-case maximum delay per logic stage for typical critical paths in the CC is about 12 ns, allowing about 58 stages of delay in a 700-ns cycle.

In addition to serving as an instruction execution and control unit, central control provides system synchronization. System timing originates from a 20-MHz crystal oscillator circuit in the active central control. This oscillator circuit provides signals for both the active and standby CCs, thus synchronizing them. The active CC also generates synchronization pulses for other units in the system, including main and auxiliary memory units. These synchronization pulses are sent every CC cycle (700 ns) on a per-unit basis.

Central control is designed to operate with any mixture of 700-ns and 1400-ns cycle time main memory units. In the normal system mode, the central controls are synchronized and running in step. In other words, they are executing the same instructions and operating on the same data. This is referred to as "duplex operation." In the 1A Processor system, transient data are generally duplicated, that is, stored in two physically separate main memory units assigned the same locations in the address spectrum. Valid system configurations include duplex central controls fetching data from duplicate stores at different speeds. Each central control monitors the speed of the main store unit accessed by the mate central control, and if either CC accesses a slow (1400-ns) module, then both CCs run at the slower rate.

The instruction execution rate is strongly influenced by the speed of the main memory units since a significant proportion of the instructions are executed in one main memory cycle time. For representative code with 1400-ns memories presently available for field use, the average execution rate is about 750,000 instructions per second. Although instructions are designed to execute in an integral number of 700-ns cycles, the "average" instruction requires only 1100 ns to execute; the execution control is idle about 18 percent of the time, particularly as a result of waiting for the target of a successful transfer to be fetched.

3.1 Instruction fetching

Modern processors achieve high capacity not only because of fast hardware but also because of the increase in concurrent operation including the overlapping of instruction fetching and executing. The fetch control may be several instructions ahead of the execution control. When the fetch control system detects a conditional transfer instruction that is not yet ready to be executed, it is assumed that the transfer will not be made and straightline fetching continues. However, when an unconditional transfer is detected and a transfer is not currently being executed, no fetch operation is initiated until the target address is calculated by the execution control system (this may save a CC cycle).

To facilitate concurrent memory operations, the main memory is split into two systems: program store and call store. Typically, the program store system provides the instruction stream and the call store system provides the data. This distinction is not rigid, however, since either system can perform either function and both systems communicate with central control in similar fashion (timing, synchronization pulses, etc.). The major difference between the program and call store systems is the width of the communication path. On each read operation, the program store returns an even-odd address pair of words and central control selects the required portion, whereas the call store returns only a single word.

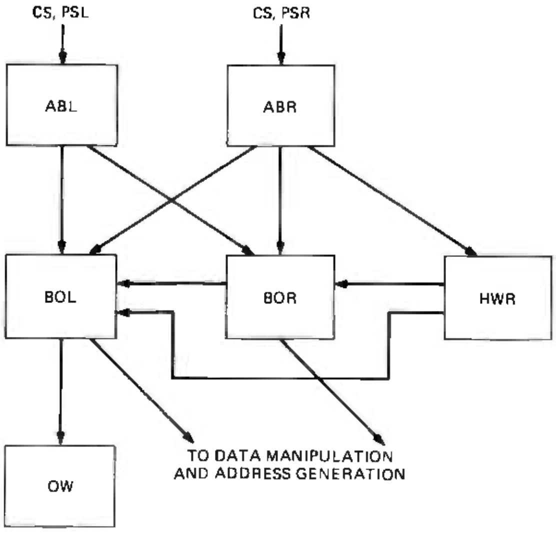
Since fetching program code from call store is an inherently slower operation than fetching from program store, this use is restricted and protection is provided so that unplanned transfers to call store are not allowed. Execution of program from call store is used for fault recovery and diagnostic for the program store system. These programs are infrequently used and therefore require a very small fraction of system real time. Gating and sequencer action for call store fetching closely emulate gating and sequencer action for program store fetching. To minimize memory access conflicts between the execution and fetch control systems, only seldom-used data are located in program store.

A program instruction may be either one or two words in length, that is, either 24 or 48 bits. The number of bits in an instruction is not directly related to the execution time, although double-word instructions tend to take more cycles to execute because they allow greater flexibility in special features and options. The two words read on each program-store fetch may be two single-word instructions, a single-word instruction and either half of a double-word instruction, halves of two double-word instructions, or one entire double-word instruction. The single-word orders save substantial program storage space since two-thirds of the instructions are typically single-word orders.

To improve efficiency in fetching long (double-word) instructions, these instructions may be aligned on even-address boundaries by inserting a special no-operation order which is not executed if encountered on an odd boundary following a short (single-word) instruction.

The instruction stack size of six (24-bit) words is chosen to best accommodate the frequency of transfers which is relatively high (about 15 percent). Because the stack is small, no attempt is made to determine if the target instruction of a transfer might already have been fetched; instead the target is fetched again.

A pictorial representation of the instruction stack is shown in Fig. 2.



GLOSSARY

ABL	AUXILIARY BUFFER ORDER WORD REGISTER LEFT
ABR	AUXILIARY BUFFER ORDER WORD REGISTER RIGHT
BOL	BUFFER ORDER WORD REGISTER LEFT
BOR	BUFFER ORDER WORD REGISTER RIGHT
CS	CALL STORE
HWR	HALF WORD REGISTER
OW	ORDER WORD REGISTER
PSL	PROGRAM STORE LEFT
PSR	PROGRAM STORE RIGHT

Fig. 2—Instruction stack.

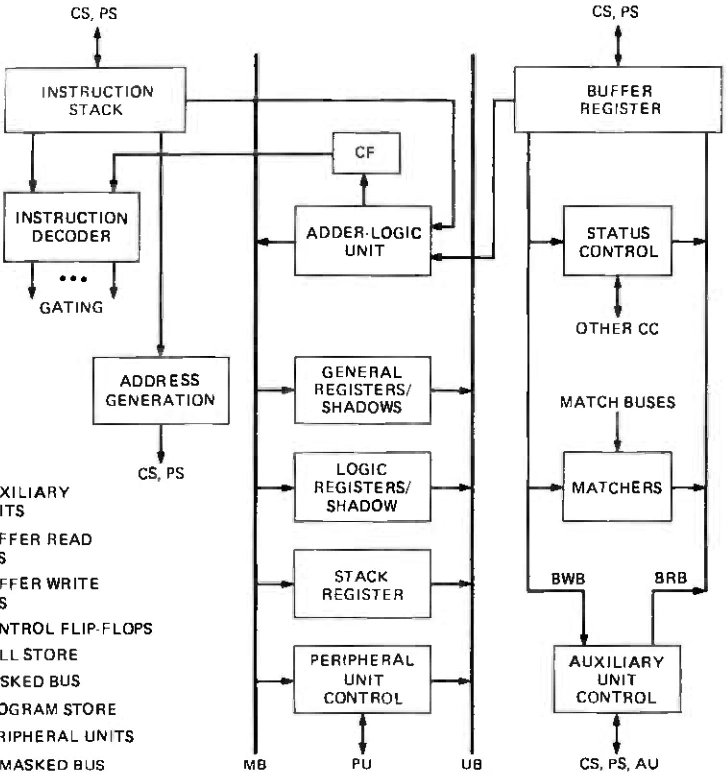
A double-word program store reply is accepted into the ABL and ABR. If both words are required and if the BOL and BOR are available (empty), then the contents of the ABL and ABR are immediately gated down into position to be executed. If the BOL contains an instruction which has not yet started execution, and the BOR and HWR are available, then the contents of the ABL and ABR are gated to the BOR and HWR, respectively. If only the right half of the reply is required (for example, on a transfer to an odd address), then the ABR contents are gated to the BOL. Other gating paths within the stack are shown in the figure.

Instruction decoding and execution begin from the BOL with the data portion of a long instruction in the BOR. In the first cycle of execution, the contents of the BOL are duplicated in the OW and the data portion of the instruction is gated to the appropriate data-manipulation or address-calculation circuitry. The remainder of instruction decoding is performed from the OW with an overlap of decoding which assures continuity of gating and control signals.

3.2 Instruction execution

The 1A Processor order set is a substantially enhanced version of the No. 1 ESS order set. One of the basic requirements of the order set is to allow for the conversion of No. 1 ESS operational code into 1A Processor code via off-line translator programs with a minimum of redesign. Although the resulting order set contains some redundant features such as two address-return mechanisms, it is efficient in both real time and memory usage. The short instructions, for example, reduce required storage for programs by about 33 percent. The mask and shift facilities allow fetching a data word from memory and adjusting an item (a contiguous set of bits) within that data word in either direction with a single-word order in one memory cycle time. A double-word instruction allows detection of the least significant (right-most) one in any general-purpose register (or the logic register) and optionally clears that bit. If the register tested contains all zeros, the instruction transfers control. Conditional instructions employ the two control flip-flops storing the sign and homogeneity (all "1s" or "0s") of the result of an arithmetic or logical operation performed by either a memory read or a test instruction.

The adder and logic unit (ALU) shown in Fig. 3 is oriented to a control and decision-making type of function. The ALU performs common logical operations and fixed-point addition and subtraction using one's complement arithmetic. Arithmetic, logic, and shift functions are provided on all seven general-purpose registers. Each of these registers and the logic register, which is used both as a masking register and as a peripheral operation data reply register, has a shadow register to save data for client



GLOSSARY	
AU	AUXILIARY UNITS
BRB	BUFFER READ BUS
BWB	BUFFER WRITE BUS
CF	CONTROL FLIP-FLOPS
CS	CALL STORE
MB	MASKED BUS
PS	PROGRAM STORE
PU	PERIPHERAL UNITS
UB	UNMASKED BUS

Fig. 3—Central control, block diagram.

programs. Single-cycle short instructions save or restore the contents of any combination of these registers.

To optimize the use of the available code space, particularly on single-word instructions, operation codes are of variable length and may be noncontiguous. The double-word instructions provide access to the full address spectrum, which should be large enough (over four million words) to allow using the 1A Processor in any high-capacity switching application.

3.2.1 Details of execution and timing

The 700-ns central control cycle is divided into fourteen 50-ns basic clock phases. Typical clear and gate pulses are 50 ns and 100 ns in duration, respectively.

As can be seen in Fig. 3, the logical flow of instructions is centered around the use of two major internal bus systems: unmasked bus (UB) and masked bus (MB). The CC cycle is divided into three internal bus phases (0T4, 4T8, 8T0), as indicated in Fig. 4. These bus phases are intentionally unsymmetrical to allow certain data processing functions (e.g., addition) to be performed in one bus phase without penalizing the system cycle time. Even so, the longest bus phase (8T0) is extended to 6T0 for preparation of data (with parity) to be written into memory. It is desirable to execute a memory operation (read or write) within one memory cycle time. Write operations present data to the memory near the end of the first cycle of the instruction and, in the case of the 1400-ns memory, wait for a completion indication during the second cycle. In the case of a 700-ns memory, the operation is completed in one central control cycle.

The time required to perform the three basic steps of a read operation exceeds the memory cycle time by about 300 ns for either memory speed. The first step of a read operation, address generation, requires approximately 350 ns and begins by decoding the instruction as it is accepted into the instruction stack (before T12); the address is ready for transmission before T5 of the first cycle. The second step of a read operation consists of the access time of the memory system, including clock uncertainty, synchronization, cable drivers and receivers with associated logic, and bus transmission delays. For a 1400-ns memory system, this access time totals about 1050 ns; for a 700-ns memory system, about 350 ns is allocated. In either case, the data reply is available in the central control data buffer register by T12 of the appropriate cycle. The third step, processing the data reply, requires as much as 300 ns and may include performing an addition or logic operation (perhaps masking and/or complementing) and placing the result in the destination register. Thus, a read operation is not completed until almost T4 of the cycle following

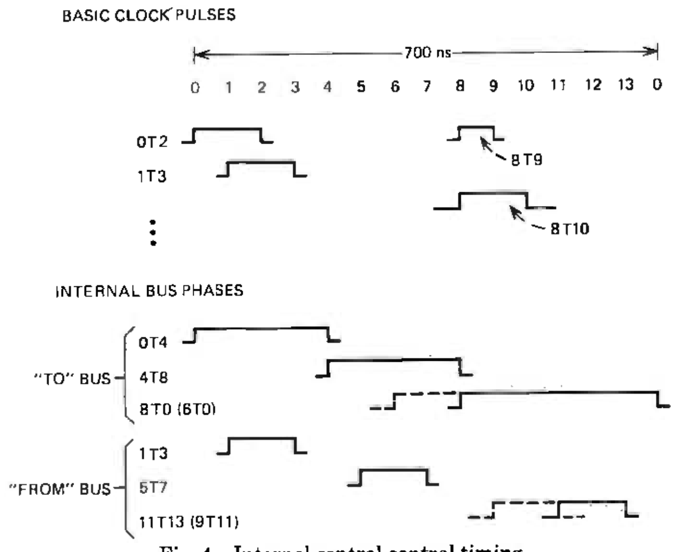


Fig. 4—Internal central control timing.

the store data reply. Decoding of a new order begins as soon as it is available in the instruction stack so that the final 300 ns of the memory read operation overlaps the first 300 ns of the following operation. This overlapping results in restrictions on the use of portions of the circuitry which may be used during this 300-ns interval by each instruction. The read operation may use the data buffer register, the ALU, the masked bus, and the destination register. The operation following the read is restricted during this interval to the unmasked bus and the address generation circuitry. If the beginning instruction requires that the data being gated into the destination register of the read instruction be used for indexing or testing, the normal sequence of accessing a register via the unmasked bus is modified. In the case of indexing, the data are loaded directly from the output of the ALU into the address-generation circuit. In the case of register testing, it is sufficient to inhibit gating the previous contents of the specified register to the control flip-flops since the read operation always loads the control flip-flops from the output of the ALU.

A beginning instruction has access to adequate resources to allow at least the beginning steps of address generation to take place during the initial 0T4 interval. This initial interval is also the time during which the preparatory steps take place for index-register modification which is completed during the 4T8 bus phase. These options are discussed later in this section. Similarly, the argument register within the ALU is commonly loaded during the 4T8 phase in preparation for the 8T0 (or 6T0) data processing phase. A mask may be prepared by gating encoded instruction data from the BOL or BOR to the size-displacement translator in the ALU during 0T4 and by gating the translated (24-bit) mask to the logic register during 4T8 for use during the data processing phase. Table I summarizes the basic operations performed in each internal bus phase.

Table I — Allocation of internal bus phases

Internal Bus Phase	Activity
0T4	Address generation Condition test Logic functions (overlap) Generate mask Preload add-one logic (A-option)
4T8	Index register modification (A-, S-, and W-options) Preload argument Return address save Move mask to logic register
8T0 (6T0)	Write data to buffer register Data word through ALU Special register, bit, or item test

The effective address is usually the sum of a data field in the instruction and either the contents of a general-purpose register (indexing) or the address of the instruction currently being executed (relative addressing). The address-generation circuitry includes a fast, carry-look-ahead, two-input adder. In the relatively infrequent case that all three of the above-mentioned components must be added to produce the effective address, a second pass through the address-generation circuitry and an additional cycle are required. A single level of indirect transfer orders is provided via hardware. For indirect transfers, the effective address generated as indicated above represents the location (i.e., address) of the address to which the instruction transfers. Since a memory read must be performed, either one or two additional cycles are required. Vector table addressing is the method of indirect addressing used to address locations external to a program unit. The location of the external address is formed as the sum of the beginning of the vector table (wired address) and the data field of the instruction that serves as an index into the vector table of addresses.

Interject transfers, like vector table transfers, employ wired addresses. The interject facility is discussed in Section 3.3.

A 64-word pushdown stack for return addresses is provided to facilitate subroutine transfers. The top of the stack is the stack (S) register in the central control, and the remainder of the stack is in a reserved location in call store. Before a return address is saved in the stack register, the previous contents of that register are stored in the call store location pointed to by the stack counter. The stack (push) option requires an extra cycle to process. The pop return-address option returns the last data stored in the call store stack to the S register. The pop option requires an extra cycle to process only if the call store is a 1400-ns unit. Automatic increment or decrement of the stack counter is performed with check for overflow or underflow each time the stack is pushed or popped.

The J return-address option, provided chiefly for compatibility with No. 1 ESS translated programs, saves the return address for successful transfer instructions in the J register, which is one of the general-purpose registers. In general, execution of the J option, which saves the address of the instruction immediately following the transfer, does not add to the number of cycles required to process an instruction.

Three index-register-modification options are available: add-one (A) option, set-register (S) option, and word (W) option. These options are available on a subset of the long instructions, some of which offer only the A option due to limited code space.

The add-one option causes the contents of the index register to be incremented after the contents are used in the indexing operation. On a store instruction, if the index-register field is null, the A option is interpreted as add one to memory. In this case, the register contents to be stored are incremented before the data are stored. On certain conditional transfer orders with the A option specified, the test register is incremented after the test has been performed but only if the test is successful.

For the set-register option, the index register is not used in address generation. Instead, the specified register is set to the value of the effective address. The word option is the same as the set-register option except that the contents of the index register are used in address generation in that case.

Index-register modification does not add to the time required to execute an instruction.

3.2.2 Typical instruction timing and cycle counts

In several cases, two normally separate instructions are encoded as a single instruction. These "combined" instructions are designed to make efficient use of the time that the CC execution circuitry is normally idle on memory-access instructions to 1400-ns memory units waiting for either a data reply (read) or a completion indication (write). During the otherwise idle time, a shift or rotate operation is performed on the data in another (usually unrelated) general-purpose register.

A combined instruction may perform a load, add, compare, product, union, or exclusive-OR read operation. The data reply from the store may be product masked with the contents of the mask register and/or complemented before the main logic operation (add, compare, etc.) is performed. Alternatively, the instruction may perform a write operation with either product or insertion masking with the contents of the logic register. Figure 5 illustrates how the adjust operation is fit into the idle data processing phase.

Minimum cycle counts for various types of orders are shown in Table II. The actual execution time for a program depends on conflicts between fetch and execution control, instruction mix, and auxiliary unit data transfers.

3.2.3 Typical instruction encoding

As previously mentioned, the length of an instruction is often determined by the length of the required data fields or by the particular options specified. Figure 6 shows the encoding of two typical conditional transfer orders: one short, the other long. Both of these instructions offer relative addressing, but the single-word order allows less range. The double-word order allows indexing and/or indirect addressing, while the single-word does not. Both orders provide a stack return-address option, but only the long instruction offers a J-return-address option or an

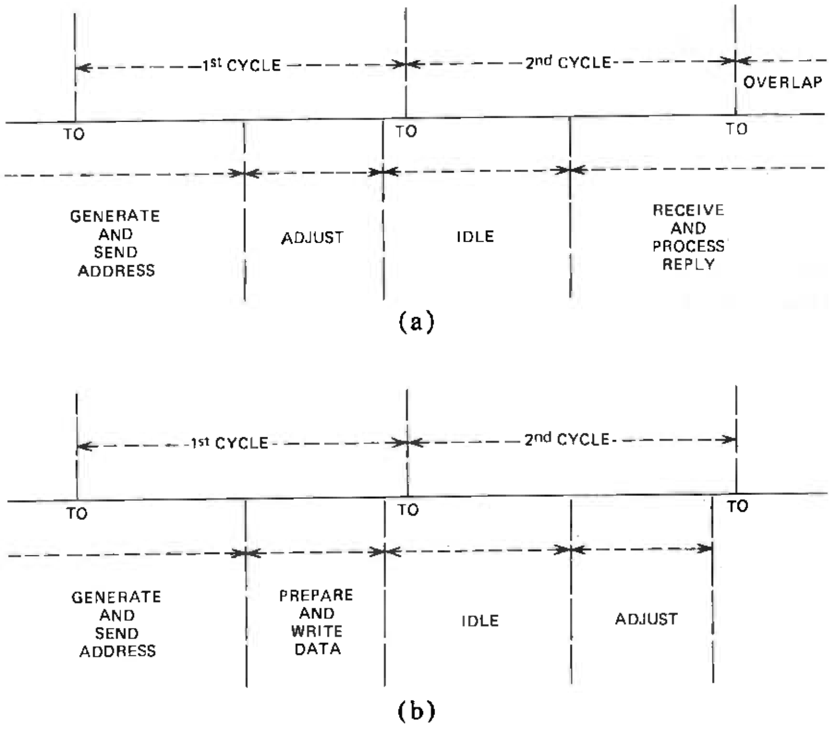
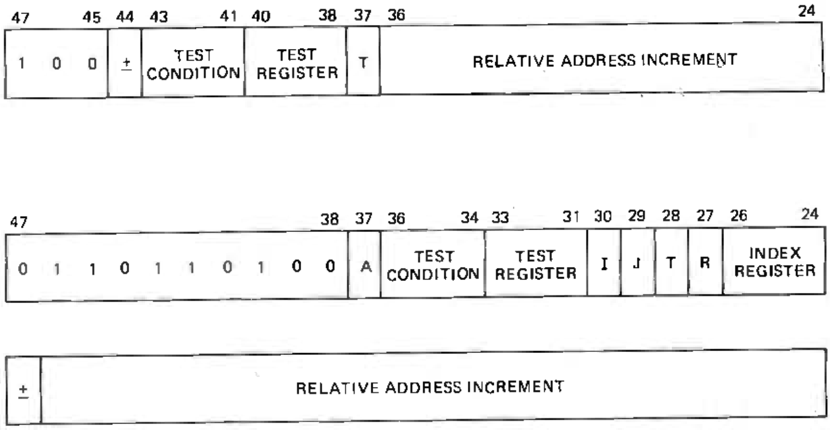


Fig. 5—Combined instruction timing. (a) Combined load. (b) Combined store.



GLOSSARY

A	ADD ONE OPTION
I	INDIRECT ADDRESSING
J	J RETURN ADDRESS OPTION
T	STACK RETURN ADDRESS OPTION
R	RELATIVE ADDRESSING

Fig. 6—Short and long instruction encoding.

Table II — Typical cycle counts

Operation	1400 ns Store	700 ns Store
EA $\circledast$ REG $\rightarrow$ REG	1	1
MEM $\circledast$ REG $\rightarrow$ REG	2	1
EA $\rightarrow$ MEM	2	1
Transfer direct	2	1
Transfer indirect	4	2
Combined	2	2
Search (N words)[†]	2N + 1	N + 1

EA	Effective address.
MEM	Any memory location.
$\circledast$	Operation.

Note: Add 1 cycle if order is both relative and indexed.
[†] The search instruction facilitates a sequential search of memory for a selected code.

add-one option. For the particular long transfer selected, the A option causes incrementing of the test register rather than the index register if the test is successful.

3.2.4 Peripheral operations

Central control monitors and controls the peripheral system by transmitting information to the peripheral units (PUs) which perform the functions of scanning, signal distribution, and network control. The CC communicates with the PUs by one of three methods. The first method, coded enabling, involves addressing PUs by transmitting an enable code to be accepted by the appropriate unit in a manner analogous to store operations. The second method, CPD enabling, requires a central pulse distributor such as that used in No. 1 ESS. Polling, the last method, employs a maintenance control pulse to obtain status from several PUs simultaneously.

Peripheral units may be located as far as 137 meters and 385 meters from the CC for CPD and coded enabling, respectively. Upon initiation of a PU operation, control is passed from instruction-execution control to peripheral-operation control, and nonconflicting instructions may be processed during the remainder of the peripheral operation. Conflicting instructions are those that either use or change the logic register, peripheral data register, or peripheral enable address register. If execution control encounters a conflicting instruction, it stalls until informed by peripheral operation control that the peripheral operation is complete.

Coded enable operations are performed on a variable-cycle basis, except for a limited class of orders intended for line scanning which are executed at a fixed rate. Variable-cycle operations were adopted to gain speed during peripheral operations. The first two cycles of coded-enable operation are used to set up the enable address and peripheral data registers and may require a store access. Address and data are transmitted in the third cycle and the CC waits as long as 30 additional cycles for a response. If no completion indication is received by the end of that interval, an interrupt flag is set. CPD enable operations are performed on a fixed-cycle basis with different types of operations requiring different numbers of cycles. Operating on a predetermined cycle basis facilitates the overlap of repetitive operations. When operating in an overlap mode, repetitive scanning operations typically require either 10 cycles (trunk or line scanning) or 11 cycles (digit scanning), and nonscanner operations repeat in 14 cycles. In a typical sequence, the first two cycles are required to set up CC registers. Address information is sent to the CPD in the third cycle. The CPD transmits the peripheral address shortly thereafter, and the CC awaits completion of the operation. On scanning operations, a data reply into the logic register is expected.

Each PU which receives a polling control pulse from central control returns a status indication such as whether or not a data buffer is full. Units respond into uniquely defined positions in the reply register. A PU polling operation consumes 32 cycles unless it is terminated early by encountering a logic-register-usage instruction which, in effect, allows a software-controlled early termination.

3.3 Interrupt system

An interrupt causes program control to be passed from the current program to the program corresponding to the interrupt level. The higher priority interrupt levels are maintenance oriented whereas the lower levels are processing oriented. The immediacy of interrupt recognition by central control is a function of the type of interrupt (maintenance or processing), the level of the interrupt, whether the software-controlled inhibit is set, and the instruction sequence currently being executed. Under certain conditions, for example, couplets of instructions or AU data transfers may be allowed to complete before an interrupt is recognized.

When an interrupt is recognized by central control, the interrrupt system saves vital information in memory that might otherwise be destroyed before it could be saved via software control. This vital information includes the return program execution address, the contents of the data buffer register, and the value of the control flip-flops. The last program word fetched (ABL/ABR) is saved if the error indication points to a problem encountered on that fetch operation. On maintenance levels, the interrupt system also freezes a group of save registers which are useful for fault recognition (save data address, save program address, save current address, and so on). When a CC mismatch is detected, the matchers are halted to preserve the mismatched data. Finally, the interrupt system initializes fetch and execution control and sets up a transfer to the appropriate interrupt level program. Having completed this sequence, the interrupt system releases fetch and execution control and allows software to perform its tasks. Next, a go-back-to-normal sequence is initiated which restores the CC to its preinterrupt state. The severity of the interrupt will determine whether or not the interrupted program is continued.

For fault situations, it is important that the level of interrupt taken be the lowest consistent with the failure indication. A main memory parity failure, for example, is not allowed to cause a CC mismatch since the latter is a higher level of interrupt.

The processing level interrupts are provided in the CC to facilitate the administration of software tasks such as input/output which must be performed within specified intervals of time. Similarly, the interject facility is used to efficiently interpose priority processing in the normal cycle of program tasks. Flags (flip-flops), which indicate that interject tasks need to be performed, may be set in the CC either by program control or by hardware in various units external to the central control (e.g., auxiliary units) that require attention. Some of these interject flags are maintenance oriented but are deferrable and should therefore not set a maintenance-level interrupt source. Certain base-level programs utilize transfer instructions which test the interject flags. An automatic check is made on this mechanism by a processing level interrupt which occurs if no interject work is performed within 10 ms.

3.4 Auxiliary unit data transfers

Direct handling of the data involved in the auxiliary unit (AU) data transfer tasks by CC execution control would require a significant amount of real time and, therefore, the control of these data transfers is handled largely by separate control on a cycle-stealing basis.

Although the AU system is connected directly to the central controls, the CCs do not have direct access to the data stored on the disks and tapes. Instead, the requests that data be transferred between bulk storage and main memory are placed in main memory by administrative software. The auxiliary unit controller receives the location of request details and initiates the data transfer. Once a job is initiated, the AU system processes the request by transferring one word at a time through the CC, allowing the CC to resolve the bus conflicts between itself and the AU. The CC gives the AU bus priority about 25 percent of the time to keep AU buffer overflows within bounds.

3.4.1 AU priority system and signaling

Central control can accommodate up to 16 auxiliary unit interface ports. In the case of multiple requests from AUs for bus usage, the CC administers bus granting on a priority structure. The AU ports are divided into two groups of eight members each. Any AU within the first group may, under program control, be assigned highest priority; members of the second group have a wired priority that is lower than that assigned to any members of the first group.

Each AU memory operation requires three cycles. The first cycle is used to establish communication between the CC and an AU. The following two cycles correspond to the actual memory operation during which the AU interface control in the CC controls buffering of the address and data information. The buffering is performed so that all memories appear to the AU to be 1400-ns memories.

Auxiliary units requiring access to the store submit requests to the CC near the beginning of the first cycle. The AU interface control in CC sends an enable to the requesting AU with the highest priority. The enabled AU replies with address information and a verify signal. If the communication path is available, a storage-access-permitted signal is sent to the AU and the AU returns a recognition signal. If the communication path is not available, a storage-access-permitted signal is not sent and the AU is blocked and must request again.

3.4.2 Bus usage administration

The operations of instruction fetching, data fetching and storing by execution control, and passing of information between main and auxiliary storage all require access to main memory. To conserve real time, execution control is normally given highest priority, fetch control is given second priority, and the AU system is normally relegated to lowest priority. However, the number of AU data-transfer tasks that must be aborted due to overflow of buffers is minimized by allowing AU system bus usage if this system is blocked in three successive attempts to obtain bus usage. The fetch or execution control may use the bus (call or program store) not required by the AU system since only a single AU task is allowed memory access at any time.

3.5 Address structure and memory protection

An address containing 22 bits allows accessing memory of up to 4.2 million words. For compatibility with No. 1 ESS programs, an alternate mode is available under program control that uses only 21 bits. In either case, the lower half of the address spectrum is assigned to call store. Except for a few thousand addresses near the top of the spectrum (which are used for direct access of CC registers, file store controller registers, and so on), the upper half of the spectrum is assigned to program store.

It is essential to protect areas of the memory spectrum which are not duplicated within main memory such as program and translation stores, and areas which vitally affect system operation such as CC internal registers, disk controller registers, main program constants, interrupt bins, etc. Therefore, the memory address spectrum is divided into protected and unprotected areas each requiring different write instructions. An interrupt results if the address is incompatible with the instruction. Typically, only a portion of call store lies in the unprotected area. The lower- and upper-protected area registers in the CC, which define the secure and unsecure areas of the spectrum, are software controllable via secure write instructions.

IV. PROCESSOR COMMUNICATIONS

4.1 Definitions and characterization

Communication between units of the 1A Processor and between the 1A Processor and the network take place over a system of high-speed digital communication channels called buses. Each bus is made up of a number of parallel channels and each channel consists of a balanced wire pair and associated transmitting and receiving circuits. Transformer coupling is employed between the bus and the transmitting and receiving circuits.

Information on a bus is represented by the presence or absence of pulses on the individual wire pairs. Information is passed between units in parallel form with pulses sent simultaneously over a group of wire pairs.

4.2 Objectives of the bus system

The 1A Processor bus system was designed to meet two primary objectives: to provide reliable communications and to be compatible with the No. 1 ESS peripheral units. To meet these objectives, the various bus groups are duplicated as shown in Fig. 7. Bus duplication methods are covered in a later section. In addition to required information channels, each bus provides channels for parity bits which are used for error detection. Pulse transformers help to achieve reliable bus operation. These pulse transformers have electrostatic shields and high-voltage breakdown characteristics for protection against longitudinal noise signals and induced currents or voltages caused by lightning strikes or power faults. Further, failure-conducive stress conditions on bus circuit devices are avoided by using devices well within allowable operating ranges.

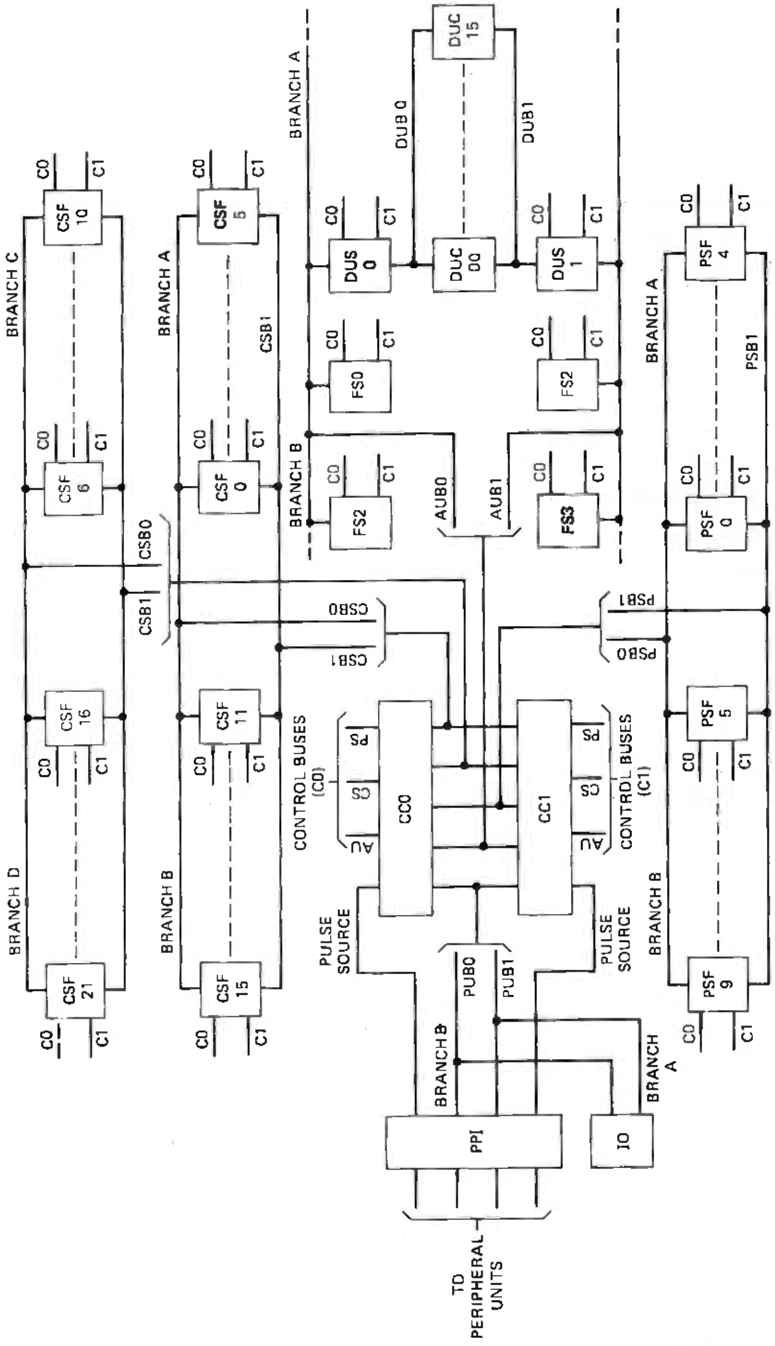


Fig. 7—Bus interconnections.

Another bus system objective is to achieve a speed compatible with the 1A Processor digital circuits. A cable with irradiated polyvinyl-chloride (IPVC) insulation is used between frames. This cable has controlled propagation delay and pulse-dispersion characteristics to minimize delay variations and degradation of pulse rise times. Within the processor frames, Teflon*-insulated cable is used for proper impedance matching and low propagation delay.

Bus driver and receiver circuits are also designed to minimize propagation delays and delay variations. This has been made possible by placing all circuit components for drivers or receivers on a circuit board and by the electrical design itself.

Another important feature is ease of bus cable installation and improved maintenance features. Use of pretested, connectorized bus cables between frames permits quick initial installation and simplifies subsequent bus growth for system additions. Connectorized cables also eliminate the common problem of wiring bus pairs incorrectly during bus installation.

Bus repair time is reduced with the use of connectorized cables and with bus circuit components mounted on plug-in circuit packs. To troubleshoot bus problems, access points are provided on bus-termination resistor assemblies. These resistor assemblies are connectorized and slip over the connectors of bus packs located at the ends of a bus.

4.3 Bus configurations

Two types of buses, shared and private, are used for communication between units of the processor. Shared buses branch out from the central control in two directions and are multipled from frame to frame, carrying data that is common to all frames on a bus. Private buses are connected between the CC and individual frames and are used for synchronization and control information. Cable assemblies consisting of eight wire pairs are used for both private and shared buses.

4.4 Bus system characteristics

The various bus groups are arranged into two main bus systems, the processor-unit bus system and the peripheral-unit bus system. Buses interconnecting units of the processor are included in the processor system. The peripheral bus system consists of buses providing communication between the processor and peripheral frames.

The processor bus system is designed for minimum delay. This objective was achieved by using the following:

* Tradename of E.I. Dupont de Nemours & Co.

(*i*) A high-speed cable driver to minimize circuit delay and recovery time.

(*ii*) A passive cable receiver capable of extracting sufficient energy from the bus to drive a low impedance without introducing an impedance discontinuity and unwanted pulse reflections.

The properties of the processor bus system are tabulated in Table III. A pulse amplitude of 7.2 V is determined by the minimum signal required by a receiver at the end of a bus having the maximum number of 10 receivers connected to it. Adequate pulse width at this receiver under worst-case timing variations is provided by a nominal 100-ns-wide pulse. The narrow pulse allows use of 700-ns memory units as well as 1400-ns memory units.

To minimize the worst-case bus cable delay on call-store and program-store buses, these cables are routed in a loop between the central control and the stores. That is, address and write cables are connected to the first store frame on a bus branch while the data-reply cables are run directly from the furthest frame. The shorter length of the reply cable compensates for the furthest store having the longest bus distance for address and write data.

The peripheral unit bus design provides an adequate signal over a long bus having up to 50 receivers connected to it. This design was implemented by the use of the following:

(*i*) A cable driver that provides a 500-ns bus pulse to compensate for pulse dispersion (rise-time degradation) and bit-to-bit timing variations inherent with long, fully loaded buses.

(*ii*) A cable receiver with an active gate for bus signal amplification, resulting in a minimum amount of energy being extracted from the bus.

Table III also lists the peripheral bus system characteristics. For compatibility with the No. 1 ESS peripheral units,[1] the 9.3-V pulse amplitude provides a sufficient signal for 50 receivers over a 137-m bus length. A minimum cycle time of 2800 ns was determined to be the time needed for bus circuits to recover, and permit system execution of peripheral instructions at maximum speed.

Bus length, as shown in Table III, may be increased by the use of a repeater for regenerating bus pulses in order to reach a maximum 385-m length with up to 50 receivers on the bus.

4.5 Bus system organization

Four bus groups make up the processor unit bus system: the call store bus (CSB), program store bus (PSB), auxiliary unit bus (AUB), and data unit bus (DUB). The call store bus group consists of two buses or four

Table III — Bus characteristics

Bus	Bus Pulse Amplitude (v)	Bus Pulse Width (ns)	Min Cycle Time (ns)	Max Loop Length* (m)
Call store	7.2	100	700	18.6
Program store	7.2	100	700	18.6
Auxiliary unit	7.2	100	700	30.5
Data unit	9.3	500	2800	143.2
Peripheral unit	9.3	500	2800	274.3†
				769.4‡

* Bus length includes path length through connectors and circuit packs.
† With 50 receivers connected.
‡ With bus repeater.

Table IV(a) — Call store bus layout

Bus Group	Bit Names	Function/Definition
Call Store Address	CSA00 → CSA15	Data location address
	CSA16 → CSA20	K-code
	CSAMP	Address parity
	CSA3T5	Timing
	CSA1T3	Timing
	CSAM	Maintenance mode
	CSAC	Control mode
	CSAW	Write mode
	CSAR	Reply mode
Call Store Write	CSWE	Write enable
	CSWP2	Data parity
	CSWP1	Data parity
	CSW00 → CSW23	Data
Call Store Reply	CSR00 → CSR23	Reply data
	CSRP2	Reply parity
	CSRP1	Reply parity
	CSRAW	All seems well
	CSRAF	All seems well fail
Call store Clock Pulse source	CS0CLK	Bus 0 synch (5T7)
	CS1CLK	Bus 1 synch (5T7)
	PSCS	Pulse source
Call Store Status display	Update	Update status display
	CSDSK0 → CSDSK4	K-code F/F's
	CSDSMT	Maintenance F/F
	CSDSR0	R0 F/F
	CSDSA0	A0 F/F
	CSDSA1	A1 F/F
	CSDSVF	Update verify

branches to accommodate the full complement of call stores. The DUB differs from the other processor buses in that it interconnects data-unit selectors within the processor and data-unit controllers that are physically located beyond the processor area. As a result, the maximum DUB

Table IV(b) — Program store bus layout

Bus Group	Bit Names	Function/Definition
Program Store Address	PSA00 → PSA15	Data location address
	PSA16 → PSA20	K-code
	PSAMP	Address parity
	PSA3T5	Timing
	PSA1T3	Timing
	PSAMT	Set maintenance F/F
	PSAM	Maintenance mode
	PSAC	Control mode
	PSAW	Write mode
	PSAR	Read mode
Program Store Write	PSWE	Write enable
	PSWP2	Data parity
	PSWP1	Data parity
	PSW00 → PSW23	Data
Program Store Reply	PSRR00 → PSRR23	Right word reply data
	PSRRP2	Right word parity
	PSRRP1	Right word parity
	PSRL00 → PSRL23	Left word reply data
	PSRLP2	Left word parity
	PSRLP1	Left word parity
	PSRAW	All seems well
	PSRAF	All seems well fail
Program Store Clock-Pulse Source	PS0CLK	Bus 0 synch (5T7)
	PS1CLK	Bus 1 synch (5T7)
	PSPS	Pulse source
Program Store Status Display	Update	Update status display
	PSDSK0 → PSDSK4	K-code F/F's
	PSDSMT	Maintenance F/F
	PSDSR0	R0 F/F
	PSDSA0	A0 F/F
	PSDSA1	A1 F/F
	PSDSVR	Update verify
Program Store Processor Configuration	VPSPCB	Set PS0 K-code
	R0PCRB	Reset R0 F/F
	R0PCSB	Set R0 F/F
	MTRCB	Reset maintenance F/F

Table IV(c) — Auxiliary unit bus layout

Bus Group	Bit Name	Function/Definition
Auxiliary Unit Address	AUA00 → AUA10	Control register address
	AUA11 → AUA15	K-code
	AUAR	Read mode
	AUAW	Write mode
	AUACTL	Control mode
	AUAPKA	Address parity
	AUASYNC	Synch
Auxiliary Unit Write	AUW00 → AUW23	Data
	AUWP2	Data parity
	AUWP1	Data parity
	AUWAWF	All seems well fail
	AUWASW	All seems well
Auxiliary Unit Store Address	AUS00 → AUS15	Data location address
	AUS16 → AUS20	Store K-code
	AUS21	CS/PS selection
	AUSR	Read mode
	AUSW	Write mode
	AUPUC	Control mode
	AUSPKA	Address parity
Auxiliary Unit Reply	AUR00 → AUR23	Reply data
	AURP2	Data parity
	AURP1	Data parity
	AURASW	All seems well
Auxiliary Unit Control	AUSP	Timing
	PCAU	Pulse source initialize
	PAUSE	{ Pump program store (FS0/FS1) System reinitialize (DUS0/DUS1)
	PAUMT	Maintenance
	SAP	Storage access permitted
	PAUEN	Bus enable
	AUBR	Bus request
	AUEV	Verify enable
	AUPUC	{ Pump complete (FS0/FS1) SR complete (DUS0/DUS1)
	AUOPIJ	Paging complete (DUS only)
	AUITJ	Maintenance request

length is much greater than for other processor buses, and the DUB characteristics are the same as the peripheral unit buses.

Within each bus group there are address, write, reply, and control buses. For the CSB, PSB, a status display bus is also included. Table IV tabulates bit layouts for each of the processor unit buses.

One bus group makes up the peripheral unit bus system but a subset of this group is used as a group in itself. This two-group arrangement permits the processor to be used with peripheral systems employing either central pulse distributor enabling or coded enabling. The bit layouts for each system are shown in Table V.

Table IV(d) — Data unit bus layout

Bus Group	Bit Name	Function/Definition
Data unit Address	DUA00 → DUA05	DUC register address
	DUAR	Read/write mode
	DUAC	Control mode
	DUAP	parity
	DUACS	Clock synch
	DUAES	End of sample
Data unit Write	DUW00 → DUW23	Data
	DUWP	Data parity
Data unit Reply	DUR00 → DUR23	Reply data
	DURP	Reply data parity
	DURAS	All seems well
Data unit Control	MI	Maintenance interject
	OI	Operational interject
	R	Read
	W	Write
	P	Parity
Data unit System Reinitialization	DUSRR	System reinitialization ready
	SRNTRY	System reinitialization not ready
	SRERR	System reinitialization error
	DUSRC	System reinitialization complete
	DUSSRA	System reinitialization activate
Data unit Select	DUS	DUC select
	DUSV	DUC select verify

4.6 Processor peripheral interface

Connections from the processor to the peripheral system are made at the peripheral processor interface (PPI) frame as shown in Fig. 7. All processors are equipped with branch B of the peripheral unit bus connected to the PPI. This provides a connection point for the peripheral system. For systems requiring bus lengths greater than that provided by one branch, branch A of the peripheral unit bus is also connected to the PPI. In addition, the PPI serves as the distribution point for the peripheral pulse source buses. These are groups of control pulse pairs. Each pair is connected to one peripheral frame in a manner similar to the private control buses used within the processor units.

4.7 Bus circuit details

Cable drivers and receivers provide the interface between logic circuits within frames and the interconnecting bus cable, as shown in Fig. 8 for central control and call store frames. Driver circuits are shunt-connected to the bus using transformer coupling. Receiver transformer connections are made in shunt or series-shunt as shown. A series-connected receiver

Table V(a) — Peripheral unit bus layout—coded enabling

Bus Group	Bit Name	Function/Definition
Peripheral Unit Enable address	PUEA00 → PUEA12	Enable address
	PUSYNC	Enable address synch
	IOCH	IO clock high
	IOCL	IO clock low
Peripheral Unit Write	PUW00 → PUW36	Data
	PUW37	*
	PUPOD	Odd address parity
	PUPEV	Even address parity
Peripheral Unit Reply	PUR00 → PUR23	Reply data
	PUASW	All seems well
	PURP	Reply data parity
	APUF	Autonomous PU F level
	APUT	Autonomous PU trouble
	APUB	Autonomous PU base level
Peripheral Unit Control	MI	Member interrogate
	GI	Group interrogate
	CLKI	Clock interrogate
	PHP	Poll high priority buffer
	PLP	Poll low priority buffer
	PSZ	Poll seizure buffer
	PDG	Poll digit buffer
	PIO	Poll input output
Peripheral unit Loop around	M0	Mapping bit 0
	M1	Mapping bit 1
	M2	Mapping bit 2
Maintenance	X	Execute

* Cabled to, but not used by system.

is used on the peripheral bus. For protection against longitudinal noise signals, a balanced bus is used.

Diodes D3, D4 provide isolation between drivers and also minimze any bus unbalance due to drivers. Circuit recovery time is controlled by D1 and D2. To protect the output transistor Q1 from burnout due to a stuck driver input, capacitor C1 is provided. With normal inputs applied, C1 is charged to a low voltage during the pulse interval and discharged between pulses by the recovery gate (REC). If an input is permanently low, C1 is charged through R1 to +3 V, back biasing Q1 and turning it off in time to protect against any damage. This protection feature is tested by holding the test (TEST) lead low, which inhibits all eight driver circuits on a pack and activates the monitor (MON) gates causing the test monitor (MON) lead to go low.

The inhibit action of the test lead also serves the useful function of controlling cable drivers when bus circuit power is turned down or turned on. This control is necessary to prevent false pulsing on the bus during these intervals. In frames that have one common power switch such as the CC, power control circuits cause the test lead to be held low prior to

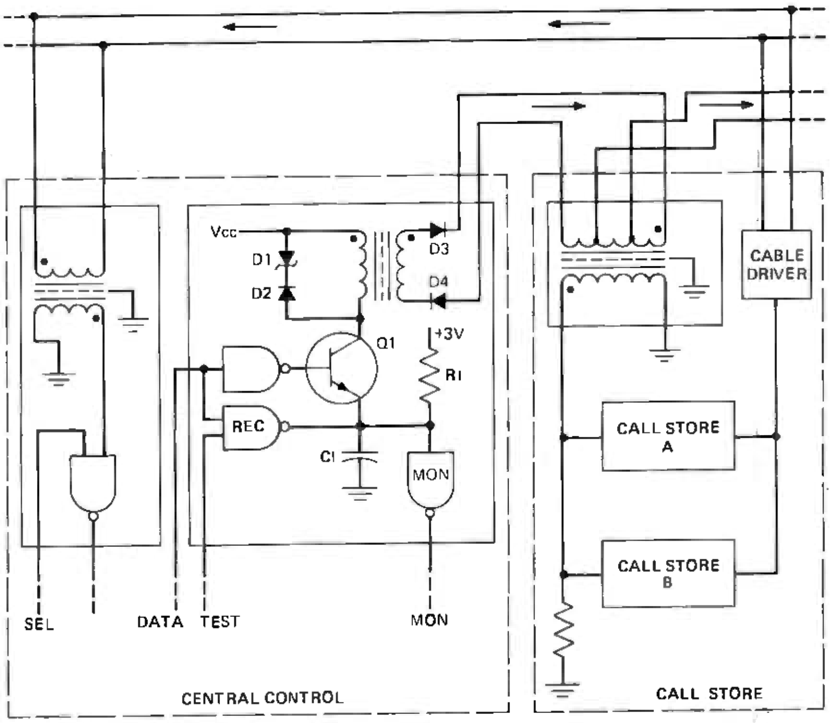


Fig. 8—Central control/call store circuits.

removal of V_{CC} or after application of V_{CC} for an interval sufficient to insure against false outpulsing. In frames that have a separate power switch for bus circuits such as CS and PS, control of the test lead is not necessary if the +3 V transition time is greater than 1 ms. In some frames, program control of the test lead is provided to select and inhibit cable drivers.

Cable receivers in processor frames other than the CC employ a series shunt-connected transformer to maintain impedance matching on the bus and also to provide drive capability into a low impedance. This is important in frames such as CS, PS where a 100-ohm intraframe bus interconnects individual memory units and the receiver output. These receivers contain eight circuits per pack.

Shunt-connected receivers are used in the CC to allow communication to one receiver from both branches of the reply buses. These receivers also contain logic gates that provide for selection of all 16 bits or individual selections of 2 bits on a pack.

To minimize the power loss on the peripheral buses, a series-connected transformer with an output buffer is used as a receiver. The high turns

Table V(b) — Peripheral unit bus layout—CPD-enabling

Bus Group	Bit Name	Function/Definition
Peripheral	PUEA00 → PUEA12	Enable address
Unit	PUSYNC	*
Enable	IOCH	*
Address	IOCL	*
	PUEA13 → PUEA31	Enable address
	T	CPD test
	R	CPD reset
	BC	CPD bus choice
Peripheral	PUW00 → PUW37	Data
Unit	PUPOD	*
Write	PUPEV	*
Peripheral unit	PUR00 → PUR15	Reply data
Reply	PUASW	All seems well
CPD	VA00 → VA23	Verify answer
Verify answer	ASWCPD	CPD-all seems well
CPD	EX00 → EX15	CPD execute
Execute		
CPD	EXR00 → EXR15	CPD execute return (Echo)
Diagnostic	A PAR	CPD A parity
Echo	B PAR	CPD B parity
	C PAR	CPD C parity
	M1	CPD maintenance
Peripheral unit	PULAM0	Mapping bit 0
Loop around	PULAM1	Mapping bit 1
Maintenance	PULAM2	Mapping bit 2
	PULAX	Execute

* Cabled to, but not used by system.

ratio of the transformer (1:25) reflects a very low impedance in series with the bus; this keeps the power loss to approximately 0.05 dB per receiver. A receiver pack contains eight circuits.

4.8 Cable details

An 8-pair switchboard cable with an irradiated polyvinylchloride (IPVC) insulation is used for the communication bus. Use of the IPVC insulation provides improved thermal properties and allows cables to be soldered to connectors. The cable has a characteristic impedance of 100 ohms and a maximum propagation delay of 5.75 ns/m. Pulse dispersion of the cable is specified over a 21.3-m length for an input pulse of 5-ns rise time, with the increase in rise time of the output pulse to be less than 5.5 ns.

V. MAN-MACHINE INTERFACE AND CONTROL

The 1A Processor has been designed with a flexible, expandable control and terminal access capability and a common hardware interface for both No. 1 and No. 4 ESS peripheral networks. To tailor the control and terminal features to the specific using system, the processor utilizes two hardware subsystems. These are the processor peripheral interface and master control console (PPI/MCC) and the input/output (IO) frame. They are located on the processor peripheral bus.

Splitting IO and MCC frames is a departure from the No. 1 ESS processor which combined the maintenance TTY channel as part of the master control center. In No. 1A ESS, maintenance data are provided on several channels. Each channel is implemented through the IO subsystem, which may grow in channel capacity as needed. The AMA tape unit is not located with the 1A Processor PPI/MCC, but is supplied as a separate unit on the auxiliary unit bus system. The memory card writer system, which is part of the master control center of the No. 1 ESS processor, is not needed since all 1A Processor memory is locally writable.

5.1 The PPI/MCC

The PPI/MCC complex serves several functions. First, it provides the control panel interface containing manual keys and displays. The keys and displays of the PPI/MCC may be divided into three groups. The first group includes all critical control and display functions required for manual system configuration and recovery control during emergency conditions when the automatic features have failed. The functions include system reinitialization, hardware reconfiguration of the duplicated processor, partial software system initialization, and processor status indications. The second and third groups of keys and displays are the processor and specific using systems panels which provide direct data entry and display, bar graphs of network activity, and system and network status indicators.

A second PPI/MCC function is to act as the single processor interface to the peripheral network. All peripheral bus interconnections are terminated at the PPI, which facilitates testing, installation, and No. 1 ESS retrofit. The third major function of the PPI is to provide power scanning and power switch lamp control, continually monitoring the status of power switches in the various processor frames. Each power switch is equipped with two scan points indicating four frame states: power on and in-service, a manually initiated state in which the frame is requested out of service (ROS), a manually powered down state, and a state indicating frame power removed by fault-sensing power circuits in the frame. Two lamps are driven from the PPI for each power switch. The acknowledge lamp (ACK) is used to indicate that the processor has recognized a manually requested change in frame state, and the out of service (OS) lamp is used to indicate that the frame has been removed from active service in the system.

The scan data are accessed routinely by the processor operating system. Thus, the frame can be smoothly moved from "in service" to "standby" status at the request of craft. Scan data are also used to initiate system reconfiguration routines to recover from the loss of an automatically powered down frame and contain the current list of available frames for use by system reconfiguration and recovery programs.

5.2 PPI/MCC configuration

The processor peripheral interface and master control console are housed in two separate frames. All keys and lamps are located on panels in the master control console and are driven and powered over cable from the PPI, which may be up to 61 m away. The MCC can be placed in a convenient location, usually with other maintenance equipment. Figure 9 shows the PPI and MCC configurations for the No. 4 and No. 1A ESS applications.

5.3 Master control console

The MCC consists of three panel sections containing keys, lamps, LED displays, and terminal connections. Of these three sections, two are common to both using systems and deal only with processor control and display functions, the processor display panel and the processor request panel. The third panel section is a system application panel, which meets the network status and key input needs of the particular using system.

All controls and displays critical to manual configuration and recovery control of the 1A Processor are located on the processor display panel, shown in Fig. 10. This panel is further divided into the processor display, processor update, override control, system reinitialization, and processor configuration sequencer sections. The processor display group of switches, lamps, and LED displays enables craft personnel to monitor basic processor configuration including CC status, bus configurations, and file store status, and request and visually display the store enable

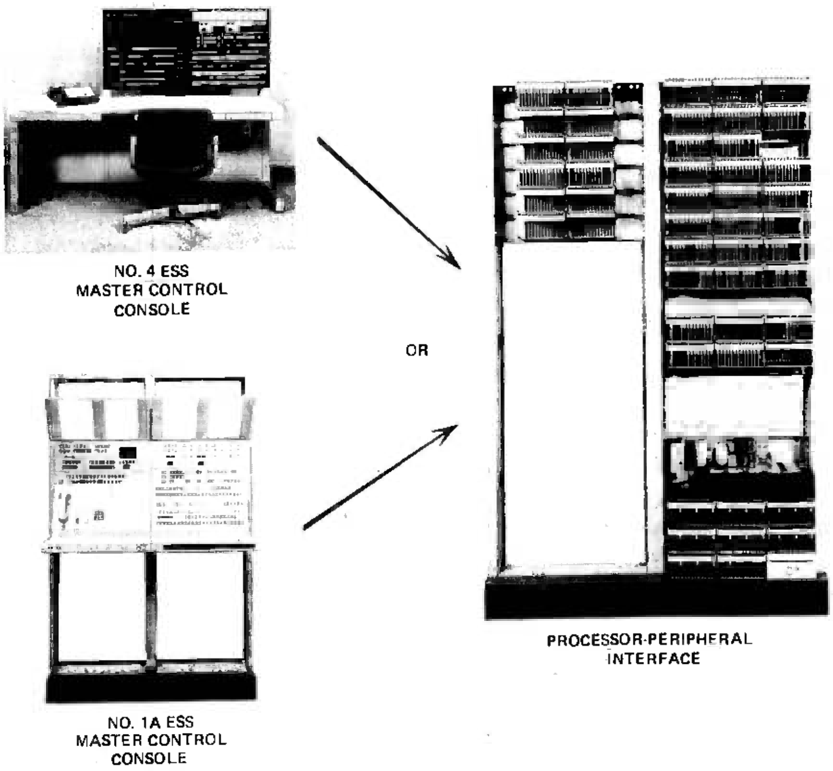


Fig. 9—Master control console and processor-peripheral interface.

code, status, and routing information of any CS/PS unit in the processor. The update group of keys and lamps enables craft personnel to observe when a program update is in progress and prevent the use of a specific file store for emergency recovery. This is used to ensure that a partially written file store is not used during system reinitialization. The override control group of switches and lamps enables craft personnel to manually activate a CC, auxiliary unit bus, program store bus, and basic program store configuration for emergency system recovery. The system reinitialization group of switches and lamps enables craft personnel to manually initiate system reinitialization in conjunction with either the override control or processor configuration sequencer group of switches and lamps. The processor configuration (PC) sequencer group of switches and lamps enables craft personnel to monitor the progress of the PC sequencer during either automatic or manual system recovery using the PC circuit in each CC and manually control the PC circuit for system recovery, or system reinitialization in conjunction with the system reinitialization controls.

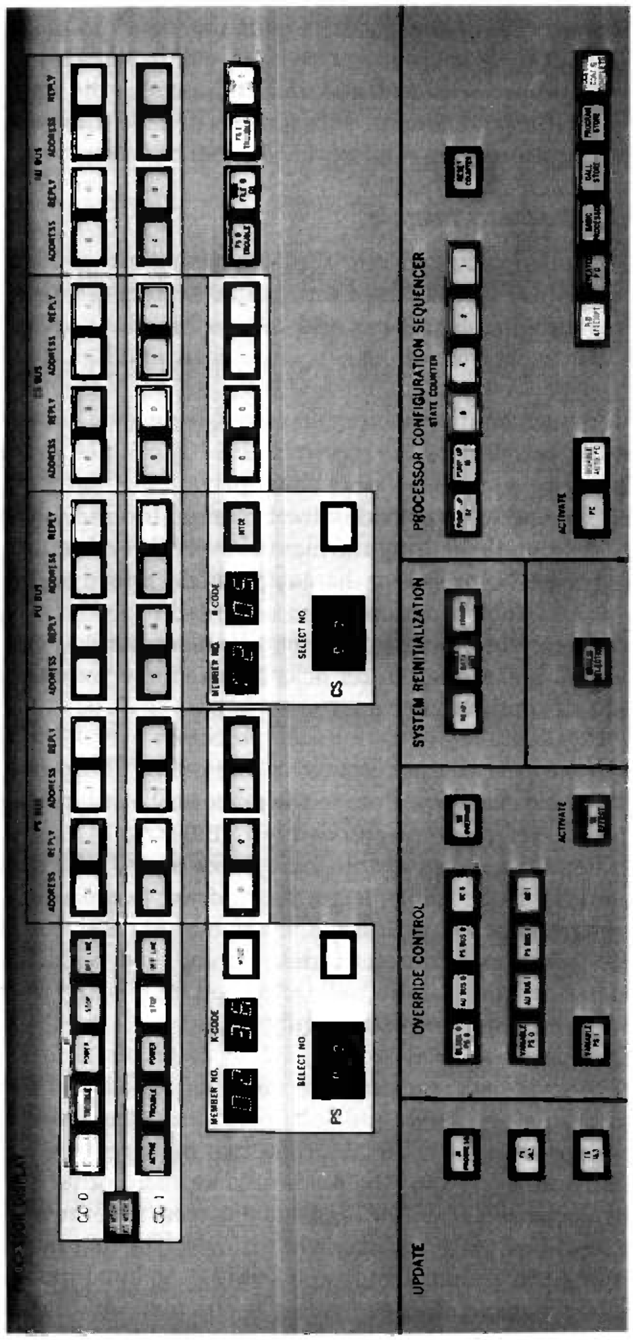


Fig. 10—Master control console display panel.

The processor request panel provides manual interaction with the system software. This panel permits craft personnel to make various A-level interrupt program requests, request and display various system data in binary or numeric form, and monitor or check the status of various MCC indicators and circuits. This panel is divided into two sections: manual interrupt program request and system display.

5.4 Processor peripheral interface

The PPI frame is divided into four functional groups: the critical control and display circuits used for manual system recovery, the peripheral bus circuits, sequencer and memory-readout logic, and the memory and circuits controlling key, lamp, scan, and signal distributor points.

The PPI frame contains memory elements, logic level drivers and receivers, and AC bus drivers and receivers, directly coupled to the central control, file store, and CS/PS stores over private bus paths. These interconnections and logic provide direct manual interaction with processor operation implementing the manual override and reinitialization functions of the MCC processor display panel described earlier. Of the displays on this panel, program access is limited to read only. This ensures that software interaction does not take place during periods when manual control is exercised. Interlocks are used to ensure that invalid manual control combinations are not selected.

Unlike other peripheral bus units, all processor peripheral bus leads terminate in the PPI (whether or not they are used). This provides a bus test capability and single interface to the processor. Much of the bus area of the PPI is, therefore, connectorized terminal strip.

The PPI itself is a coded enable logic frame as are all No. 4 ESS peripherals and the 1A Processor IO frame. Address, instruction, and data are sent down the peripheral bus, and the address is decoded. On an address match, the operation on the data is completed, and if all internal checks are satisfied an all seems well (ASW) and reply parity are returned on the answer bus, along with any reply data.

Most PPI logic is associated with two 32- by 24-bit matrices of flip-flops and status receiver logic; the access to these matrices is achieved by the peripheral bus, lamps, keys, and scan point and other status lead receivers, as represented in Fig. 11. The two matrices are completely symmetrical and, except for the read-only scan and other status indicators, are comprised of either toggle or set-reset flip-flops which are, with few exceptions, PU bus readable or writable. The first matrix is used for controlling the various lamps and storing the key inputs from the various MCC panels; it also provides PU bus read-only access to the configuration status of key processor elements. Approximately one-half

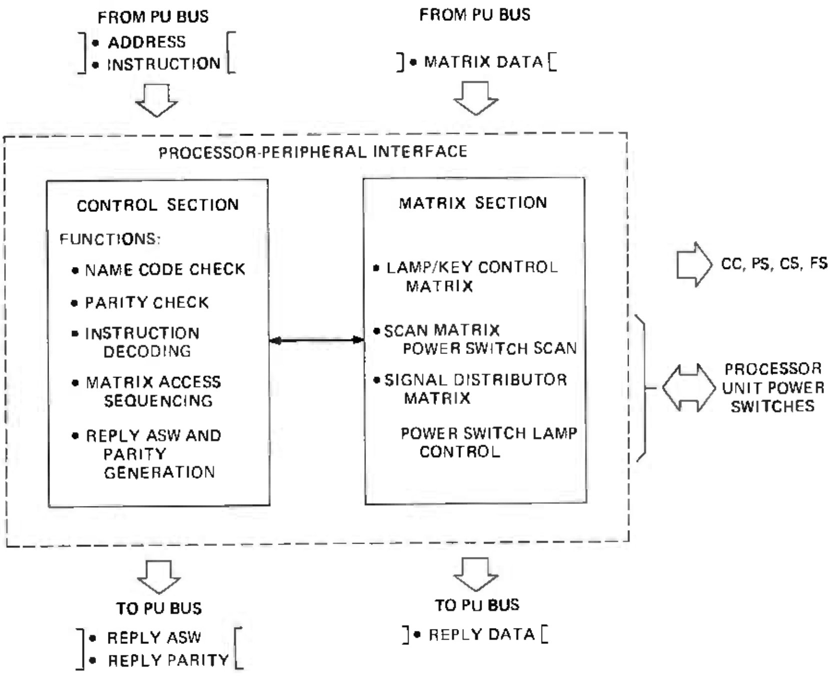

FROM PU BUS
• ADDRESS
• INSTRUCTION
FROM PU BUS
• MATRIX DATA
PROCESSOR-PERIPHERAL INTERFACE
CONTROL SECTION
FUNCTIONS:
• NAME CODE CHECK
• PARITY CHECK
• INSTRUCTION DECODING
• MATRIX ACCESS SEQUENCING
• REPLY ASW AND PARITY GENERATION
MATRIX SECTION
• LAMP/KEY CONTROL MATRIX
• SCAN MATRIX POWER SWITCH SCAN
• SIGNAL DISTRIBUTOR MATRIX
POWER SWITCH LAMP CONTROL
CC, PS, CS, FS
PROCESSOR UNIT POWER SWITCHES
TO PU BUS
• REPLY ASW
• REPLY PARITY
TO PU BUS
• REPLY DATA

Fig. 11—Processor-peripheral interface-operational overview.

of the 672 flip-flops of this matrix are reserved for common processor panel control and display. The second half is available to the using system for its own panels.

The scan and signal distributor (SD) matrix can be subdivided into two symmetrical parts, with half the rows devoted to read-only scanner indications of the status of the various power switches in the office, and the second-half of the matrix devoted to flip-flops controlling power switch lamps on the various frames. The scan and SD matrix is also symmetrical in that corresponding bits in the scan half and SD half are assigned to the same processor unit.

Other minor functions are served by the PPI. A telephone jack circuit for the in-office telephone and an AC driven time-of-day clock are provided, as is a separate 4A timer which provides pulses at several second intervals for timing out certain displays.

5.5 Input/output subsystem features

Craft personnel usually interact with the system via the keyboard/CRT/printer interface. The 1A Processor hardware is capable of supporting up to 96 low- and medium-speed, half-duplex, asynchronous channels with up to three ASCII terminal devices per channel. Provisions are made for either direct terminal interconnection or remote interaction via modem.

A 110 or 1200 b/s EIA RS 232 standard asynchronous terminal or the 20-mA current loop, low-speed teletype interface is currently supported. CRT, printer, keyboard, cassette tape, and paper tape terminals are used for routine maintenance of the system and network, traffic management, system status, recent change information, and retrieval of related data of other using systems.

5.6 The input/output subsystem description

The 1A Processor IO implements the terminal interface to the processor. Each frame consists of a peripheral bus interface and up to two IO unit selectors (IOUSs), which may each he equipped with up to eight IO unit controllers (IOUCs). Each IOUC, in turn, fans out to the three ports with either an EIA standard or the teletype current loop interface, as shown in Fig. 12.

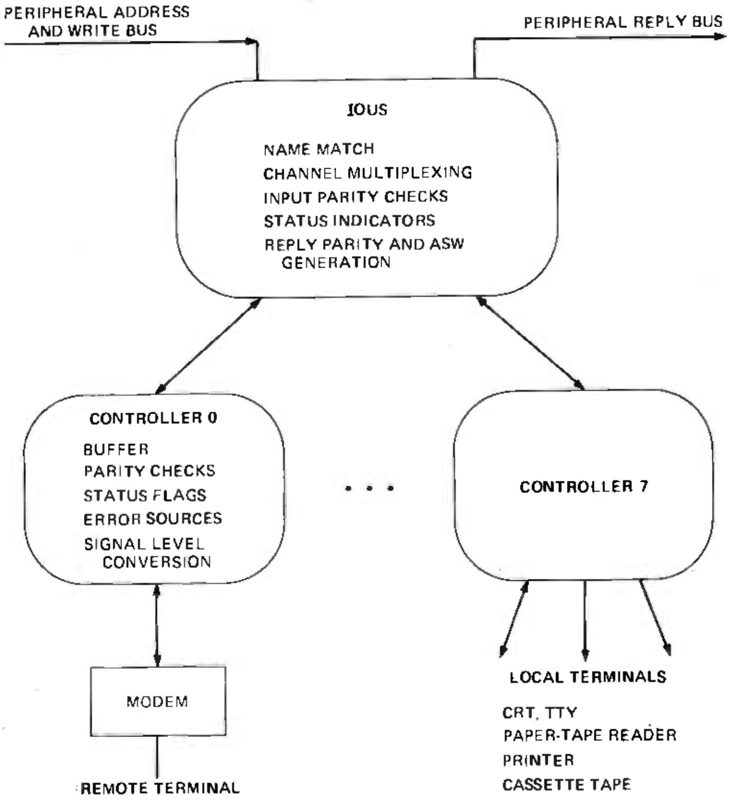


Fig. 12—Input/output frame access.

The IOUS acts as a multiplexer to the peripheral bus. Its primary functions are to provide initial decoding of address, to route data between the processor and each controller, to pass both maintenance-related and normal activity flags back to the processor on request, and to perform common maintenance tasks such as bus parity checking and generation, and validity checks on data and instructions.

The IOUC acts as the asynchronous buffer between the IOUS and a channel. Its primary functions are to buffer incoming and outgoing data, provide parity checks on the ASCII characters, generate service flags to indicate channel state changes and buffer status, and to report maintenance and service flags to the selector upon a poll request.

Three EIA interface ports are provided on each channel. Data coming in one port are echoed on the others during input to the system, and the same data go out on all three ports during system output.

Each processor includes two IOUSs within its fixed floor-space arrangement. This provides up to 16 channels with the basic processor and satisfies the minimum channel requirements for processor operation. Additional channel requirements are satisfied by adding IO frames to the peripheral bus in the network area of the office.

5.7 Input/output polling

The 1A Processor IO system is based on a 60-ms polling scheme. The polling function starts when the CC delivers a poll pulse to all equipped IO circuits. Each IOUS then distributes the pulse to its controllers and collects a 2-bit response from each controller in a poll-request register (PRR) indicating that the controller needs either service or maintenance. Service requests indicate the beginning of an input message, the end of an output message, a break, or a buffer that needs to be filled or emptied. After the polling pulse gates the service or maintenance requests into the PRR, it clears the sources of the requests in responding channels.

The logical OR function of all the requests provides a summary response, gated to the CC only when no requests exist. If any request is found, no polling response is delivered and the PRR must be read to identify the active channels. Since more than one IO selector can be equipped in an office each responds to the polling pulse simultaneously, but in a peripheral unit reply bus bit position which matches its unique enable code.

5.8 General data transfer

All data transfer is initiated by the CC and is carried out through the execution of an instruction sent by the CC. Each instruction to the IOUS begins with a start-of-sync pulse from the CC, sent along with a peripheral bus order, which clears the input registers and the main sequencer. The input registers are then loaded in parallel with an address (or enable code), an instruction, and data. As the gating ends, the enable code is examined to see if it matches the wired address of the IO circuit. If no match occurs, the sequencer stops and waits for the next sync pulse, but if a match does occur, the sequencer continues with instruction execution.

On write instructions, the 24-bit data portion of the data input register is transferred to a destination register specified by the write instruction either in the IOUS or an IOUC. Transmit instructions transfer three 8-bit characters from the input data register to the IOUC character buffer specified by the channel decoder. Read instructions transfer data from the source register in the IOUS or an IOUC specified by the instruction to the reply register. Receive instructions transfer three characters to the reply register from the IOUC buffer specified by the channel decorder. For testing purposes, the IOUS input data register can be connected to the reply register to loop data from the write bus to the reply bus.

As with the PPI, once the operation is complete, an all seems well, parity, and reply data are gated back to the central control if all internal checks have passed.

VI. HARDWARE MAINTENANCE FEATURES

6.1 System features

The need for high reliability and maintainability in the 1A Processor control system imposes important design objectives. Many requirements that concern component selection, subsystem design, system architecture, and program structure were based on the need for very high processor dependability. Complete duplication of processing capability and memory access had to be provided because of its critical impact on system performance. All critical processor communication buses are duplicated in some manner.[4] Input/output that is critical to office integrity is duplicated or backed up with alternate IO channels. Those parts of MCC and PPI that are critical to processing capability or processor maintenance are provided with functional redundancy. The memory backup scheme is also duplicated. The overall system requirement is that all processor capabilities have subsystem backup except when the function is not vital to processor integrity.

In all of the control system units, hardware maintenance features have been provided to allow rapid detection of faults. This permits quick recovery through subsystem reconfiguration and facilitates rapid repair by good fault isolation.

Hardware and software maintenance features are designed to complement each other, with software utilizing the hardware features to achieve maintenance objectives.

6.2 Error detection and recovery

Prime error-detection techniques in the hardware are provided through redundant information (by check bits), matching internal operations, round-trip checks on data and addresses, internal operational checks, and address-range validation. The failure of any of these checks results in the transfer of processor control, via program interrupt, to one of a prioritized set of recovery programs. The level of interrupt and recovery is determined by the potential severity of the fault on system performance and the need for specialized programs to handle fault recovery for each subsystem.

6.3 Power control and alarms

All power that is critical to processor availability is duplicated. Each of a pair of duplicated units is powered from a separate distribution system. Other critical unduplicated units receive power from both power systems, either one of which is sufficient to operate the unit. Each unit has power-sequencing circuitry to allow power removal and restoral without affecting processor operations. Since all units have bus interconnections, the circuitry is required to power logic circuits before bus circuits, control critical routing flip-flops, initialize hardware sequencers, and perform any other initializations required to enable the unit for further program actions. An alarm circuit is provided in each unit to monitor and initiate visual and audible alarms and request IO teletypewriter records for all power failures or manually initiated power operations.

6.4 Central control

6.4.1 Matching

Duplicated and matched CCs form the basic program execution unit of the 1A Processor. All data movement and addressing is carried out between subsystems on duplicated communication buses by way of CC interfaces. The CCs have direct communication paths to all other processor units to effect recovery, reconfiguration, and other maintenance operations.

Normally, the CCs operate in step. Both perform identical operations with one CC active and the other standby. In this configuration, each CC matches itself with its mate to insure that both execute the same instructions and operate on identical data.

Each CC has two separate match circuits, and each circuit has the ability to match 24 internal bits to 24 bits from its mate once each 700-ns machine cycle. Each matcher has access to any one of 16 different 24-bit internal match groups. During normal in-step operation (routine

Table VI — Central control routine matching

	Quantity Matched			
	Active CC		Standby CC	
Instruction	Matcher 0	Matcher 1	Matcher 0	Matcher 1
Combined load from memory and data adjust	Store address at 8T0	Data reply after processing at 0T4	Internal bus at 4T8— No data*	Internal bus at 8T0— Adjusted data
Combined store into memory and data adjust	Store address at 8T0	Data being Stored at 0T4	Internal bus at 4T8— No data*	Internal bus at 8T0— Adjusted data

* Important data appear on the internal bus at this time for some instructions of these types.

matching mode), the timing of the match and the groups to be matched in each of the four match circuits is determined by the type of instruction being executed and by the active/standby status of the CC. In this manner, the four distinct internal groups which are necessary to assure correct execution of any instruction can be matched in the same machine cycle. For example, for the instruction shown in Fig. 5, the matches would be as shown in Table VI. Routine matching can be specified in one or both CCs when they are running in step. Three other match modes are available for maintenance program usage. In directed matching, the user can specify the match group, match time, and whether the other CC or a constant should be matched. The sampled match mode is used to take a snapshot of one of the match groups; it is controlled by selection of the group match time and by specifying the machine cycle during which the snapshot should be initiated. Utility matching can be used to monitor and match memory operations and to generate a program interrupt at the detection of the match condition specified.

6.4.2 Clock synchronization and checking

The oscillator in the active CC is used as the reference source for all processor timing information. A ring counter generates all the necessary CC clock pulses. There is a maintenance clock that runs continuously and an operational clock that can be stopped in the standby CC and started under program control. Each clock phase can be inhibited, one at a time, by program. The clock output is checked for proper pulse overlap and sequencing. Oscillator level monitors and inter-CC clock synchronization monitors are also provided. A separate analog clock is provided to allow reconfiguration in the event of operational clock failure in the active CC.

6.4.3 Processor configuration

The active CC can generate pulses that control its mate CC and the configuration of all other processor units. This allows the CC to control one of the duplicated pair of memories (or backup units) that will be part of the active processor complex. CC pulses are used for changing system bus configurations during interrupt recovery and for controlling units during fault diagnosis. Internal checks of pulse circuitry are provided to verify correct circuit operation.

System troubles normally are detected by trouble-detection circuits; system fault recovery follows after being initiated by the program interrupt. Since this approach requires a sane processor, an autonomous hardware processor configuration circuit is provided in each CC to handle faults when the processor loses sanity. Normally, only the active CC activates the circuit. Reconfiguration is triggered by loss of operational fault recovery, or configuration program sanity; clock failures; and CC configuration problems. Manual and programmed initiation of the reconfiguration circuit are also provided. The circuit consists of various timers to implement sanity checks on base (operational), interrupt (fault recovery), and configuration level programs. The outputs of the circuit control CC-to-program memory configuration, reloading program memory from disk backup when required, and isolating various subsystems from the CC until program sanity is restored.

6.4.4 System reinitialization

To allow the system to be started from the powered-up state, a facility called system reinitialization (SR) is provided in the CC, which is manually activated from the MCC. When selected, this feature provides a minimal configuration of CC, PS, and a tape unit; it also overrides most error checks. The basic block of program is loaded into PS from tape, and control is then passed to configuration level programs to assemble the remainder of the units on line, load programs and data, and restore a functioning processor.

6.4.5 Diagnostic access and control

Special hardware is provided in all 1A Processor subsystems to allow fault isolation and repair without interfering with normal system operation. A number of circuits are provided to allow the active CC to diagnose the standby without affecting its normal operation capability. Each CC, when active, can initiate many actions in its mate; for example, it can stop and start the operational clock, cause the CC to start and run n cycles, directly set or reset many internal control points, and configure the communication buses. Most registers in the standby CC can be control written and read while the operational clock is stopped. Each sequencer flip-flop can be individually controlled. These features are particularly powerful for isolating many problems, since they do not depend on the ability of the standby CC to execute instructions.

Some CC circuitry is only used when the CC is active. To facilitate fault diagnosis, these circuits can be activated in the standby mode under maintenance program control.

All bus communication circuits can be thoroughly tested since all bus bits can be individually exercised under diagnostic control. Standby CC addressing and data sending buses can be looped back to the active CC through external units.

Each clock phase of the standby can be inhibited to allow testing clock gates and clock error detectors. The matcher circuits, previously described, are used extensively for testing and fault isolation.

6.5 Master control console—processor peripheral interface

6.5.1 Master control console

The primary function of the master control console (MCC) is to provide manual control of various system features and to provide visual displays for administration and maintenance functions. Since it is not critical to system operational capability, it is not duplicated or backed up, but has been designed so that no MCC fault can cause a system outage. Diagnostic program tests are for fault detection and repair of the MCC since interrupt level programs are not required. All display and manual control features, which are interfaced to the system by the processor peripheral interface, are exercised in an interactive manner by the program under operator control.

6.5.2 Processor peripheral interface

Orders from the processor to the PPI are communicated over the peripheral bus, with parity over the address and data used as the error check. There are several PPI features to ensure availability and to isolate problems. Each peripheral bus has a dedicated PPI matrix controller. The matrix row addresses are encoded so that each address is at least Hamming-distance of two from any other. Failure to properly select a row once a PPI name code match has occurred results in an all-seems-well (ASW) failure to the CC. Other ASW components are input data parity and PPI configuration checks that insure against false responses. All of the communication and operational features of the PPI are extensively tested by the diagnostic program.

The interface portion of the PPI, which connects the peripheral bus into the application system, is provided with circuitry to loop around all processor transmit-bus bits into the receive-bus bits. This looping, which is program controlled, provides a powerful maintenance capability for isolating peripheral bus troubles.

6.6 Input/output

The input/output frame is used to provide a teletypewriter data path into and out of the processor for various administrative and maintenance tasks. Since the IO is connected to the CC by the peripheral bus, error detection makes use of such bits as the interleaved transmit parity and answer parity. The input data to each IO selector can be looped back to the reply bus. Internal maintenance circuits provide additional fault detection and isolation within each IO selector. A control pulse is provided for each IOUS to gate back to the CC all the pertinent configuration flop-flop states, routing bits, and error indicators. Internal to the IOUS, checks are made of various operations which, if all checks pass, result in the generation of an ASW response to the CC. The items checked are: data parity, instruction validity, name-code validity, IOUS-to-IOUC bus parity, and IOUC response. Any test failure is recorded in an error-summary register for use by the appropriate fault recovery program. Checks are also made of the synchronization signals received from the CC.

Every IO is polled by CC programs every 60 ms to determine if any work should be processed. Certain IOUS failures can be detected through improper response to the polling pulse. Circuitry is also provided to test each IO controller served by an IO selector. Failure to load or unload the 24-character buffer in the IOUC results in an overflow indication administered by the input/output control program. Carrier failure (if provided for a terminal) is also detected and monitored. Under program control, a signal can be sent to the IO terminal to request an automatic response, which checks out the complete IO loop. The detection of any communication error between a CC and an IOUS will result in an immediate interrupt and entry to a fault-recovery program. In the event of an IO failure, all data is rerouted by program to specifically assigned backup channels on an operational IOUS.

6.7 Communication bus maintenance

All vital communication buses are completely duplicated, and each contains redundant information for checking validity and for data-error detection. Most processor units can generate and check error-detect information although some bits are checked in a round-trip fashion by the original sending unit. When a unit other than the CC detects an error in a bus transmission, it indicates to the CC an all-seems-well failure, which in turn results in a maintenance interrupt.

The error-check schemes on the various buses are designed to handle the expected failure modes of the various units. For instance, call and program store addressing uses a discrete code for order type, a coded unit name enable, and two parity bits over the code and store address. Each of these bits is generated over a partially overlapped, interleaved set of data bits and over the code and address. This overlapping pattern is designed to cope with particular memory circuit failures that have the potential to generate errors in more than one bit of a word. The two parity bits are stored with the data in memory. Data transmission to and from file stores is handled in a similar way.

For the peripheral bus, coded enable operations utilize two parity bits in an interleaved, nonoverlapped manner. CPD enabling has extensive checking identical to that of the No. 1 ESS. The peripheral bus is tested using the PPI loop-around feature without dependence on the application system. The other buses are tested by using the processor units as test generators and receptors. By careful attention to test sequencing and by judicious use of the mate duplicate bus, faults can be detected and isolated.

VII. CONCLUSION

The control system has been designed with the goal of achieving maximum performance consistent with maintenance and cost objectives. A number of challenging design problems were resolved, such as maintaining software compatibility with No. 1 ESS while adding new features, achieving internal CC timing margins with a relatively short cycle and a complex decoding job, providing for compatibility with both 700- and 1400-ns memory cycle times, and meeting environmental requirements. Results of early manufacturing and field experience indicate that the goal is being realized.

VIII. ACKNOWLEDGMENTS

The work described in this article was accomplished through the combined efforts of many designers throughout Bell Laboratories and Western Electric. Without their contributions, too numerous to acknowledge individually, the goals for the control system could not have been met. The authors wish to thank J. T. Mahoney for his help in preparing this article, particularly the section on communication buses.

REFERENCES

1. "No. 1 Electronic Switching System," B.S.T.J., *43,* No. 5 (September 1964), Parts 1 and 2.
2. J. S. Nowak, "No. 1A ESS—A New High Capacity Switching System," International Switching Symposium, Kyoto, Japan, October 1976.
3. A. E. Spencer, Jr., and H. E. Vaughan, "No. 4 ESS—A Full-Fledged Toll Switching Node," International Switching Symposiu, Kyoto, Japan, October 1976.
4. P. W. Bowman, M. R. Dubman, F. M. Goetz, R. F. Kranzmann, E. H. Stredde, and R. J. Watters, "Maintenance Software," B.S.T.J., this issue, pp. 255–287.

Copyright © 1977 American Telephone and Telegraph Company
THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 56, No. 2, February 1977
Printed in U.S.A.

1A Processor:

Memory Systems

By C. F. AULT, J. H. BREWSTER, T. S. GREENWOOD,
R. E. HAGLUND, W. A. READ, and M. W. ROLUND

(Manuscript received July 17, 1976)

The memory hierarchy of the 1A Processor consists of call/program stores, file stores, and an auxiliary data system utilizing magnetic tape. The call/program store is a 65,536-word, 26-bit-per-word memory system using a two-wire, coincident-current, ferrite-core array as the memory medium, and is capable of operating with a 1.4-μs read/write cycle time. The file store (FS) with its disk-file memory provides the 1A Processor with a high-performance bulk-storage system. Each FS controller can control from one to four disk files to provide a maximum storage of 2.56 × 10^6 words. Two FSs make up a file-store community of 5.12 × 10^6 words of duplicated storage. The auxiliary data system (ADS) provides a means for efficiently transferring data between the 1A Processor call- or program-store memories and magnetic tape. The tape medium is used for inputting generic programs, office data, and trouble-location information and for outputting billing, traffic, and error-analysis data. The ADS hardware consists of data unit selectors (DUSs) and tape frames (TFs).

I. INTRODUCTION

1.1 Memory hierarchy

The 1A Processor memories are call stores (CSs), program stores (PSs), file stores (FSs), and tape. The program and call stores are very similar in hardware. The main difference is a two-word or 52-bit output from the PS and a one-word or 26-bit output from the CS.

The PSs contain programs that are normally resident and paging areas that are used for only occasional programs. The CSs are for transient data and for translation data. Each CS containing transient data is duplicated in core while translation CSs are simplexed with the data duplicated in disk. In case of a PS failure, a roving PS spare is substituted and pumped up with program from the file store. Failure of one of a fully duplicated CS pair does not involve a pump up from the FS. Failure of a simplex CS calls for replacement by one of a duplicated CS pair and a pump up from the FS.

The file store is used as a source of data and programs and, in addition, it is used to accumulate data. The file store is essentially nonvolatile; i.e., the contents can be trusted when first powered up. A new unit of memory must be written when first installed. The source can be either the duplicate unit or tape.

The tape units are part of the auxiliary data system (ADS). The tape units are used to accumulate data and to supply program and translation data to the store.

1.2 Memory buses

There are three memory buses: program store, call store, and auxiliary unit (AU).[1] The PS bus services from 2 to 22 stores of 65,536 (26-bit) words. The CS bus services from 2 to 44 stores of 65,536 (26-bit) words. Both of these buses are fully duplicated and operate at multiples of 700 ns. The AU bus services the FS, ADS, and other types of equipment. This bus also operates at multiples of 700 ns. The AU bus communicates with the program stores and call stores via the central control under hardware control. This direct memory access to CS/PS is one of the major differences between the 1A Processor and the No. 1 ESS processor. The file stores are paired, but operate as simplex stores. Each FS can store up to 2,560,000 24-bit words.

There can be two or more pairs of FSs depending on system needs and on other uses of the AU bus. The tape units communicate over the AU bus via the data unit selector (DUS). Both the FS and DUS operate at multiples of 700 ns.

1.3 Cycle times

The 1A Processor is designed to operate with CSs/PSs that operate at either 700-ns or 1400-ns cycle times. The units on the AU bus can transmit data as often as one word every 2.1 μs. The PSs/CSs that are currently in use are magnetic core units operating at 1400 ns. The file stores use disk memories with a transfer rate of 10 μs/word and the tapes are 800 b/in. with a transfer rate of 150 μs/word. Each pair of DUSs can service up to 16 tape units simultaneously.

II. CALL AND PROGRAM STORES

2.1 Introduction

The CS/PS is a 65,536-word, 26-bit-per-word memory system using a two-wire, coincident-current, ferrite-core array as the memory medium, and capable of operating in a 1.4-μs read/write cycle time. As implied in the name, the same basic store unit is used in either the CS or PS communities. When used as a PS, the unit responds to read commands with a two-word (26 bits/word) reply. The differences between call- and program-store operations are determined by minor circuit-pack and bus-cabling options.

When used in a No. 4 ESS, the 1A Processor will contain typically 13 PSs and 12 CSs. A maximum of 22 PSs may be connected to the PS bus, and a maximum of 44 CSs to the CS bus.

The heart of the store design is the 20 A/B memory module. This unit has been in volume manufacture since 1970 for its No. 1 ESS 32K-word call-store application.[2] It is a proven, reliable building block with a relatively low, established cost.

The store will be described in four sections—controller, access circuits, memory module, and power unit.

2.2 Controller

The controller interfaces with the central control (CC) and the memory-access circuits. The controller responds to system requests by determining if the request is for it, deciding what type of order is requested, and generating all the timing pulses and control functions necessary to access the memory circuits and to receive data from or reply data to the CC. The store can perform control or memory type orders. The control orders are used to configure or change the status of the unit, or to return the status to the CC. Additional control orders are generally used by the diagnostic program to verify bus integrity, aid in determining if the unit is working correctly, and, if failing, to resolve faults.

The address format is shown in Table I. The four mode and order bits, R, W, C, and M, determine whether the store will respond via a normal,

Table I — Address format.

Address bit	Function
25	Parity
24–21	Mode & order (R, W, C, M)
20–16	Name
15–0	Memory address

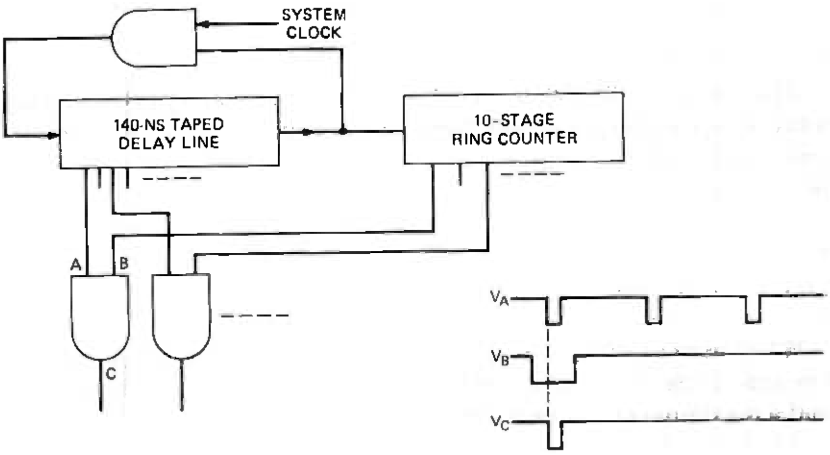


Fig. 1—Clock circuit.

control, or maintenance operation. Each store is assigned a 5-bit name or K-code, which may be changed by a control-write operation. Certain program stores may be set to K-code 20 with a single private pulse. This initializes one store for system recovery.

The store cycle is initiated by receiving a set of sync pulses from the CC. An address window is opened and the internal clock started. If a name match and valid operation (NVOP) is not obtained, the store will shut down, awaiting the next set of syncs. These are repeated every 700 ns. If an NVOP is obtained, the store cycle continues for 1200 ns. The store clock generates precise timing edges with 3-ns resolution over the 1200-ns store cycle. This is realized using a 10-stage ring counter, a 140-ns delay line with 3-ns taps, and logic as shown in Fig. 1. The ring-counter stages define windows during the store cycle. The tapped-delay-line pulses are logically ANDed with the windows to yield precise timing edges for control of store operation.

The controller performs many diagnostic and error-checking functions. A variable-timed strobe, controlled by diagnostic software, is internally generated. It enables the diagnostic to observe the state of critical nodes within the store as a function of time. In addition, incoming address and data are verified for correct parity, the presence of all system clocks is checked, and internal clocks are checked for proper operation. Memory maintenance is also performed by monitoring signals returned from the memory module for level, timing, and double selection. Errors early in the cycle will result in error signals being returned to the CC. These early errors, as well as later errors, are trapped in registers internal to the unit to permit fault resolution.

All control signals to the access circuits are buffered using balanced, differential, emitter-coupled, logic-type drivers in the logic section and

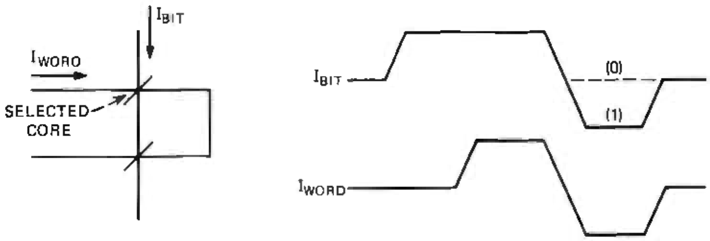


Fig. 2—Bit and word current.

corresponding receivers at the access circuit. This is required to provide adequate noise margin between the logic and the high-voltage (72 V), high-current (1.4 A) access circuits.

2.3 Memory module

The memory module consists of two back-to-back core planes each containing an 832 by 512 array of 23-mil-diameter ferrite cores. Each core has two orthogonal wires passing through it: the bit (or B line) and the word (or A line). The core is switched by coincident currents of 300 mA in each line. During the first part of a read or write memory cycle, the core is read out; i.e., if in a 1 state, switched to a 0. If a 1 is to be written back, the bit and word currents are reversed during the last part of the cycle. The word current is always driven, because it drives more than one bit. During write back, the bit current exists only if a 1 is to be written. As shown in Fig. 2, the bit current is brought up first in the read portion of the cycle. After about 150 ns, the word current is then activated. This is done since the readout signal (the core switching voltage of about 30 mV) is sensed on the bit line. The delay permits the bit line to recover from the high-amplitude drive transient. As also shown in Fig. 2, the word line is looped back through another core on the bit line. This core is selected by reversing the polarities of word current. Since bipolar current is required for one core, the selection of this core is provided only by reversing the timing sequence of the word current. Cancellation of core shuttle noise due to word current also occurs.

2.3.1 B-current selection

For each bit in the memory, a bit line is driven on both the front and back plane. Half of the A loops are on each plane. These two bit lines are driven from a Balun transformer as shown in Fig. 3. The Balun assures the front and back current are each approximately equal to one-half of the input current. The core signal (V_{out}) is sensed across this same transformer. This Balun and the Balun for the adjacent bit are driven from a set of transformer-coupled read and write current sources.

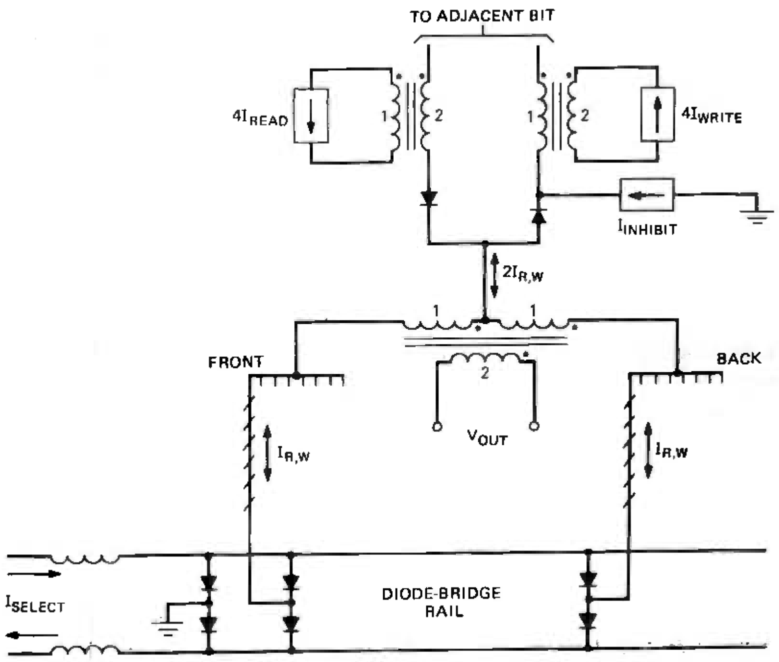


Fig. 3—B-current selection.

Transformer coupling allows all bits to be driven in series, which reduces the number of current drivers. It also ensures that all bit currents are equal. The B-current is inhibited in the memory lines during the write portions when storing a 1 by driving the bit-inhibit current (also transformer coupled), which then provides the write current rather than the bit line.

There are 32 pairs of bit lines (front and back) associated with each bit of a word. Eight pairs are multipled to a given Balun transformer with current-routing circuits selecting one of four such groups of Baluns. The other ends of the bit lines are selected by a balanced multi-armed diode-bridge arrangement, as shown in Fig. 3. By making I_{SELECT} sufficiently larger than the total memory currents, all diodes on the selected rail are forward-biased, providing a virtual ground at the bottom ends of the bit lines. This technique isolates virtually all of the parasitics of the selection circuits, allowing them to be placed remote from the module without adversely affecting the control of the bit-current waveshape. This is important in a two-wire memory since the core switching signal is sensed across the same bit lines that are carrying the high-amplitude bit current.

As with the read and write currents, the rail-select current is also transformer-coupled. Eight sets of rail-selection circuits in conjunction with the four Balun selectors provide the necessary 1-out-of-32-bit pair selection.

2.3.2 Word selection

The word access consists of 512 loops. A one-of-eight select is performed on the nondriven end of the word loop using the same type of bridge-rail circuit as described in the B-current section. Sixty-four drive transformers arranged in an 8-by-8 matrix complete the selection. Memory current is provided on one of eight inputs on one axis of the matrix (called verticals). A one-of-eight selection is performed on the other axis using the bridge-rail circuit (called horizontals). (See Fig. 4.) As shown in Fig. 4, a transformer is located in each vertical to provide a sample of the A current to the maintenance and readout strobing circuits.

2.4 Controller to memory module access

2.4.1 Bit-current selection

All active circuits used to access the core are contained external to the core memory module to meet the reliability and repair time requirements for the store. The memory module contains transformers, diodes, and resistors. The basic drive element used to drive the memory

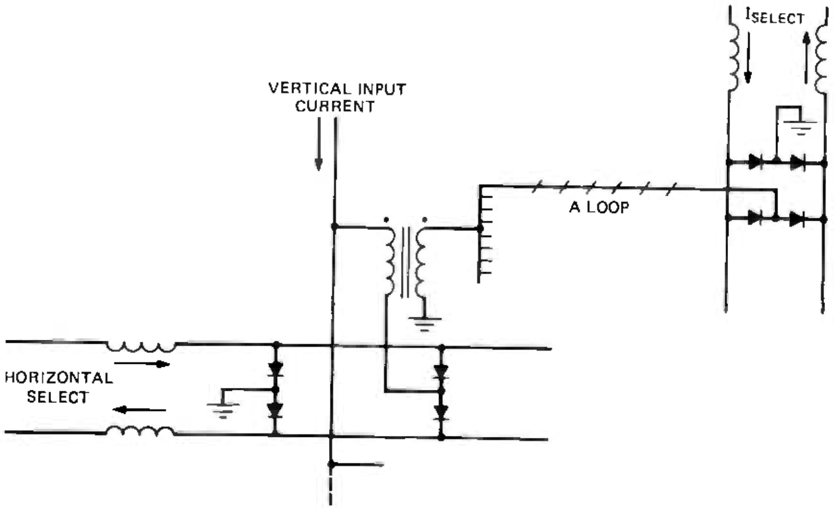


Fig. 4—Word line selection.

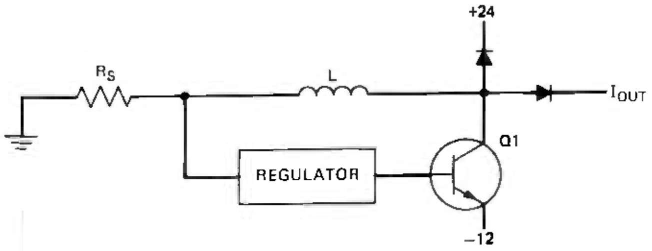


Fig. 5—Current driver.

module is a regulated current driver whose current level is controlled by a reference voltage (VREF). VREF is a memory module output voltage and is determined by the temperature of the core cover plate internal to the module. This is necessary for the drive current to track the core characteristics as a function of temperature. To drive the high-inductance load and obtain the necessary rise and fall times of the memory current, drive voltages of up to 65 volts are required. To reduce power dissipation in the driver, a switching regulator is used to control the current in a large (1-mH) inductor. This large inductor then can drive the inductive load of the memory (approximately 4 μH) and maintain good current control. The basic regulator is shown in Fig. 5, where the inductor is switched to −12 V or clamped at +24 V to control the current. When driving the memory, regulator action is inhibited by turning off Q1.

The output of the drivers must be switched to the desired module inputs. The basic floating high-current switch that performs this function is shown in Fig. 6. This circuit must switch 1.4 A with a 65-V driver level. Switching speeds in the tens of nanoseconds are necessary to meet the system timing requirements. Switch transitions must be fast (<15 ns)

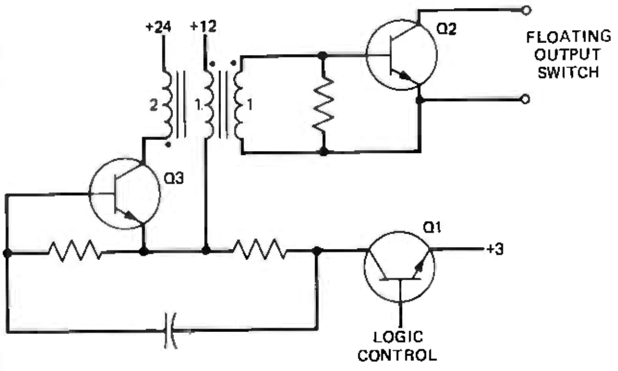


Fig. 6—High-current switch.

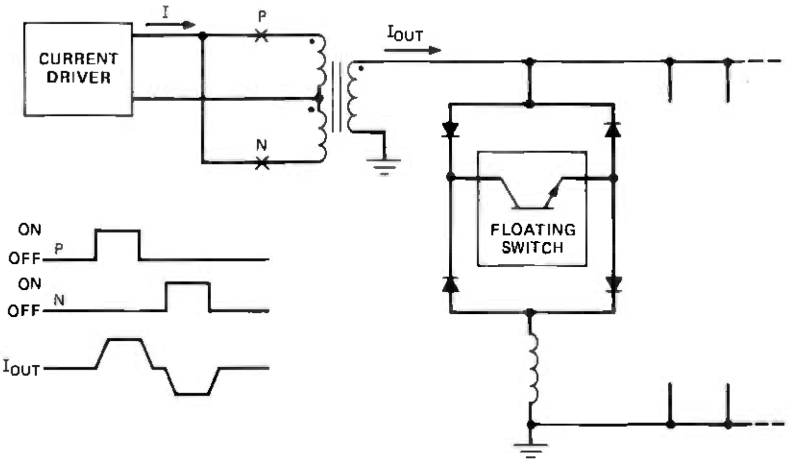


Fig. 7—Vertical current.

to minimize power dissipation. The circuit of Fig. 6 employs a high-speed turn-off of Q2 using Q3. This permits Q2 to be saturated to reduce power dissipation.

To perform the rail selection described in 2.3.1, it is necessary to drive one-of-eight B-rail drive strings. This is accomplished using a single current driver, which is routed into the selected transformer string via a set of eight of the floating, high-current switches described previously.

The one-out-of-four B-read and B-write drive strings are driven in a similar manner. In this case, one current driver is used for both the read and write current. A series switch determines whether read or write current will be driven and isolates the current driver. Selection switches then select one of four drive strings during the B-read and B-write portion of the cycle. Although the memory bit-line currents are on the order of 300 mA, the various transformers used in the selection and routing process provide an effective 4-to-1 ratio from current driver to memory. Hence, the bit-access switches handle currents of the order of 1.2 A.

2.4.2 Word-current selection

Except for lower current, word or A-current selection is similar to the B access described previously. In this case, however, bipolar current is necessary to drive the A vertical inputs. As shown in Fig. 7, a transformer and two switches are used to generate the bipolar current from a unipolar current driver. The operation of switch P results in positive current, whereas N results in negative current since the primary is reversed. The selection of 1-out-of-8 switches then completes the A vertical selection. These switches are diode-bridge switches to handle the bipolar current.

The total 1-of-1024 A-word line selection is achieved via 1-of-8 bridge rails, 1-of-8 horizontal rails, 1-of-8 vertical drives, and a 1-of-2 A vertical current polarity.

2.4.3 Signal detection

As described in the module section, the core signal appears across the secondary of the driven B-current Balun. As shown in Fig. 8, two secondaries are connected in series to reduce the number of detectors required without significantly loading the core signal. The resulting two pairs of leads are applied to the data channel. The input stage then selects one of the two pairs. The pair selected is determined by which B current was driven. The core signal rides on top of a significant offset voltage caused by the diode-bridge uncertainties and variations in resistive voltage drops on the driven and nondriven B lines in the memory. In addition, during the bit-current rise time, large voltages due to inductive unbalance of the B line and partial switching of nonselected cores will exist on the readout line. To establish a ground reference for de-

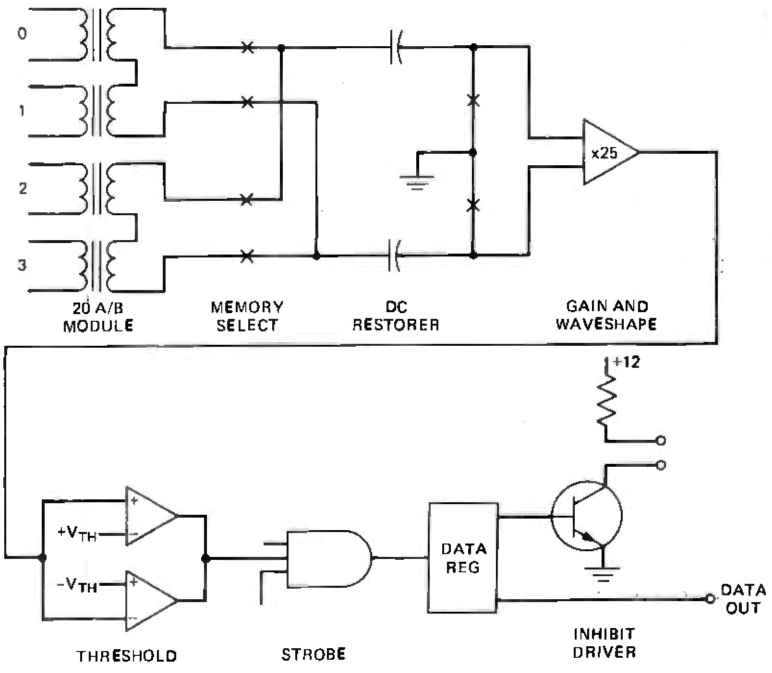


Fig. 8—Data channel.

tecting the core signal, the offset voltage is stored on the restorer capacitors. Just prior to the start of the core switching voltage, the switch holding one side of the capacitor to ground opens and allows the following signal to be amplified. The memory signals are balanced up to the amplifier. Therefore, two restorer switches and capacitors are required. The output of the amplifier then drives two detectors whose outputs are ORed. One detects positive 1s and the other negative 1s. The detector output is then strobed into a data register. The strobe is timed a fixed delay from the 50-percent point of the A current to match the peak of the core signal. The A-current timing is obtained from the A-current sample leads of the 20 A/B memory. The data register then controls the inhibit driver, which when active (data = 0) inhibits the write back of a 1.

A test mode is available where the detector threshold level may be raised or lowered to detect marginal operating conditions.

2.5 Power

In addition to the −48 V system power, the store uses five regulated voltages. These are supplied by a power unit, which also provides high-voltage protection, current limiting, voltage monitors, and various diagnostic test features.

III. FILE STORE—A DISK MEMORY SYSTEM (FIG. 9)

3.1 Introduction

In the No. 1 ESS processor, all programs are stored in the permanent magnet twistor (PMT). The cost of storing a program or data in the PMT is the same whether it is vital or ancillary. Changes are also slow, requiring rewritten magnet cards, which are manually changed. To reduce cost and provide flexibility for changes, the objective in the 1A Processor was to have an electrically alterable memory that would hold data during power loss or hardware faults. A disk admirably meets these objectives. The cost is low and the information resides in the magnetic state of the disk medium and is not easily destroyed. The disk, however, is sometimes too slow to operate out of directly. Therefore, it is used instead as a source of programs, and data, which are moved to the faster random-access stores before being used.

The combination of disk and random-access memory (RAM) is less expensive than the equivalent PMT and is much more flexible. One reason for the low cost is that much more is stored on disk than is in RAM at any instant of time. For instance, most diagnostic programs are brought to RAM as needed and the same RAM space is shared among different programs.

The cost of a bit stored on disk is significantly less than a bit stored in RAM. If the performance of a disk memory (its delay in returning

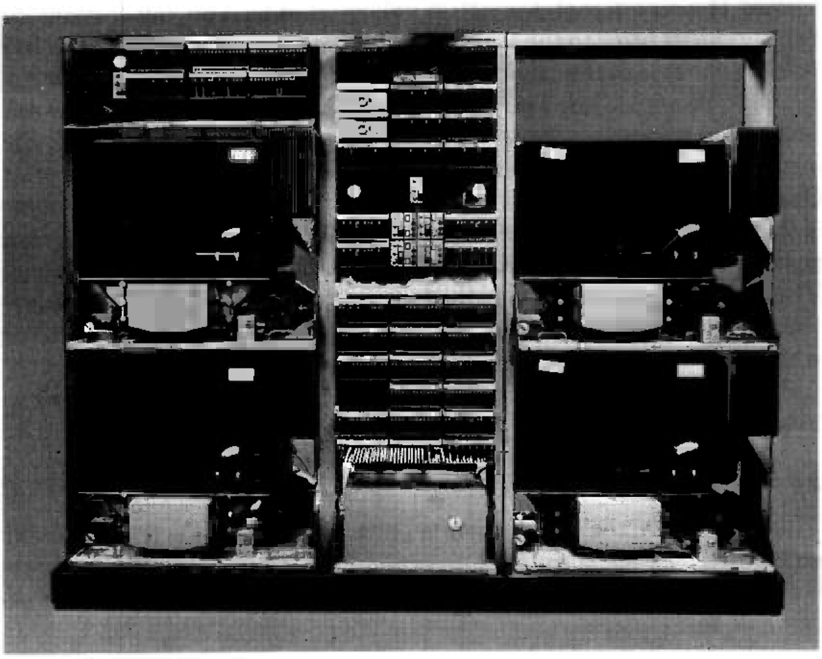


Fig. 9—File-store frame.

needed data) is good enough for a given application, then that class of data will not have to be stored in RAM. If the delay is too long, then the data must be stored in RAM and on disk.

A major objective in the design of the file store was high performance. High performance means low latency, i.e., data is returned with the minimum delay. If the system were to request one piece of data at a time, not much could be done in hardware to reduce latency. Each request on the average would have to wait a half revolution of the disk.

If a group of requests is available, significant improvements in performance can be made. It is possible for the FS to handle 50 separate requests (jobs) in one revolution, giving a delay of less than 1 ms between jobs.

The disk memories are reliable and nonvolatile, but they and the controller can fail. To assure a continuous source of data and program, FSs are duplicated. A community of FSs has two controllers each with from one to four disks. The FS system is designed to operate with disk failures in each controller as long as duplicate pairs of disks have a maximum of one failure. Each controller is on a separate AU bus.

The two controllers in a community communicate the phase relationship of duplicate disk pairs. This information is used to maintain a 180-degree phase relationship. This has two purposes. A job can be submitted to both controllers simultaneously. The first disk to reach the address does the job. The 180-degree phase relationship assures that the job will be completed before the other can start. Completion cancels the job for the other disk. This protection is needed to prevent the latter controller from overwriting a data set which has just been modified. There is another obvious advantage to the 180-degree relationship. It reduces latency by assuring that one of the pair of disks is always within a half rotation of any address.

3.2 Disk-file memory and support circuitry

The commercial disk chosen for this application meets the main requirements of reliability and availability. The characteristics are:

1800 RPM
16×10^6 bits (gross storage)
200 heads
2.4-MHz data rate
Plated media
Continuous air filtration.

The memory units are organized as follows:

100 tracks/face
2 faces/disk
101 sectors
34 words/sector (32 data, 1 preamble, 1 error code)
24 bits/word.

One hundred sectors are used for data and the last sector in each track is used for diagnostic testing.

All power must be derived from either +24 dc or −48 dc. The disk motor is a single-phase, 208-V induction motor with a capacitor start winding. The motor is powered by an inverter, which is part of the 180-degree servo loop. The frequency of the oscillator controlling the inverter is continuously changed to control the 180-degree phase relationship.

The motor continuously accelerates and decelerates the disk; i.e., supplies or accepts energy. When accepting energy, the voltage across the silicon-controlled rectifier (SCR) in the inverter that is being turned on can be zero. This will delay the actual start of conduction. Two things were done to assure reliable commutation when accepting energy and when the start winding kicks out. The inverter output transformer is clamped with diodes. The effect of this is to reduce the reverse voltage across an inductor in the emitter circuit of the SCRs. This low reverse voltage causes a large circulating current to flow. This current tends to offset the effects of the reverse flow of energy from the motor. Commutation is assured by supplying many trigger pulses to the SCRs. Each trigger has a fast rise time to force the SCRs into conduction rapidly and the pulse is repeated, so that conduction will start reliably whenever the voltage across the SCR becomes positive.

The FS must operate over an appreciable temperature range and it is desirable to minimize adjustments to circuit packs, particularly in the field. To meet these objectives, both timing and gain were made adaptive. No adjustments are required when a disk or a circuit pack is replaced.

The timing circuit was designed with phase adaption in mind. The disk has two clock tracks, bit and sector. All active timing is derived from a phase-lock-loop locked to the bit track. This circuit consists of an oscillator many times the frequency of the bit clock. The oscillator output is counted down to the frequency of the bit clock. The phase of the two are compared and the frequency of the oscillator adjusted to maintain phase lock. One advantage of this approach is that accurate timing is available at periods much shorter than a 1-bit clock cycle. Another advantage is that minor defects in the bit clock do not affect the accuracy of the derived timing. Failure is detected by matching the derived count with the count stored in the sector clock.

Writing on the disk is timed directly from the timing circuitry. When a sector is written, a preamble is written before any data. This preamble is used to first establish a decision level for the read circuitry and then to establish accurate strobe timing. This accurate strobe timing is derived by forcing a counter (Fig. 10) not to count down the oscillator frequency. This causes the preamble to be shifted into a register at a very high rate. When the appropriate pattern (preamble) is recognized in the register, the counter is allowed to operate normally. The next output of the counter will coincide with the center of the first bit of data.

The 180-degree servo loop between duplicated disk files does not require high accuracy, but was difficult to design for several reasons. For economic reasons, it is desirable to minimize the exchanging of position data between communities. The minimum possible is used, one bit per disk. The motor is single phase, and it produces a marginal excess torque at low battery. The motor is coupled to the disk via a belt. Large torque changes at high battery had to be avoided for good belt and motor-mount lifetime.

The servo is a master-slave type capable of operating in either mode. The disk that is the master controls its speed and the disk that is the slave maintains phase lock to the master. If the slave disk detects an out-of-speed indication, it switches to master mode. If a disk is out of speed for three revolutions, the heads are retracted. This three-revolu-

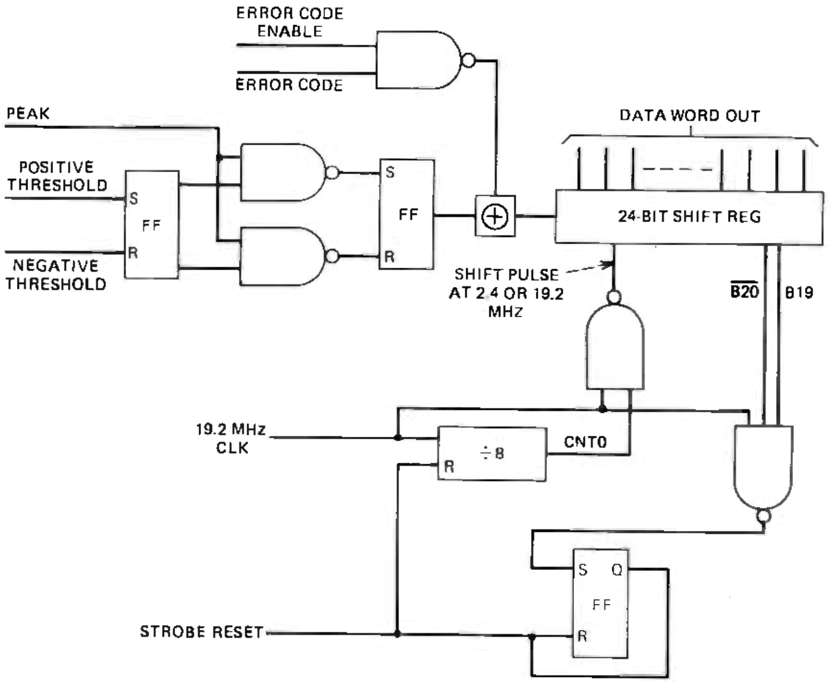


Fig. 10—Adaptive strobe circuit.

tion delay is sufficient to allow the slave-to-master transition to return to the correct speed without retracting heads.

The servo is a digital circuit that minimizes torque excesses and acquires and holds phase lock under all possible conditions:

(*i*) One or two flip-flops are set, depending on the magnitude of the phase error, if the error is increasing.
(*ii*) The phase polarity and the two FFs drive a D-to-A converter with three states increasing and three decreasing the frequency.
(*iii*) The rate of decreasing phase error is made small so that little overshoot occurs, and for all small errors the restoral force is the least output of the converter (making for low torque changes while phase locked).

The servo will acquire the 180-degree phase lock in less than one minute after the heads fly. It will hold lock from 41.75 V to 52 V. There is no measurable increase in wear on the motor or belt due to the servo.

3.3 Job setup and execution

Because the disk revolution rate is extremely long compared to the system cycle time, requests for disk jobs are queued in the FS. Because of cost considerations, only the minimum amount of information necessary to find a disk job is stored in each register of the job queue. Corresponding to each queue register is a disk request (DR) block in core memory where the job is fully specified. Administering the queue and the DR blocks is a disk-administration program.

Clients submit jobs to the administration program, where an idle queue register in the mate FS is simultaneously loaded. An active FS that is not doing a disk job is continuously searching the queue for work. Once it detects a job that starts within the next one or two sectors, searching stops and the FS autonomously accesses core to retrieve the DR block.

Disk-read jobs, which are no longer than 1024 words, are sent to both FSs simultaneously. Given that corresponding disk files in the duplicate FS are servoed 180 degrees out of phase and the fact that a 1024-word job is only one-third of a revolution in length, one FS can do the job and return status to the DR block before the duplicate FS accesses the DR block. When the duplicate FS reads the status word and finds the read job has been done, it cancels the job in its queue and restarts the search sequencer. A disk-write job is not considered done until both FSs have done it successfully.

When the status word in the DR block is retrieved during job setup, it is checked using the above rules to determine if the job should be done. If it should be done, job setup (retrieving the remaining words in the DR block) continues. To do the disk job, core store is again autonomously accessed to obtain data to write on disk or to write core with data read from disk. At the conclusion of the job, the FS writes job status in the status word of the DR block, idles the job queue register, and starts searching again.

It is the disk-administration program function to notify clients when their job is completed and whether it was done successfully or not.

3.4 Auxiliary unit bus interaction

The FS communicates with CS/PS via the CC and the auxiliary unit (AU) bus. Each AU utilizes a handshaking routine with CC to perform information transfers to and from CS/PS.

An AU first requests use of the bus by sending a bus request. When the request is granted, the AU acknowledges receiving the use of the bus with a verify signal. The unit then sends address and data (if a write to CS/PS) and waits for an acknowledgment that the transfer has been completed. After acknowledgment, the FS returns an acknowledgment-verification signal to the AU sequencer in CC.

AU bus activity is controlled by CC according to a priority structure and to a CC "give up" algorithm. The priority structure gives CC the highest priority followed by FS 0, FS 1, FS 2, FS 3, and then the other AUs.

The priorities for FSs are programmable within the allocated priority groups, while those of other AUs are hardwired. The "give up" algorithm basically says that CC must give up 1-out-of-4 cycles to AUs.

If bus blockage occurs, the FS will continue to request the bus every 700 ns until the request is granted or the FS buffers overflow and the job is resubmitted under program control starting at the point of overflow.

CC "give up" of the AU bus when continued blockage occurs is based on the following criteria:

(*i*) If AUs have been blocked three times, and if during the blockage no AU successfully completed a store operation, CC will give up the next available cycle to an AU.

(*ii*) If an AU to CS/PS operation is in progress, and the CC wants to read or write an AU, the CC inhibits enables to all AUs and waits for the operation to complete before accessing the AU. Enables are inhibited for a maximum of three successive CC cycles.

3.5 Controller organizations

The file-store controller is a 25,000-gate machine with 22 functional circuit areas. The seven major circuit areas are described below.

3.5.1 Disk-request registers

These registers are the hardware job queue for the FS. The queue is made up of either 24 or 48 registers, depending on system requirements. The registers are 13 bits wide and contain only a subset of the information needed to process the transfer to or from disk. This information includes disk sector address, disk file, disk face, read/write bit, an update bit, and the busy/idle bit.

3.5.2 Address translator

The address translator translates a 17-bit record address into two 7-bit fields (sector and track) plus a 3-bit field (disk and face) and allows the system to view the FS memory as having a closed binary address field. The translator converts that binary address to provide the addressing for a disk with 100 tracks, 100 sectors/track, and 32 words/sector.

3.5.3 Search-and-match sequencer

This sequencer searches through all the job requests in the queue until it finds one that is for a sector close to the present position of the read/write heads. Once a match has occurred, the job is passed to the store sequencer for processing. After the store sequencer has processed the job, a signal is returned to the search sequencer. The signal then sets the idle bit in the job queue and starts searching for the next job.

3.5.4 *Job-control registers*

These registers contain the detailed job-control information. The four registers are loaded from the DR block and contain the following information: number of words, identification tag, record address, starting word in sector, core-store starting address, plus status information from the other FS in the community if the job has been set to both FSs.

3.5.5 *Store sequencer*

This sequencer manages information transfers to and from core stores. In addition, it processes all errors detected by the FS. Depending on the nature of the error, the sequencer will either stop the FS, take it off-line and notify the system, or it will report job failure status to the system via the status word in the DR block and then start the search sequencer.

3.5.6 *File-store timing*

The FS operates on two timing systems. The controller operates on timing sychronized with the CC while the disks operate on disk timing. Information transfers cross this timing interface asynchronously. Transfers from the disks are accepted asynchronously at the timing interface and synchronized with CC timing in the controller. Information transfers to disk are placed in a buffer and moved asynchronously via disk timing to the disk.

3.5.6.1 *Controller timing.* The system-related timing circuit in the controller generates all the system sychronized timing pulses needed in the FS. Fourteen 100-ns and fourteen 50-ns timing pulses are generated for normal-mode timing. This timing chain is duplicated, and the duplicate chain is used to drive FS maintenance circuitry. The use of a mask circuit to inhibit normal-mode clock pulses for diagnostic testing makes the duplicate unmaskable chain a necessity.

CC sends a sychronization pulse to the FS every 700 ns where it is used to start a 20-MHz gated oscillator. This oscillator drives the clock generator, decoder, and fanout circuits. Near the end of the 700-ns cycle, the gated oscillator control is reset. The oscillator then waits for another synchronization pulse before generating another string of timing pulses. The oscillator has been designed to hold its operating tolerance over a 40-year lifetime.

3.5.6.2 *Decision time sequencer—disk timing.* Reading or writing a disk is done with disk timing, which is independent of processor timing. Each disk has its own decision time generator consisting of a voltage-controlled oscillator. It is phase locked to the bit clock track on the disk, a phase-locked counter, and associated decoding circuitry to provide disk timing pulses.

3.5.7 Automatic exerciser

The automatic wired logic exerciser reads the disk file when the FS is not processing a disk read or write. Thus, the integrity of the recorded data is routinely verified without system intervention. Read errors are counted until an overflow of the error counter (modulo 64) occurs. On overflow, the system is notified by the FS and the FS then stops in the maintenance mode. The last failing disk address is stored in the exerciser register.

3.6 Achieving system design objectives

The following sections enumerate FS design objectives and how each objective was met.

3.6.1 Performance requirements—high throughput, low latency

Performance requirements for the FSs were set by the worst-case throughput required in a No. 4 ESS. This required the FSs to complete 450 requests per second with an average delay of 30 ms per request. The 30 ms represents the elapsed time from job insertion in the hardware queue until job completion. It does not include task-dispenser time used to inform the requesting program that the transfer has been completed.

To achieve high throughput and low latency, the FS performs a closest-sector-address match on jobs in the hardware queue. Thus, the job closest to the present position of the read/write heads is always done first. The job queue is searched for jobs at the rate of one register every 700 ns. Thus, a 48-register queue is searched nine times during each disk sector.

Latency is reduced by a factor of two for read jobs which are no larger than 1024 words by the 180-degree phase relationship of duplicate disks. These jobs are submitted to both FSs and the first FS to process the job has time to complete and return status to the DR block before the second FS acquires the job.

Throughput requirements were met with one community of FSs operational. With only one controller in operation, the requirements were met by limiting low-priority jobs and restricting the number of large jobs in the queue.

Simulation results at 450 requests/second show 2 FSs have an average message delay of 19 to 33 ms (depending on average job size) while four FSs exhibit delays of 12 to 14 ms. Stress testing of the FSs show them meeting system requirements at 600 requests/second.

3.6.2 Reliability

Since the 1A Processor system is expected to provide essentially continuous on-line service, the file store, as part of that system, must be highly reliable.

File stores are duplicated for reliability, but mates must be able to operate independently, thus running in step and matching cannot be used as the error-detecting scheme. As a result, the FS must be self-checking. Many self-checking circuits are provided to assure proper operation, particularly to minimize the probability of mutilating CS/PS data.

One important circuit area that is checked is the registers containing the job-control information, the DR block. These work registers are duplicated and as each word is received, it is gated to duplicate registers over different buses at different times. The contents of the registers are then matched to verify that words arrived in the work registers.

The record address loaded in the job queue is compared with the record address in the DR block to ensure that the correct job was accessed. Since this address has to be translated both times it is received, this procedure also checks the translator.

Addresses generated by the FS to access CS/PS are checked within the FS. Each address sent to the output-address register is returned to the copy in the work registers and matched. This checks the bus path within the FS. Also, the duplicate copies of the CS/PS address are independently incremented and then matched to determine that the addresses were correctly incremented.

Data is protected in the FS by parity (when it is moving over an internal bus) or by a cyclic redundancy code (when on disk). The cyclic code also contains address information and thus provides a check on track-selection failures.

System reliability requirements for the FSs have been demonstrated. With an excess of 500 thousand hours on the disk files, they are exhibiting a meantime-before-failure of four years.

3.6.3 Low system overhead

Low system overhead is achieved by allowing the FS to autonomously process jobs. Once a job has been loaded in the hardware queue, the FS processes the job without system intervention. Following a sector-address match, the FS accesses CS to obtain the complete job-control information. Direct memory access then permits data transfers with a minimum of data-processing cycles stolen from CC. Only when a job status word has been returned to CS by the FS is system action again required.

3.6.4 Diagnosability

The objective of the FS diagnostic program is to detect at least 95 percent of all faults and to resolve these to either the controller or a disk file. For the case of controller faults, fault resolution to two replaceable circuit packs on the average is the objective.

To meet these objectives, diagnostic-test design was done in parallel with the hardware design. This resulted in the addition of maintenance-access circuits that are necessary for testing the error-detection circuits as well as for testing basic operations. Maintenance access includes 300 nodes that can be read to check circuit operation plus 600 control nodes that can be activated to test FS circuitry. Maintenance circuitry plus error-detection circuitry account for over 30 percent of the FS circuitry.

3.6.5 Equipage

The FSs are required to be growable to meet the memory requirements of a range of systems. The FSs can grow from a minimum equipage of one disk (duplicated) to a maximum system composed of two communities of FSs with four disks in each file store. The typical No. 4 ESS will have one community with four disks per FS and the typical No. 1A ESS will have one community with three disks per FS.

IV. AUXILIARY DATA SYSTEM

The auxiliary data system (ADS) provides the 1A Processor with a flexible and efficient means of transferring data from call- or program-store memory to magnetic tape or from magnetic tape to memory. The ADS structure permits other data input or output devices to be attached to the 1A Processor in a manner similar to that of the magnetic tape.

The initial uses of the auxiliary data system are:

(*i*) Automatic message accounting—The recording of call data for subsequent processing for billing purposes.

(*ii*) System reinitialization and system update—System reinitialization is required for initial loading of generic programs and office data into CS/PS memories. Subsequent system reinitialization could be required if data multilation in those memories should occur simultaneously with a duplex failure of the backup information in the file-store memories. Because of its severity, system reinitialization (SR) can only be initiated manually.

System update, on the other hand, is used to load new generic program or office data into one copy of the backup memory while the system continues to operate on the original program and data. After the system

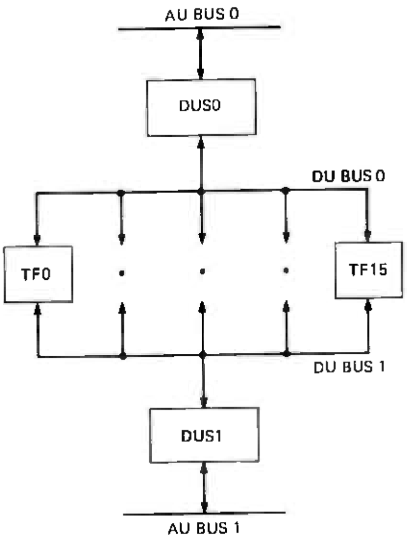


Fig. 11—ADS connection to AU bus.

update is complete and verified, it is loaded into the appropriate areas of active call and program-store memory.

(*iii*) Trouble location—Data bases used for generating lists of suspected circuit packs corresponding to diagnostic failure patterns are stored on magnetic tape.

(*iv*) General use—A number of uses of the tape system are made in addition to the above. These involve outputting of operational or maintenance data from call- or file-store memory, loading special nongeneric programs, verifying the resident copy of generic program and office data, and writing new generic and office-data tapes.

4.1 ADS organization

As indicated in Fig. 11, the ADS consists of data unit selectors (DUS) and tape frames (TF). The 1A Processor can have up to two pairs of DUSs. The even numbered DUS of the pair is connected to AU bus 0 and the odd-numbered DUS is connected to AU bus 1. Each DUS has access, via the data unit (DU) bus, to up to 16 TFs. Both DUSs of a pair access the same TFs. Several TFs can have data-transfer jobs in progress simultaneously since the DUS serves to multiplex their access to memory. The maximum data-transfer rate through a DUS depends upon blockage that can occur due to memory conflicts with higher-priority AU bus users. In

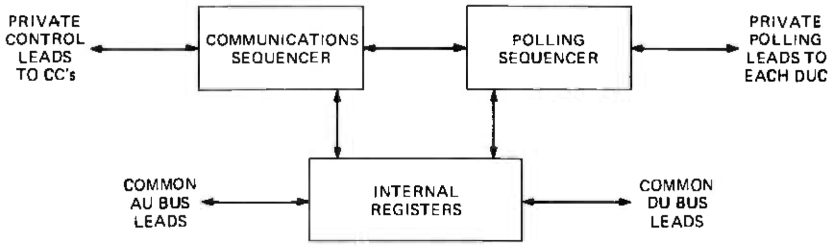


Fig. 12—Data-unit selector organization.

the absence of blockage, DUS throughput can exceed 2×10^6 bits/second.

The organization of the DUS is shown in Fig. 12. The AU bus and DU bus communications are under the control of sequencing logic that gates address, data, and control information from the AU or DU bus to internal DUS registers and vice versa. Polling control is also shown in Fig. 11. The polling function is normally performed by the DUS if it is not in the process of relaying data between memory and a TF. By administering the polling-control logic, the 1A Processor programs can cause the DUS to poll all, none, or any subset of the equipped TFs. The TFs response to the poll will indicate whether a data transfer is being requested (read or write) or if other attention is required. Other attention might be of a normal operational nature; for example, as a result of a successfully completed job, or of a maintenance nature, such as a result of a failure of some operation in the TF. If the poll response indicates a read or write request, the DUS will request access to core memory through the central control. When the central control permits access, the DUS initiates the memory transaction. This is accomplished without program intervention in the central control and normally does not require stealing cycles from program execution. If the poll response indicates a need for operational or maintenance attention, the DUS will cause an operational or maintenance-request flag to be set within central control so that appropriate programs will take action.

The TF contains a tape unit controller (TUC) and a tape transport. The TUC contains circuits for sequencing commands, checking for errors, and maintaining an orderly data flow to the transport.

The salient operational features of the TUC are depicted in Fig. 13. The address register specifies the memory address that the TUC is to access. The character count register contains the count of the number of 8-bit characters remaining to be transferred during a read or write job. The add 1 logic is used to increment the address register each time a word is transferred to or from memory. It is also used to decrement the character count each time a character is transferred to or from the tape unit.

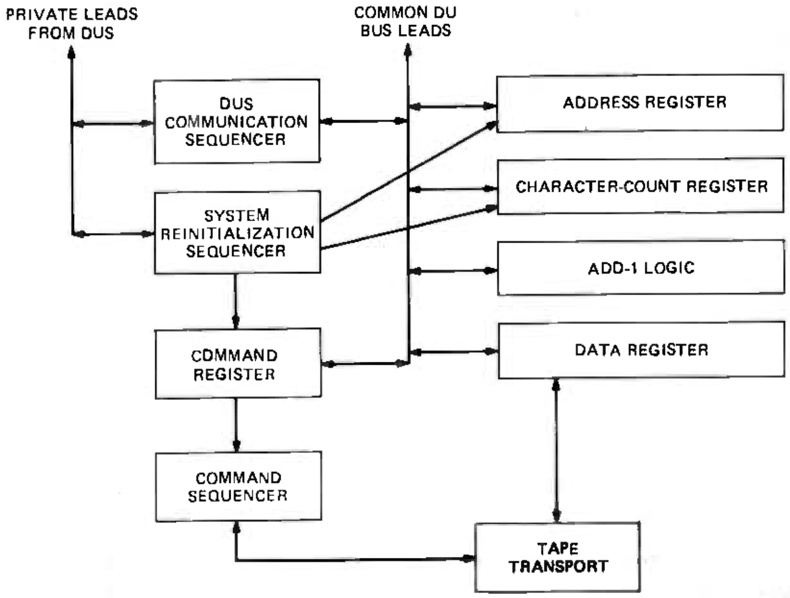


Fig. 13—Tape-unit controller organization.

The command register specifies the job such as read, write, backspace, etc., that the TUC is to perform. The data register serves as a buffer for data en route between the tape unit and the DUS. The TUC-to-tape-transport communications are in units of eight data bits plus parity while communications with the DUS are in units of 24 bits plus parity. Thus, the data register provides a 3-to-1 mapping of characters to words and vice versa.

When the system program initiates a read or write job in TUC, it writes the starting memory address into the address register, the job size into the character-count register, and the command code into the command register. The TUC will then process the job under control of the command and communication sequencers. Successful job completion is indicated by returning an operational interject response to the DUS poll. The interject response causes the DUS to set an interject source bit in the central control, which notifies the program that attention is required.

As mentioned previously, system reinitialization is a function of the ADS. An SR request causes the SR sequencer in the selected TUC to initialize the address, character count, and command registers in a manner that will cause a bootstrap program to be loaded into memory. Once the bootstrap program is loaded, it controls the loading of subsequent blocks of program and call-store memory.

The tape transport is a 9-track, 800-character-per-inch unit, which reads and writes at a speed of 25 in./s. The data rate (excluding parity) is 160 kb/s. Fast forward and rewind speeds are 100 in./s. The tape unit is powered from a converter that enables it to operate from −48 V office battery.

4.2 Trouble-detection-and-reporting techniques

The ADS incorporates a number of techniques to detect errors. These include parity over data and addresses, sequencer- and decoder-validity checks, echo checks on select and data signals between DUSs and TUCs, and implementation of the standard tape error-detection codes and read-after-write data checks. One of the most important error-detection features is incorporated in the add-1 circuit since incrementing failures could cause incorrect address generation and result in multilation of call or program memory. The add-1 circuit is self-checking and can detect any classical fault (gate input open, output stuck at 1 or 0) that would affect the outcome of the add-1 operation. When an error is detected, the ADS will suspend further accesses of memory and notify the program either via the operational-interject response or by the maintenance-interject response that is also used by the file store for trouble reporting. The program will then test the unit reporting the error and possibly remove it from service.

V. CONCLUSION

The 1A Processor memory systems herein described have demonstrated their reliability and their ability to meet performance goals. Performance has been verified by the systems now in operation; five systems are operating at Bell Laboratories, Indian Hill, Illinois and five No. 4 ESSs with their 1A Processors have been placed in commercial service.

VI. ACKNOWLEDGMENTS

This report is based on the work of many people in Bell Laboratories. All store designs were achieved by close cooperation between store-design groups and the diagnostic, fault-recognition, and system-design groups.

REFERENCES

1. A. H. Budlong et al., "Control System," B.S.T.J., this issue, pp. 135–179.
2. J. G. Chevalier and M. W. Rolund, "New Memory Reduces No. 1 ESS Cost and Size," Bell Laboratories Record, *50*, No. 4 (April 1972), p. 121.

Copyright © 1977 American Telephone and Telegraph Company
THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 56, No. 2, February 1977
Printed in U.S.A.

1A Processor:

Technology and Physical Design

By J. O. BECKER, J. G. CHEVALIER, R. K. EISENHART,
J. H. FORSTER, A. W. FULTON, and W. L. HARROD

(Manuscript received August 5, 1976)

The physical design of large electronic switching systems (ESSs) requires a comprehensive set of technologies and design tools. The technology used for the 1A Processor and several ESSs consists of silicon integrated circuits, thin-film hybrid circuits, discrete component packaging, and apparatus to interconnect, power, and communicate with these devices. The application tools consist of comprehensive design guidelines and computer aids for circuit, logic, thermal, and physical design. In addition, computer aids are provided for documentation and preparation of information for manufacturing and testing. This paper uses text and illustrations to describe the design details and achieved performance of each of the major elements of the technology.

I. INTRODUCTION

1A technology is a standard set of devices, apparatus, and design tools which are used not only to design the 1A Processor and No. 4 ESS system, but also new switching systems such as No. 3 ESS[1] and additions to present systems such as the remreed network for No. 1 ESS.[2] It has been made a broad and flexible technology applicable to an entire new generation of telephone switching equipment.

The hardware design requirements of these "new-generation" switching systems differ in many important ways from the switching hardware typified in the No. 1 ESS system.[3] The advent of silicon and hybrid integrated circuits in the mid-1960s provided an opportunity for dramatic miniaturization in hardware. The first-level package in these new switching systems is often a silicon integrated-circuit chip about 50 thousandths of an inch square. Such a chip often contains more circuit components than a 4- by 7-inch plug-in package in previous systems. The influence of miniaturization is characterized at every level of packaging by the reduced size and increased density of required interconnections.

Interconnections are also affected by another continuing trend, that of upgrading the performance of new switching equipment. Increased machine speeds and faster pulse rise times are required to provide more throughput and improved service. As the speed increases, so does the sensitivity to noise and transmission losses. As a consequence, stringent electrical requirements are imposed on the interconnections in faster machines such as the 1A Processor. Careful attention must be paid to the design of the interconnection system to insure that it meets all requirements.

An additional factor of importance in interconnections is cost. The conventional design, assembly, and wiring techniques used in previous switching systems have a relatively high labor content per interconnection. Rising labor costs and the need for more numerous and better controlled interconnections have been strong driving forces in the development of a hardware system with low labor sensitivity.

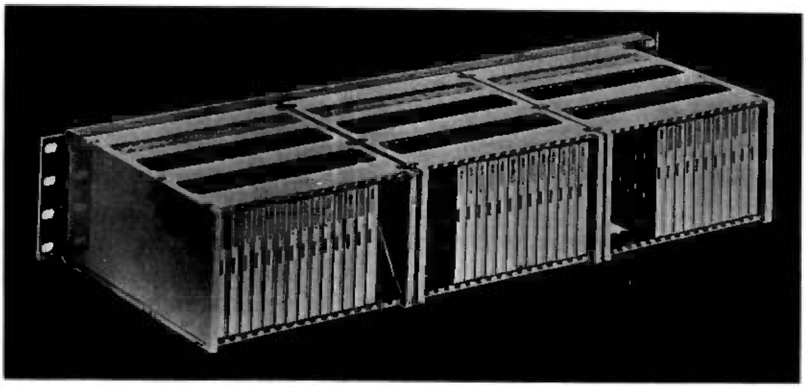
Three factors shaped the overall philosophy of 1A technology: miniaturization, high-speed electrical performance, and the need for low-cost design of manufacturing and assembly. Figure 1 contrasts 1A technology hardware with that used as building blocks for the No. 1 ESS system.

II. INTEGRATED CIRCUITS AND APPARATUS

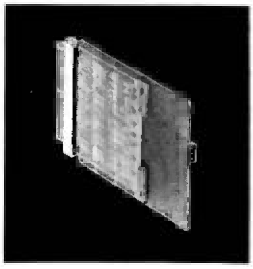
2.1 CDI-TTL integrated circuits

The design of the high-speed silicon integrated logic circuits evolved concurrently with the processor design in such a way that the overall system design was optimized. Choices of silicon integrated circuit (SIC) structure, processing, and logic gate design were based on general system performance requirements, available integration level, and the evolving physical design of the system during the 1968–1970 time frame.

Because of the processor cycle time, high-speed logic gates with propagation delays of from 5 to 10 ns were required. However, since relatively high gate packing density was desirable, and local frame ambient temperatures can reach 80°C (convectional cooling only), the gate power level had to be low. Accordingly, a low power-delay product was necessary. The combination of the collector-diffusion-isolation (CDI) structure[4–6] and the diode-modified transistor-transistor logic (DTTL) gate configuration provides a nominal power delay product better than 30 picojoules (for fanout = 1). This power delay product exceeded the best available at the time of the design, and provided a high gate density at low cost.



(a)



(b)

Fig. 1—Eight-bit adder comparison. (a) No. 1 ESS. (b) 1A technology.

Other system requirements, such as noise immunity, power distribution, and reliability influenced the choice of logic gate configuration, which evolved into a specific CDI-DTTL structure. Performance, reliability, and cost considerations finally dictated the adoption of a family of thirteen SIC codes, each embodied as a beam-leaded chip[7] of standard dimensions, and fabricated with a sealed junction technology.[8]

The CDI structure and processing, the CDI-DTTL gate design, and the SIC chip codes are discussed in the following.

2.1.1 CDI structure

Three device structures were considered for the 1A logic gates. These were the standard junction-isolated, buried collector (SBC), the air-isolated monolithic,[9] and the CDI structure.

The CDI offered the best combination of speed, power, and noise immunity, and, in addition, provided superior packing density and the simplest processing.

The CDI and SBC isolated transistor structures are compared in Fig. 2. In the SBC structure, the base and collector junctions are diffused into an n-type epitaxial layer, and a separate p+ isolation diffusion is required. In contrast, the CDI structure utilizes a buried n+ layer as its collector and a thin p-type epitaxial layer for the base region. It obtains isolation with a deep n+ diffusion, which also provides deep collector contact.

These differences make possible a 3-to-1 reduction in the required area of an isolated transistor, and reduce the number of required masked diffusion steps from five to three. It should be noted that the structure introduces some differences in electrical device parameters, notably in collector base breakdown voltage, inverse current gain, and device capacitances related to the collector structure. The CDI-DTTL circuit used in the 1A Processor chips maximizes the attainable performance for the CDI structure.

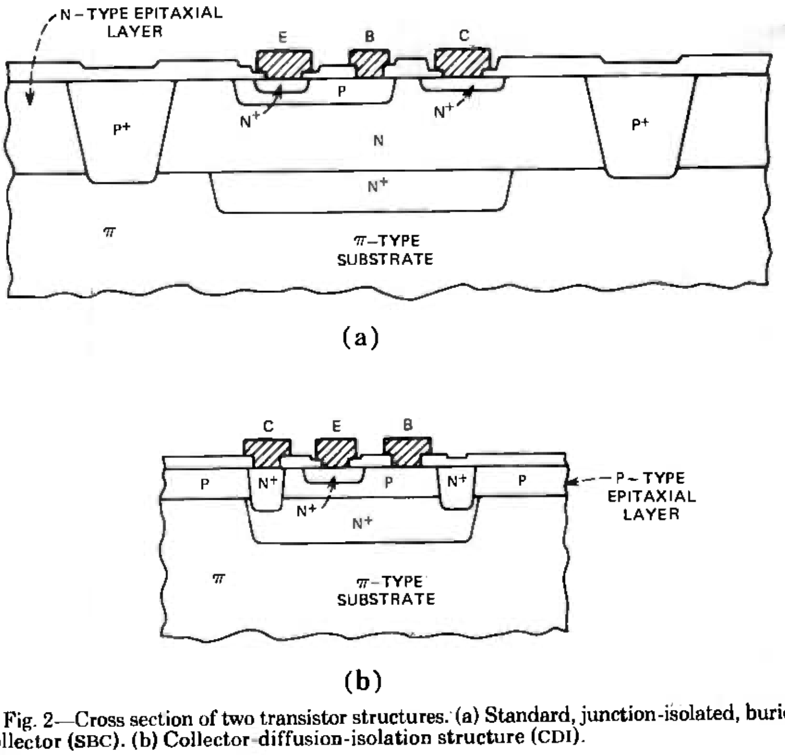


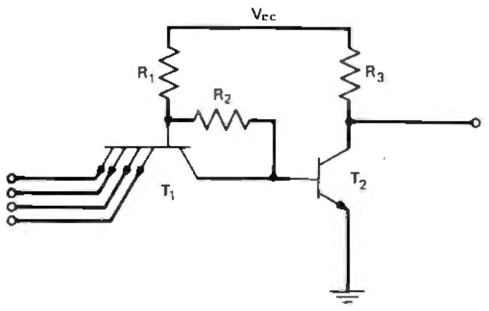
Fig. 2—Cross section of two transistor structures. (a) Standard, junction-isolated, buried collector (SBC). (b) Collector-diffusion-isolation structure (CDI).

2.1.2 CDI-DTTL gate configuration

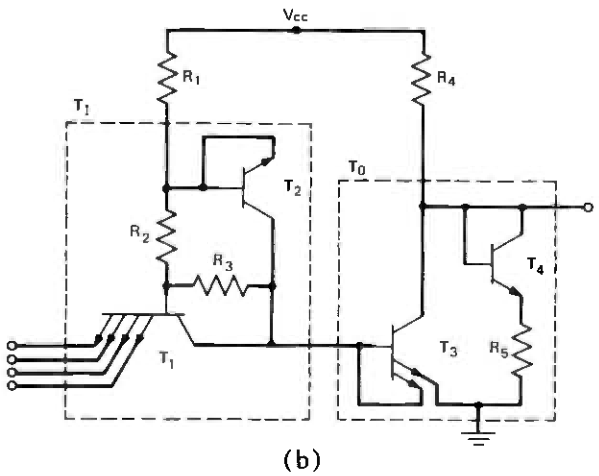
The objectives for gate performance included low-power, high-speed operation and ample noise margin for the operational junction temperatures of 0° to 90°C.

The basic gate shown in Fig. 3a can provide a power-delay product considerably better than other circuits, such as conventional TTL or ECL, primarily because it can be operated with reduced signal swings and low power-supply voltage. This circuit also allows outputs to be tied together to produce the wired AND function.

Taking this as a starting point, the gate design was modified to minimize the effects of higher inverse gain (α_I) and base collector capacitance



(a)



(b)

Fig. 3—(a) Basic TTL gate. (b) DTTL gate configuration for CDI.

associated with the CDI structure. As indicated in Fig. 3b, an input voltage divider network was added to the input circuit with a reference voltage obtained from a collector base diode. This serves to reduce and stabilize the effective α_I of the input array, and slightly increases the gate threshold. In addition, the output configuration has been modified by adding a diode clamp (T_4 and R_5) which controls the logic-1 level and reduces the gate propagation delay. Charge storage delays are minimized by the Buie clamp (emitter connected to base) on the output transistor, which acts to limit the degree of saturation.

The input and output structures shown inside the dotted lines in Fig. 3b (T_I and T_O) are readily fabricated as composite structures, resulting in an efficient layout with a minimum gate area.

This gate design provided performance consistent with system requirements for a power supply voltage $V_{CC} = 3.0 \pm 0.1$ V as follows:

(*i*) Propagation delay less than 5 ns for fanout = 1, nominal less than 7 ns for fanout = 3, and 20-pF loading.
(*ii*) Maximum fanout = 8.
(*iii*) Better than 200-mV minimum dc noise margins; temperature range 0° to 90°C.
(*iv*) Nominal power/gate = 6.3 mW.

2.1.3 Family of logic codes

A family of logic codes was developed to enable the synthesis of random logic functions with minimal design and assembly problems and a high degree of flexibility. The chips are all beam leaded to facilitate assembly on the thin-film hybrid circuits on which up to 52 chips can be interconnected. The chips are all identical in external dimensions (50 mils tip to tip), and each chip has 28 beam leads. The chips are protected during assembly and life with a junction seal[8] based on silicon nitride and, when utilized in conjunction with silicone rubber encapsulation, provide a highly reliable logic function.

The family of codes consists of six general-purpose codes and seven specific-function codes. General-purpose devices include a quad gate (two 3-input and two 5-input), a hex gate (four 2-input and two 3-input), a clipping hex, a clipping oct, a clipping input buffer (eight single RTL gates), and a dual high-power gate (two 4-input gates). The clipping gates are designed with special circuits to reduce ringing on long lines. The input buffer chip is designed to facilitate buffering between the backplane and the ceramic. The high-power gate, designed for driving high fanouts, low-impedance lines, large capacitances, or discrete devices, is designed with an active pull-up.

The seven functional chips are: a 2-bit register, a 2-bit register with gating, a gated delay flip-flop, a general-purpose translator, a dual-gated

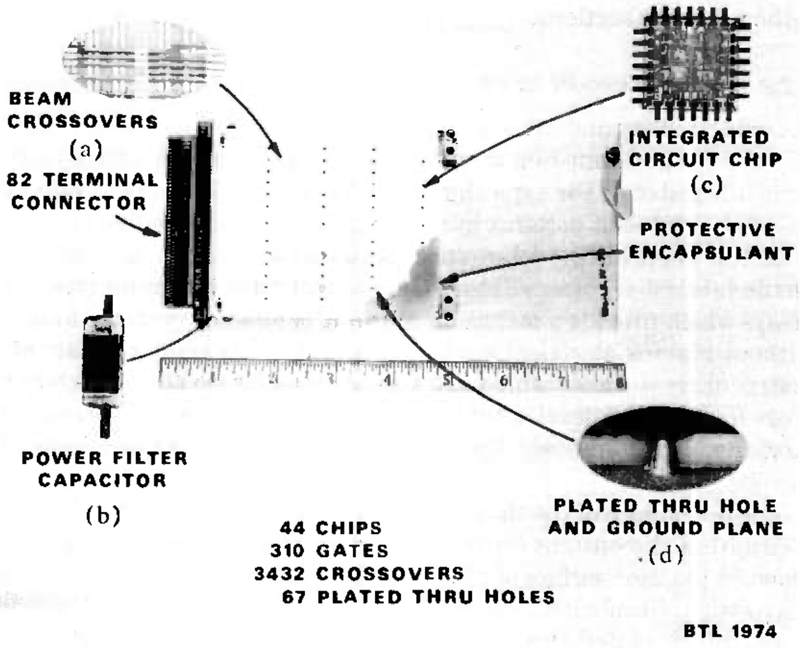


Fig. 4—Logic integrated-circuit packs.

quad, an unpowered dual-gated quad, and a dual-gated dual-gate configuration. The latter three codes are sets of gates with common gating leads.

The layout of these logic chips involved detailed attention to the effect of chip layout on ceramic wiring, since system packaging costs decrease with the number of gates per circuit pack, and increase with ceramic complexity. Underchip routing was also used advantageously where possible. Figure 4b is a microphotograph of a CDI-DTTL chip.

2.2 Hybrid integrated-circuit packages

Beam lead SICs of the types described in 2.1 are packaged and interconnected by the use of thin-film circuit patterns deposited on ceramic substrates.[10] These assemblies are termed HICs (hybrid integrated circuits). Two types of thin-film substrates are provided: a large 3.25- by 4.00-inch substrate which is the basis of a complete plug-in package and a family of small substrates with dual-in-line or staggered leads interconnected on printed-wiring boards. The thin-film circuits are described in the following sections.

2.2.1 Thin-film circuits for FA circuit packs

Logic circuitry and other circuits, which can be built almost entirely of SIC chips and thin-film components, are packaged on 3.25- by 4.00-inch substrates of the type shown in Fig. 4. The chips are bonded to thin-film conductor patterns by a thermocompression bonding process[11] (Fig. 4c). There are predetermined locations for a maximum of 52 chips on the substrate. There are also 841 standardized locations for crossover arrays which provide a means for allowing conductor patterns to cross without making electrical contact (Fig. 4a). Each array consists of a matrix of crossovers[12] which allow a maximum of seven conductors to cross four bottom-level conductors. All crossovers on a substrate are fabricated simultaneously. Crossover counts on ceramics range from 250 to 4000.

The impedance of the thin-film conductors is controlled by choosing appropriate dimensions for the conductors and by providing a ground plane on the back surface of the ceramic, so that signal paths are 75-ohm microstrip transmission lines. Ground is distributed to each chip through laser-drilled, plated-through holes in the ceramic (Fig. 4d). Power is provided to each chip with a grid of wide, low-impedance, vertical and horizontal conductors. A pair of power filter capacitors are mounted on each ceramic circuit (Fig. 4b).

In some applications, thin films are used to provide resistors as well as interconnections. Circuit packs requiring resistors usually combine analog and logic functions on a single ceramic. As many as 50 SICs, 90 resistors, and 35 appliqued capacitors have been required on a single ceramic substrate.

As described in 2.4, a connector assembly is added to the thin-film circuits to form a complete plug-in circuit pack. The circuits are protected from corrosion, dust, and mechanical damage after assembly by a silicone rubber encapsulant.[13] The encapsulant can be removed for circuit repairs. A ceramic circuit pack is shown in Fig. 4.

Standardization of the design features of the ceramics and SICs permits the use of a comprehensive computer-aided design system (Section 3.4) and permits the manufacture of the 500 circuit codes required for ESS systems on a standardized manufacturing line.

2.2.2 Lead frame HICs

Circuitry which requires a combination of SICs and discrete components (power transistors, inductors, etc.) is not packaged on a large ce-

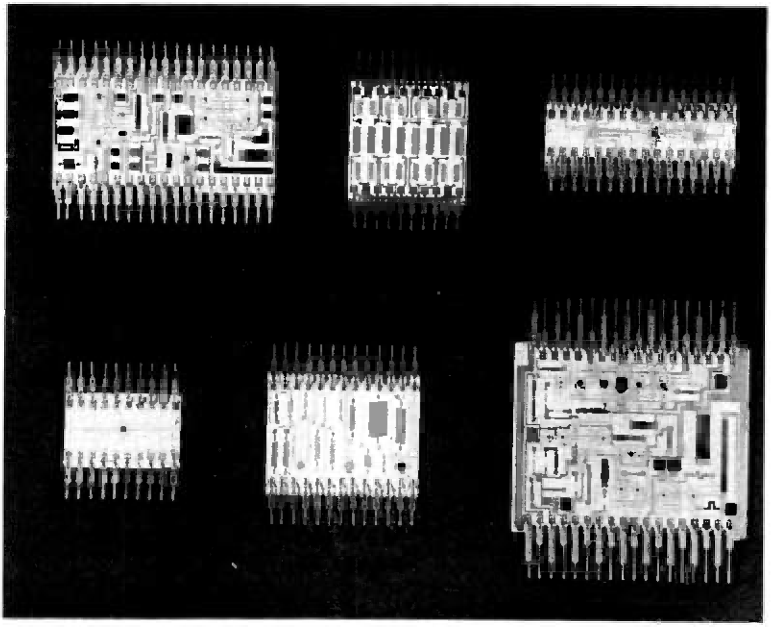


Fig. 5—Small hybrid integrated circuits.

ramic substrate. Instead, the SICs are bonded to thin-film circuits on small ceramic substrates. Dual-in-line or staggered leads are then thermocompression bonded to the substrate to form packages that can be inserted in printed-wiring boards for interconnection to other components. A representative sample of small HIC designs is shown in Fig. 5. Note that there are often many thin-film resistors and crossovers in addition to the thin-film interconnection patterns.

2.3 Discrete circuit packs

Circuits involving combinations of lead frame HICs and discrete components are packaged on discrete circuit packs.

The HICs are standoff mounted through holes on $\frac{1}{16}$-inch-thick epoxy-glass printed-wiring boards measuring 3.67 by 7 inches. A typical discrete circuit pack is shown in Fig. 6. Some of the packs require printed wiring on both sides of the board and in these cases plated-through holes are used to interconnect the two levels of wiring. However, many of the circuit packs use single-sided boards. All components and printed-wiring conductors are located on a standardized grid of 0.05 inch to facilitate computer-aided design and also to permit automated assembly and test.

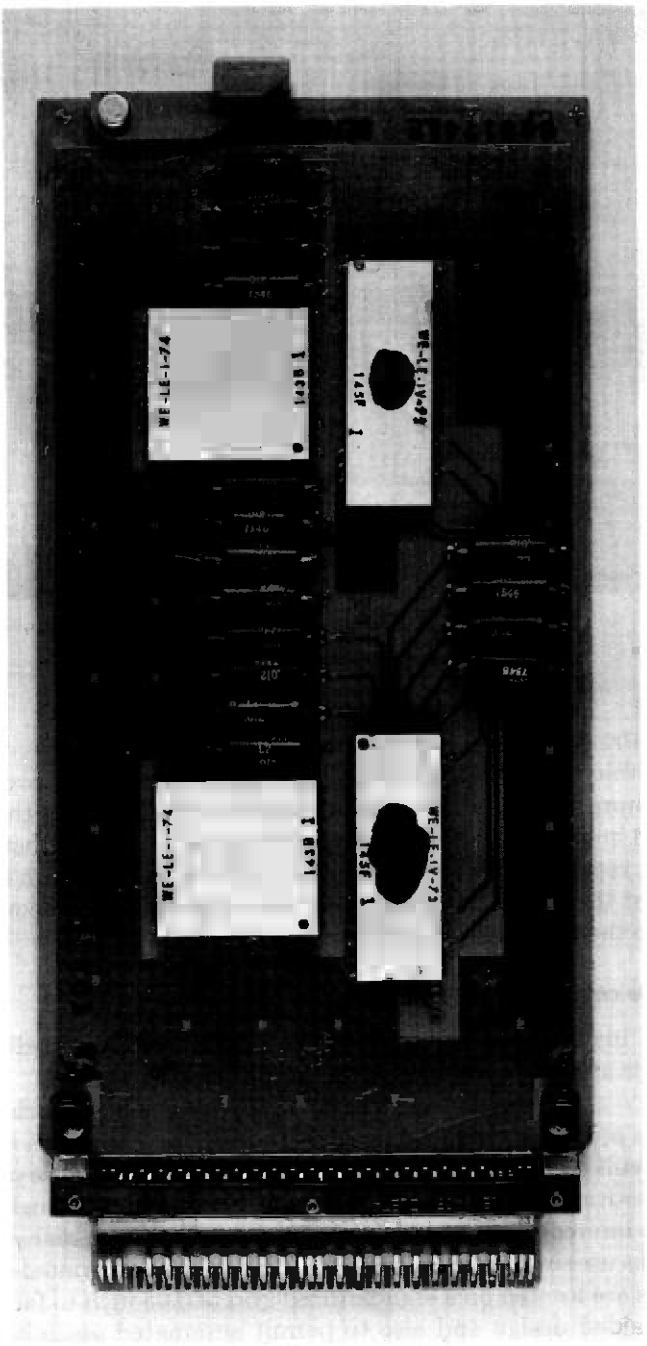


Fig. 6—Discrete circuit.

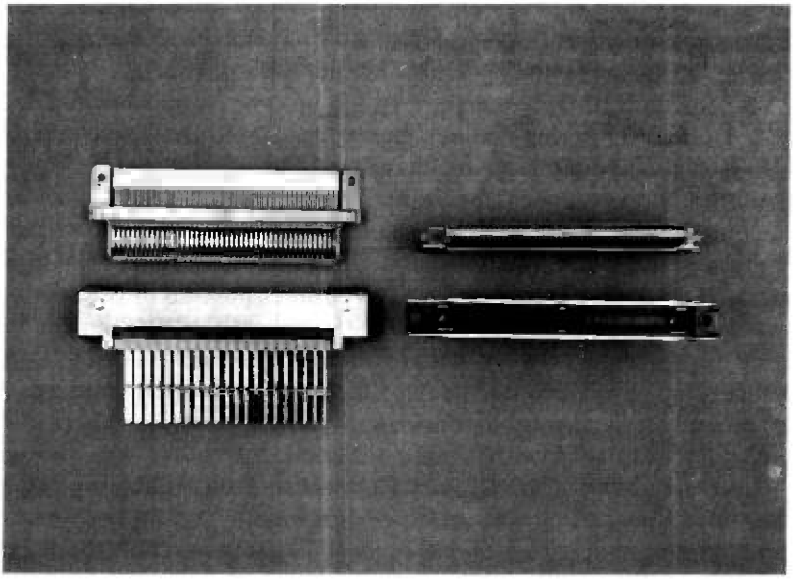


Fig. 7—Circuit-pack connectors.

An insulating cover coat is applied over all printed-wiring conductors to protect them from fingerprints and other contaminants that could degrade the insulation resistance of the pack. Conductors on the component side are coated before component assembly. The wiring side is coated after all assembly, mass soldering, and electrical testing is completed.

2.4 Circuit-pack connector

The circuit-pack connectors[14] shown in Fig. 7 link the circuit packs to the next higher level of assembly, the backplane unit. These connectors posed some difficult design problems because of the stringent requirements imposed by the physical and reliability needs of the system and also by the importance of low cost. Connector design requirements include a low and stable contact resistance over a 40-year life. This is particularly important in new systems, such as the 1A Processor, because of their low signal levels. Design requirements also include low contact surface wear for several hundred insertions and withdrawals, accommodation of manufacturing tolerances to permit low-cost manufacture, and a minimum end-of-life contact force of 100 grams. Further, a high contact density is required to be compatible with the high-density packaging needs of the 1A Processor. In addition, the high speed of the logic gates impose significant constraints on the electrical performance of the connector. Controlled impedance is necessary as is a high degree of crosstalk isolation between contacts. Electrical performance of the connector is a key factor in establishing design concepts.

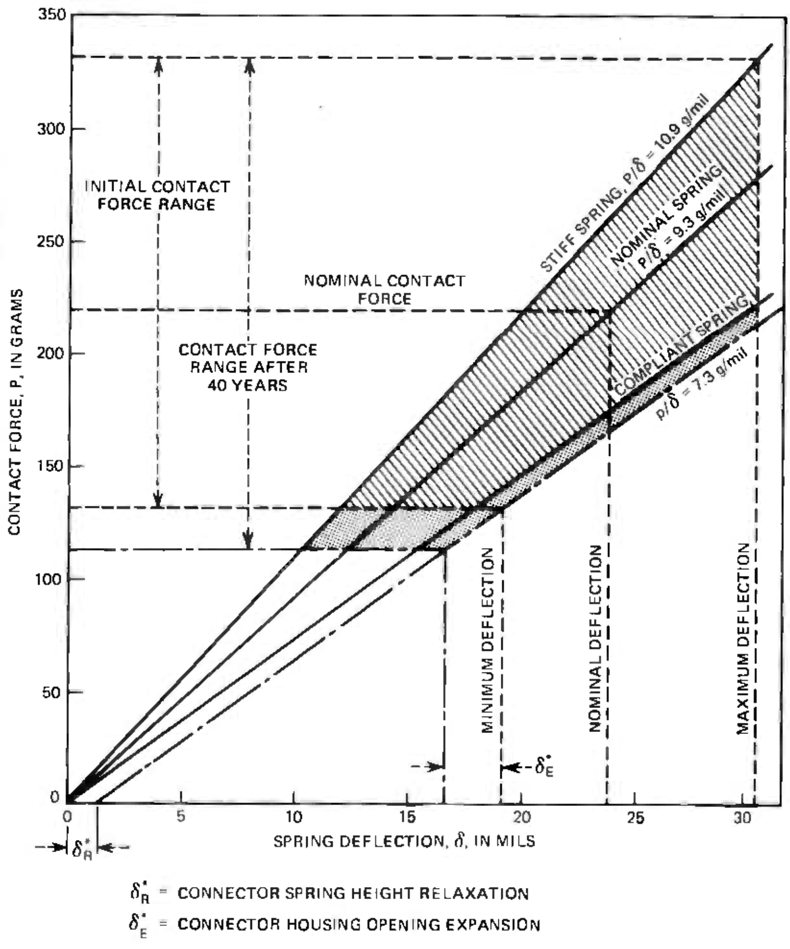
The plug half of the connector, which becomes a part of the circuit pack, contains the spring elements that apply the required contact forces. The individual contact springs have two distinct portions: a contact portion where electrical contact is made with a mating receptacle contact and a terminal portion where a permanent electrical connection is made with the circuit-pack conductor metallization. Deflection of the spring occurs in the area of the contact. The contact portion of the spring is beryllium copper and the terminal portion is either beryllium copper or copper, depending upon whether it is to be attached to a discrete pack by soldering or to a ceramic pack by thermocompression bonding.

The contact portions of the springs are electroplated first with a nickel underplate and then with a wear-resistant gold alloy. The terminal portions are plated with 24K gold. Connectors are available with either 42 or 82 contacts. Eighty-two contact versions are used for both ceramic and discrete circuit packs. The 42-contact design is used only for discrete packs.

The receptacle half of the connector, which becomes a part of the backplane unit, includes the mating contacts incorporated into a housing that protects the contact surfaces from damage. The housing also locates the circuit-pack plug correctly so that upon insertion, the necessary mating contacts are in proper alignment. Metal side plates riveted to the housing serve two functions. They resist the load due to the contact force of an inserted plug connector and they provide a supplementary electrical ground to the frame.

The receptacle contacts do not deflect. They are made from a copper-nickel-tin alloy with a 24K gold strip inlaid in the contact area. The terminal ends of the contacts are formed so that the double row of contacts inside the housing is converted into four columns of terminals (in the case of the 82-pin connector) for backplane interconnections. The terminals are 0.025 inch square and are located on ⅛-inch centers in both the horizontal and vertical directions. The 42-contact connector is similar, except that it has only two vertical columns of terminals and these are located on ¼-inch horizontal centers. Both the plug and the receptacle have special power and ground contacts to provide low resistance and low inductance. Dedicated power and ground terminals appear at each end of the connector, with two ground terminals in the center.

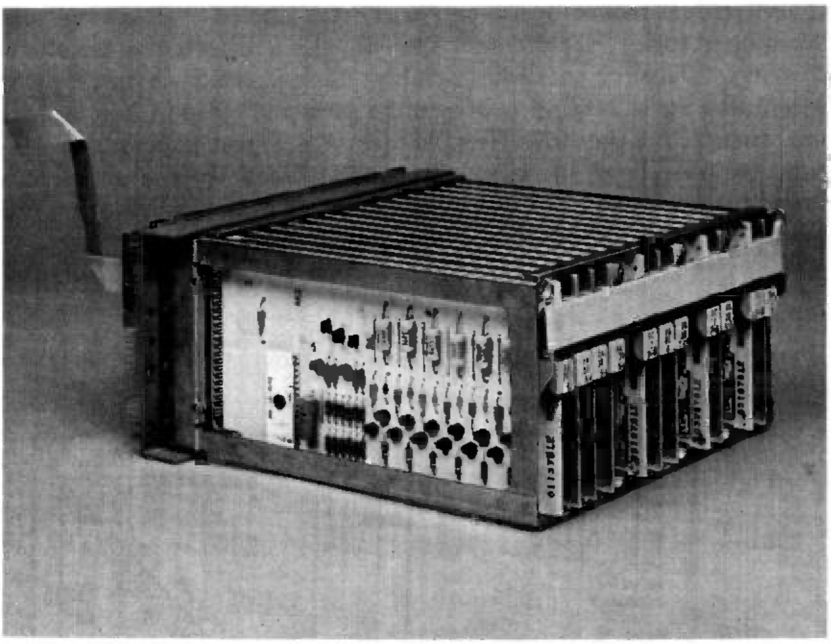
The contact force of the connector is affected by many factors, as is shown in the force deflection curve of Fig. 8. Variations in the spring properties of the beryllium copper material or its dimensions can, for a given deflection, appreciably affect the contact force. The extent of this effect is indicated by the curves in Fig. 8 labeled "stiff spring" and "compliant spring." The contact force will also be affected by variations in many of the piece-part and assembly dimensions in both the plug and receptacle. Stress relaxation in metallic parts and creep in plastic parts are other factors that will affect contact force. When all possible variations are considered and their effects determined, the contact force is found to vary from its nominal value of 220 grams to a maximum of 335 grams and a minimum end-of-life value of 113 grams.



δ_R^* = CONNECTOR SPRING HEIGHT RELAXATION

δ_E^* = CONNECTOR HOUSING OPENING EXPANSION

Fig. 8—Replaceable terminal connector, contact force window.



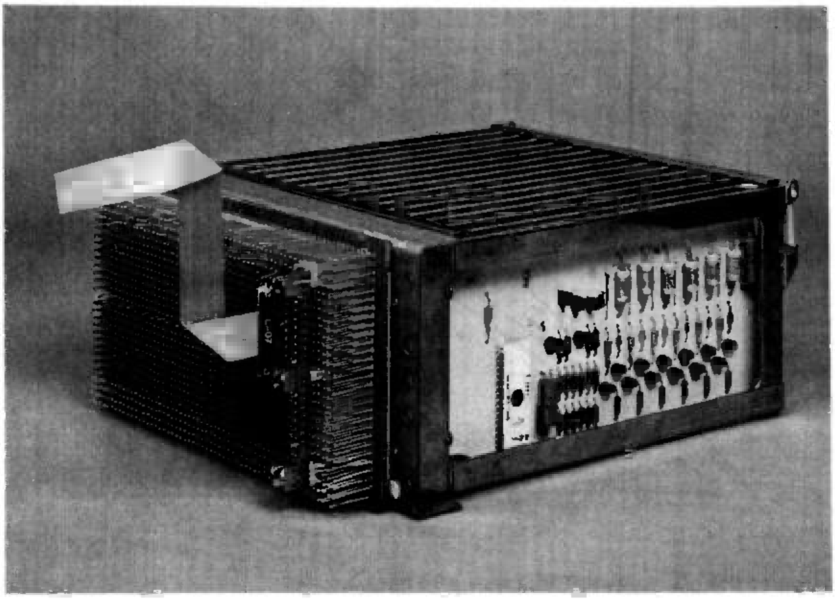


Fig. 9—Unit backplane.

2.5 Backplane unit

The next level of assembly is the backplane unit. This is built up of a number of circuit packs assembled on a common mounting structure and interconnected at the backplane to perform one or more specific functions. Generally, in a backplane unit the circuit packs are located on ½-inch horizontal centers (or multiples thereof) and 4-inch vertical centers. Occasionally, ¾-inch horizontal spacings are used.

The connector receptacles are secured to a mounting plate and apparatus mountings are added to the assembly, as shown in Fig. 9. The apparatus mountings serve to support the circuit packs and also provide alignment during insertion so that the circuit-pack plugs engage the receptacles correctly. The apparatus mounting is designed to provide a minimum impedance to the circulation of air past the circuit packs. A designation strip is provided on the front of the mounting so that the circuit packs can be readily identified. Package designations in the form of a letter-number combination are included both on the designation strip and on a plastic faceplate riveted to the front end of each circuit pack. Correct location of a circuit pack is assured by matching the code number of the circuit-pack faceplate with that in the apparatus mounting designation strip. Experience has demonstrated that this method provides adequate safeguards against incorrect pack insertions.

Interconnections between the circuit-pack receptacle terminals in a given backplane unit are accomplished by two methods. Multilayer boards are used for power and ground connections and some signal connections. Most of the signal connections, however, are made with 30-gauge wire that is machine wrapped. Electrically sensitive leads are applied manually, generally as tight-twisted pairs or miniature coaxial cables.

The multilayer boards range from 2 to 15 layers and vary in size from 4 by 8 inches to 12 by 22 inches. Interconnection between the various layers is accomplished with plated-through holes 0.040 inch in diameter after plating. The holes are located on ⅛-inch centers in both directions to match the terminal spacing of the circuit-pack connector.

Multilayer boards are assembled to the backplane unit, as shown in Fig. 9. Each terminal of the connector protrudes through a plated-through hole. A mass soldering operation completes the assembly of the backplane. In this process, each terminal is soldered to an external plated-through hole land on the multilayer board. The design of the board determines the connectivity between the connector terminal and the internal layers of the multilayer board.

Although the circuit-pack connectors are mounted on 4-inch vertical centers, their terminals occupy only 2½ inches of this dimension in the backplane terminal field. The balance of the area is filled with additional terminals (0.025 inch square) used for terminating coaxial cables and tight-twisted pairs. In the 1A Processor, coaxial cables originate and terminate on these terminals rather than on the connector terminals. Machine-wrapped 30-gauge wires are used for the connections between the connector terminals and the coax terminating field. This practice restricts coax congestion to the areas between connectors where wire guides can be located and leaves the connector terminal field relatively unobstructed for trouble shooting and for accepting cable connectors.

The connector terminals and the coax terminating field terminals extend past the multilayer board for a length sufficient to accommodate two wire-wrapped connections. There is also enough additional length to permit cable connectors to be plugged directly onto the connector terminals. Special versions of connectors are available with gold plated terminal tips so that they can serve as male contacts for the cable connectors.

2.6 Frame

The final level of packaging is the frame that consists of one or more backplane units mounted on a common structural frame. Interconnection at this level may be between backplane units on the same frame or between different frames. In either case, it is accomplished with connectorized cable assemblies. All 1A Processor backplane units are designed to be mounted from the front of the frame. These two features reduce frame assembly and test costs, and facilitate field maintenance and growth.

The contacts used in the cable connectors are of a box-type construction designed to mate with 0.025-inch-square male terminals. These contacts are incorporated into a molded plastic housing to form a connector assembly with contact spacings that are compatible with those of the circuit-pack connector terminals. There are 10- and 20-contact versions of the connector available. A 24K gold button is welded at the appropriate point in each contact to form its contact finish.

The design requirements for the cable connector are similar to those of the circuit-pack connector. Contact force of the cable connector varies from a nominal value of 260 grams to a maximum of 365 grams with a minimum end-of-life force of 112 grams.

For attachment to cables, the connector assemblies are mounted on small printed-wiring boards, as shown in Fig. 10. The cable wires are also terminated on the board, and printed conductors interconnect the cable wiring with the female contacts of the connector. The solder joints of the cable wires are stress relieved by the addition of small plastic clamps that are riveted to the board so that they clamp the cables securely.

Two basic types of cable are used for backplane unit or frame inter-

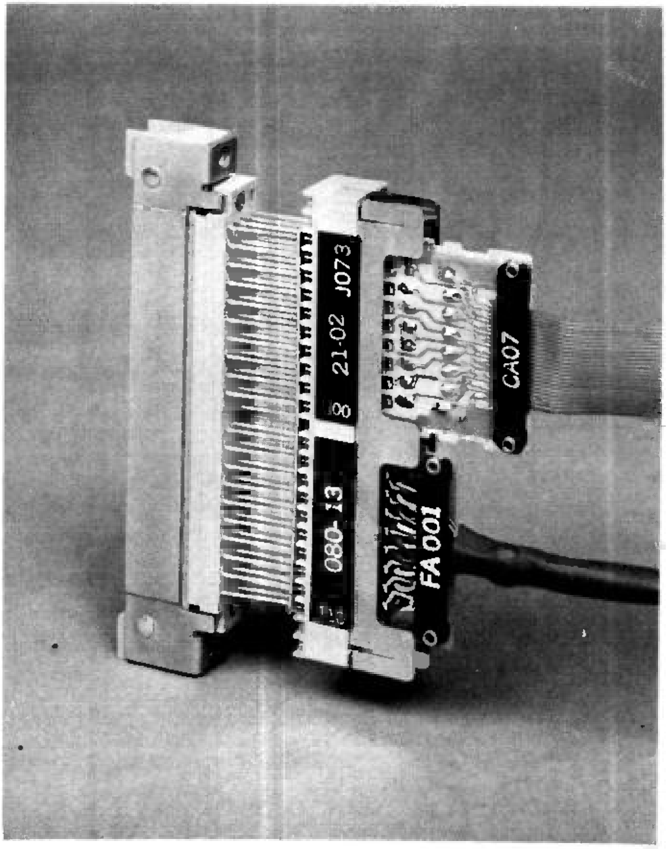


Fig. 10—Backplane cable connectorization.

connections. They are switchboard cable and flat tape cable. The switchboard cable is used for the transmission of balanced signals. It is composed of a number of pairs (usually eight) of tight twisted 26-gauge wire, with pvc insulation. The cable is contained inside a pvc jacket for abrasion resistance. The flat tape cable is used for dc or unbalanced signal transmission. A widely used version of flat tape cable contains 31 parallel conductors of 30-gauge wire spaced on 26-mm centers and imbedded in Teflon.* Only eight of the wires are used for signal transmission with the remaining 23 being used for grounds. Three ground wires are interposed between each signal wire for crosstalk isolation.

* Registered trademark of E.I. Dupont de Nemours & Co.

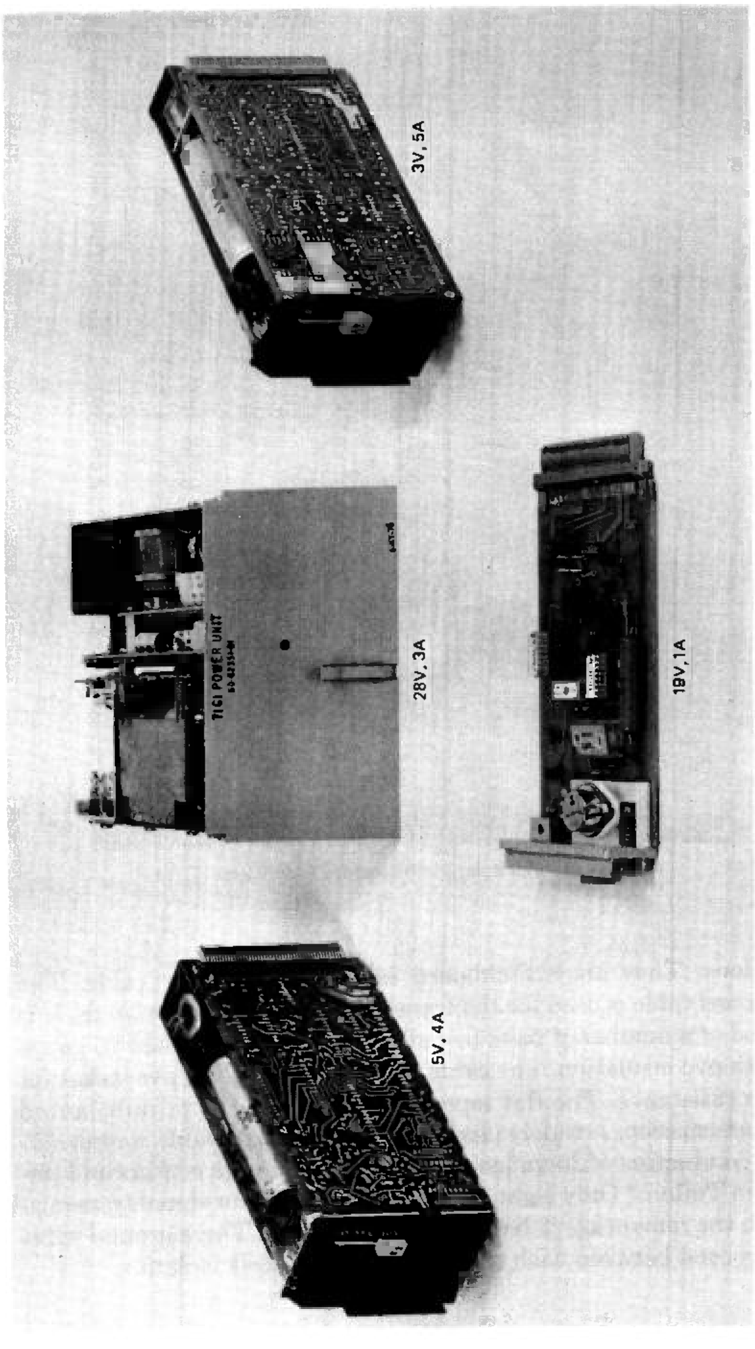


Fig. 11—1A technology power supplies.

2.7 Power supplies

The design of power supplies for the 1A Processor was based on commonality within the family, performance consistent with 1A technology, pluggability, extended environmental performance, and low cost. To achieve these characteristics the following criteria were met:

(*i*) Members of the family have a common physical format and incorporate standard codes of components and subassemblies.

(*ii*) The family of power supplies are mountable in standard apparatus housings and can be installed either adjacent to or remote from the circuitry it powers.

(*iii*) All power supplies are pluggable using standard 1A connectors and operate from −48 V and +24 V.

The result is a family of power modules (see Fig. 11) that have many common features, thereby enhancing overall manufacture. A detailed description of the power supply design is covered elsewhere.[15]

To complement the use of the power supply, a common use circuit pack was developed to be used in each frame. This pack is used to implement automatic power alarm testing, monitor power system status, and control the power switch. The power switch (see Fig. 12) developed for each independent power system (i.e., a frame or unit) is used for switching high currents, signaling, controlling status lights, fuse alarms, and lamp testing. The switch provides two pushbuttons on the front face for turning the power on or off. Visual power system status is provided by four lamps located on the front face. The switch, in conjunction with the power supply and common pack, provides a comprehensive means of providing power to the individual units.

III. TECHNOLOGY DESIGN AIDS

3.1 Crosstalk and noise margin design

All aspects of noise generation were extensively analyzed. These sources included connector crosstalk, stripline coupling on thin-film ceramics, backplane wiring coupling and EMI susceptibility, ground and power transients, resistive losses, etc. Each of these sources was correlated with physical design parameters and the total noise margin was allocated among the individual sources in a manner designed to minimize overall hardware costs. The approximate allocations used were 30 percent for the connector, 10 percent for ceramic-circuit-pack interconnection, 50 percent for backplanes, and 10 percent for all other sources.

The development intervals were minimized by using these allocations to develop a set of interconnection guidelines that provided audit thresholds for computer screening of potential trouble areas. This re-

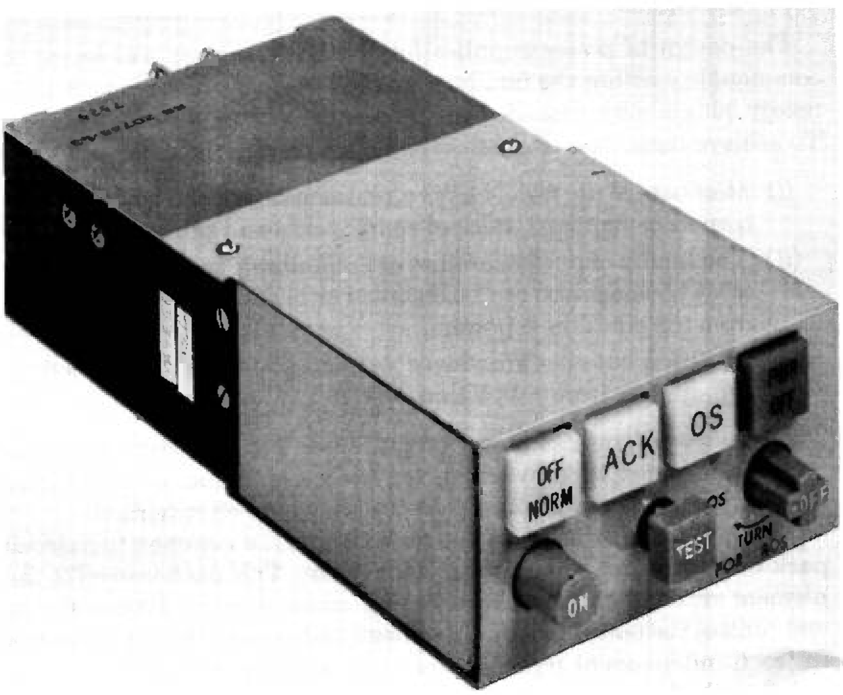


Fig. 12—Frame power and control switch.

lieved design engineers of much of the normal noise analysis and increased accuracy.

3.2 Timing analysis programs

To accurately predict critical timing delays in logic chains efficiently, two computer-aided design programs were developed in conjunction with the technology. The first program is an interactive program designed to implement the delay calculations outlined in a comprehensive electrical design guide developed for the technology. The delays through each gate are broken up into three components: intrinsic gate delay, delay due to capacitive loading, and signal propagation delay in the interconnections. In a typical path, these components may account for 50, 40, and 10 percent of the delay time, respectively. The parameters used in the program include worst-case gate characteristics, type and number of driven gates (loads), the number of load resistors, the number of collector tied outputs, the type and length of each wiring component, the type and number of connectors, the number of crossovers on the ceramic, the

Table I — Major thermal analysis programs

Topic	Program Name	Nature of Program
Silicon integrated circuits (SICs)	TASIC	Analytical
Ceramic and discrete circuit packs	FLAME	Analytical
Finned heat sinks	HEATSINK	Experimental
Frame thermal performance	FRAME	Experimental, analytical

length of adjacent wiring, and the type of the driven gate. These parameters can be entered manually or automatically from a design data base. Output is maximum, minimum, and typical propagation delay times.

For cases where the full analog characterization of the gates driving a transmission line interconnection media is necessary (typically long, high-speed, nets), the second program was developed. This program incorporates a lossless transmission line model and a library of gates modelled at the transistor level into an existing full-capability circuit-analysis program.

These programs have been used extensively in the design of ESS frames and typically predict delays to within 5 percent of actual measurements.

3.3 Thermal analysis programs

Thermal design aids were developed from both analytical and experimental studies to provide equipment designers with the tools to accurately predict device junction temperatures and optimize the use of natural convective cooling. These aids cover the whole range of design from SIC thermal analysis to entire frame thermal analysis. The major programs are shown in Table I. In addition to the specific design programs, extensive attention was directed towards the thermal optimization of 1A technology. Ceramic circuit-pack thermal performance was amplified through the incorporation of a metallic heat sink. Apparatus mountings and associated pack spacing were designed to maximize natural convection air currents while optimizing packaging density. Aids such as frame baffles and air channels were developed to further enhance thermal performance. As a result of this basic effort in apparatus design, the high-performance hardware, coupled with the previously mentioned programs, allowed the frame designers to densely pack the frame while assuring the required level of thermal performance.

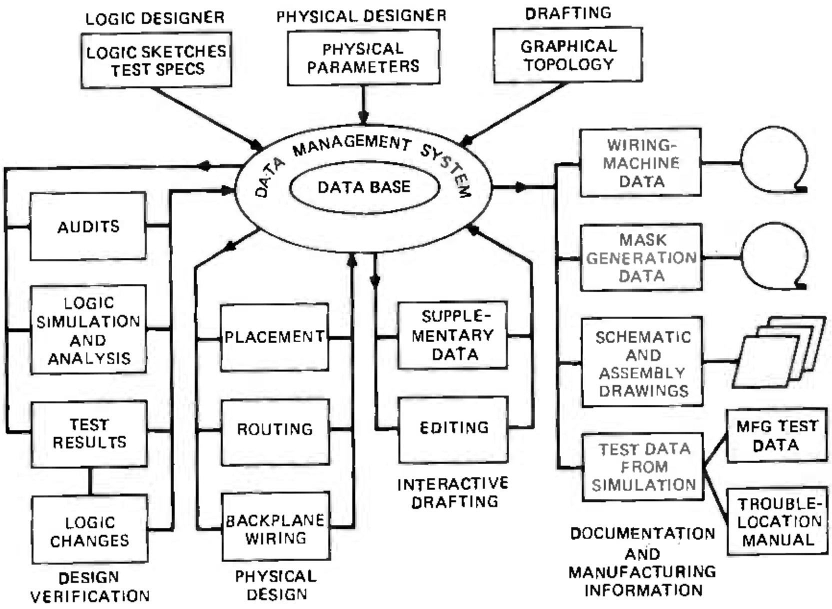


Fig. 13—Computer-aided development and manufacturing system.

3.4 Computer-aided development and manufacturing systems

A complete computer-aided development and manufacturing system was developed for 1A technology. It permits the design specifications to be introduced into a data base and enables their subsequent use by a variety of application programs (Fig. 13). Through these programs, the design is built and simulated in a general-purpose computer. The simulation allows the elimination of many clerical, encoding, and logic errors before any hardware is constructed. This data base, in conjunction with application programs, is used to do the partitioning, layout, and routing of the ceramic circuit packs. Extensive audits, such as crosstalk exposure, loading, etc., are incorporated to ensure adequate electrical design margins. While most programs are automatic, the capability is also available for interactive design in preparing circuit designs and test programs. Interactive capability is also provided to add supplementary data and edit the design.

Manufacturing information is generated from this data base and is used by Western Electric to generate art masters for circuit packs and backplanes. This provides machine-readable information in controlling automatic wiring machines. To assure compatibility of test requirements with test facilities, Bell Laboratories and Western Electric jointly developed computer-controlled test facilities. This assured that the test

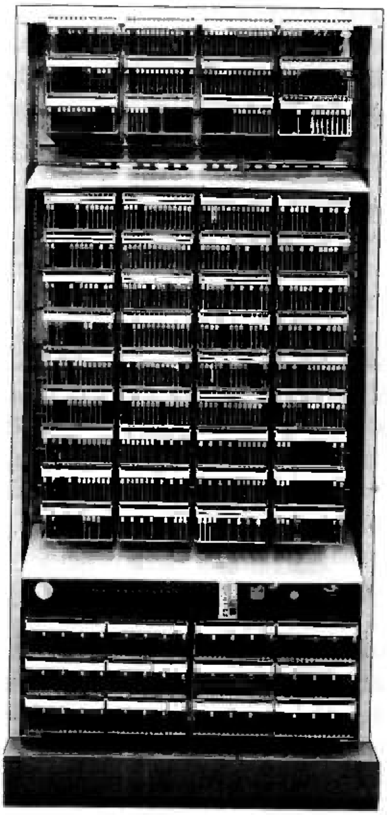


Fig. 14—Front view of central control.

information derived from the design data base could be used directly to test circuit packs, backplanes, frames, and systems during manufacture.

This same information is also used to generate documentation such as schematic diagrams and trouble location manuals furnished to the telephone companies. Figure 13 summarizes the features of the machine-aids system.

The machine aids have been invaluable for a development the size of No. 4 ESS. Without them it is questionable whether the job could have been done. Certainly it could not have been done on the schedule achieved.

IV. EXAMPLE OF 1A PROCESSOR FRAME

The central control frame provides a comprehensive example of a 1A Processor frame, as shown in Fig. 14. The circuit-pack equipment on each

CC frame is comprised of 234 logic packs, twenty-six 40-pin discrete circuit packs, two-hundred 80-pin discrete circuit packs, forty-eight 3-V power modules, and seven 19-V power modules. There are also a power control switch, fuse panels, and a number of general-purpose relays for power-control sequencing. This equipment is accommodated by a single-bay sheet-metal framework 7 feet high and 3 feet 3 inches wide. Structurally, this framework is identical to those used in No. 1 ESS except that the front-to-back depth is 1 foot 6 inches instead of 1 foot. The additional depth was utilized in the disk and core memory designs and also provided more wiring space for connectorization in the backplane.

Functionally, the CC circuit packs are subdivided as logic consisting of 50,000 gates on 250 ceramic and discrete circuit packs, communication bus consisting of 160 discrete packs, and power modules and associated fuses and control relays. As illustrated in Fig. 14, the first two levels are primarily equipped with communication-bus packs for the system peripheral units. The next seven levels accommodate the logic packs. The last three circuit-pack levels accommodate communication-bus packs for call stores, program stores, and auxiliary units. Power modules occupy the first three levels from the bottom of the frame, with fuses, power switch, and control relays being located directly above the power modules. The two horizontal gray panels are baffles that deflect heat from the power supplies to the rear of the frame. The design considerations for physical placement of these circuit packs were essentially (*i*) reliable circuit operation, (*ii*) manufacturing methods, (*iii*) installation methods, (*iv*) ease of maintenance, and (*v*) ability to implement changes. While these design considerations are not new, they were greatly influenced by measurable increases in device packing density, wiring terminals, switching speeds, and power dissipation per unit area. They were also influenced by the requirements for complete factory system tests.

Figure 15 shows the frame wiring consisting of 42,000 segments and associated hardware. Power and ground for the 3-V logic is implemented by means of multilayer printed-wire boards. This accounts for about 20,000 wire segments and provides for low-impedance power distribution to meet noise margin requirements of the high-speed logic devices. Of the remaining 22,000 wire segments, 90 percent are automatically wrapped, 30-gauge wires. The segment lengths for these wires are for the most part 13 inches or less. Longer segments are normally 75-ohm coax or 30-gauge twisted pairs, as determined by crosstalk susceptibility. A small number of wire segments are 100-ohm coax to provide for special clock pulse transmission. As indicated by Fig. 15, special wire guides are provided to retain the coax wiring in an orderly array and allow easy access to connector terminal field.

Interunit wiring logic connections between units are made by con-

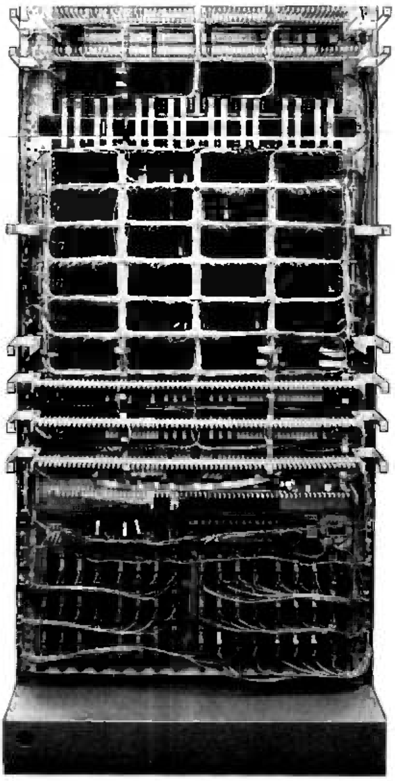


Fig. 15—Rear view of central control.

nectorized flat ribbon cables, as described in Section 2.6. The connectors are plugged either directly on wiring terminals or 80-pin connectors or on terminals of specially arranged connector fields. Interunit connections from the power unit to other units are also made by connectorized cable assemblies. These cables are made up of combinations of 14- or 16-gauge pairs, shielded 28-gauge pairs, and 26-gauge singles and pairs, as determined by power-interconnection requirements. This approach allows simplified unit replacement for repairs or design changes.

Interframe bus cable assemblies are connectorized lengths of 8-pair, 26-gauge sheathed cables. CC-to-CC logic interframe cable assemblies are connectorized flat ribbon cables. Power feeder cables from the power distributing frame are two pairs of 10-gauge wires for each CC. These cables are also connectorized and they are plugged into jacks located on the rear of the frame top channel.

As noted earlier, the frameworks for 1A Processor frames differ from those of No. 1 ESS only in that they are 1 foot 6 inches deep rather than

1 foot, and added depth is in the rear. In the case of the central control, the added depth is used to accommodate the volume of intraframe and interframe wiring and associated retaining hardware. With file stores and core stores, which account for most of the processor frames, added depth is used to accommodate the memory modules.

V. STANDARD FLOOR PLAN AND POWER SYSTEM

5.1 1A Processor floor plan

The 1A Processor utilizes a fixed floor plan with associated standard cabling and hardware (see Fig. 16). The fixed floor plan concept provides a number of significant advantages that assure the proper operation of the processor while minimizing overall cost. These advantages are:

(*i*) Line engineering is reduced to a minimum. Since there are no variations or options, decision making is almost nonexistent.

(*ii*) Manufacturing is simplified. All hardware, power and bus cable, cross-aisle racks, etc. are the same for all processors. All cables are pretested in the factory.

(*iii*) Installation interval is reduced to a minimum. Since every installation is the same and all cables are factory made, the processor can be installed in less than a week with a minimum of errors.

(*iv*) Growth of the processor is accomplished in an orderly fashion. Frames are grown only as sequentially numbered and always in the position indicated on the floor plan.

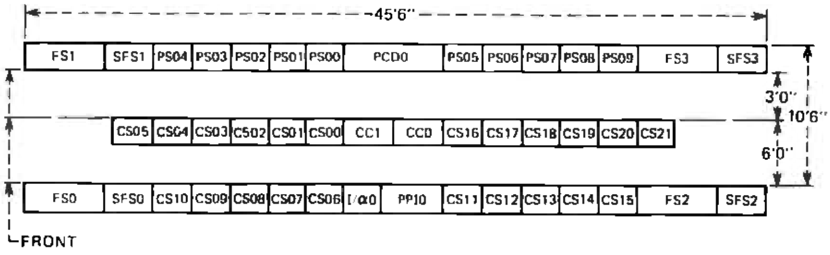
(*v*) Critical interframe bus timing is assured. Because of the critical timing parameters between the CC and the store frames, bus cable length is a critical factor. Since the cables are precisely made and tested at the factory, and not subject to installation circumstances, the pulse windows are guaranteed.

(*vi*) Thermal and floor-loading performance is optimized and assured. The fixed floor plan defines the boundaries of thermal flux and floor loading, thereby simplifying the office engineering.

While the fixed floor plan has sacrificed some installation flexibility, the advantages have far outweighed the disadvantages and have helped assure the high performance of the processor at overall minimum cost.

5.2 No. 1A Processor power system

Power in the 1A Processor has been integrated to a degree not found in earlier systems. From the −48 V energy source (battery or converter), power is supplied by duplicated feeders to the power conversion and



CC	CENTRAL CONTROL
CS	CALL STORE
PS	PROGRAM STORE
FS	FILE STORE
SFS	SUPLEMENTARY FILE STORE
PCD	POWER CONVERSION AND DISTRIBUTION
I/O	INPUT/OUTPUT
PPI	PROCESSOR PERIPHERAL INTERFACE

Fig. 16—Fixed floor plant of 1A Processor at full growth.

distribution frame (PCDF). This duplicated frame contains bulk −48 V to +24 V converters, +24 V and −48 V distribution filters, fuses, and built-in filter-charging facilities. A solid state fuse alarm and indicating system is employed that is testable under manual or system control. Newly developed, fast-blowing fuses are employed to minimize voltage transients caused by feeder faults.

Frames that are duplicated are fed power from one section of the PCDF. Nonduplicated frames are fed from both sections of the PCDF and the power is ORed at the frame. The dc-to-dc converters on the frame provide the required voltages and contain overcurrent detectors for fault detection and voltage monitors to assure in-range voltages. Standardized man-machine and frame-system interfaces are a feature of the power system and permit tests of the monitors and alarms. Particular attention has been paid to assure a reliable, safe, easily maintainable, and recoverable system.

VI. TECHNOLOGY PERFORMANCE

6.1 Electrical performance

One of the major challenges of the 1A Processor technology was to build complex, high-performance systems, such as No. 4 ESS, utilizing millions of logic gates with small signal swings, low noise margins, and low-power-supply voltages and, at the same time, to achieve a minimum of noise problems, high-speed operation at low power, and superior electromagnetic susceptibility and radiation performance. A number of factors permitted the use of low voltages including the low saturation

Table II — Failure rates (in FITs) of No. 4 ESS components and circuit packs in system labs

	Component Hr	Cumulative	Instantaneous	Objective
Ceramic circuit packs	2.8×10^8	733	310	450
CDI TTL integrated circuits	8.0×10^9	9	4	10
Connectors	6.2×10^8	9	≪1	19
Backplanes	7.5×10^{10}*	0.04	≪1	0.03
Power supplies†	1.9×10^7	1090	1700	2500

* Plated-through holes.
† Excludes three codes that are undergoing redesign.

voltage of the CDI process, the gate circuit design, careful design of the ground and power distribution, high-density packaging, controlled impedance wiring, and a comprehensive set of wiring rules, audits, product specifications, and controls. The net result was a fully characterized logic family and interconnection technology capable of achieving typical propagation delays of 7 to 10 ns (in a frame that includes all loading factors) at a power of 6 mW/gate. The controlled interconnections, high-density packaging, and power and ground distribution resulted in an electromagnetic susceptibility and radiation performance exceeding previous ESS systems, which used slower and higher noise margin logic families.

6.2 Reliability performance

The measured failure rates of 1A technology hardware are very low. A large amount of hardware has been built and placed in operation in the No. 4 ESS Systems Labs at Bell Laboratories in Naperville, Illinois. This equipment, plus that in the first commercial No. 4 ESS office in Chicago, provides a basis for measuring the reliability of the hardware. The unit of measure used to describe the failure rate of individual components is the FIT. One FIT is defined as one failure in 10^9 component operating hours. As can be seen in Table II, the cumulative failure rates are very near the objective failure rates established for long-term system operations. Much of the hardware upon which the reliability data of Table II are based is of early manufacturing vintage and has been exposed to much change activity. Nevertheless, the failure rates are approaching the objectives. The column labeled "instantaneous" in Table II represents the slope of the cumulative failure rate with time and can be regarded as representative of the current performance of the hardware. Note that the instantaneous failure rates are all below the objective failure rate. The reliability information obtained to date indicates that the system will exceed its design objectives. The low failure rates of components indicate the success of a design philosophy that is based on careful and thorough engineering design using well tested and characterized materials and manufacturing processes. These create inherently reliable products and avoid "testing" or "burning in" reliability.

6.3 Thermal performance

Thermal considerations and design for the 1A Processor began very early in the development program. Strict attention was paid to maximizing the thermal performance of the hardware and developing techniques to analytically evaluate frame-temperature levels. As a consequence, even at the early prototype stage, thermal behavior of the frames was known under all limitations of design specifications and temperature environments (0° to 49°C office). As designs evolved and frames became available, thermal evaluation shifted from analytical to actual experimental testing under all environmental conditions. All frames were heat tested at elevated temperature and, when necessary, design improvements were implemented to assure proper functional performance. In all known cases, the operating device temperatures are consistent with noise margin, speed, and reliability requirements. Finally, the entire 1A Processor was heat tested as a system at the first No. 4 ESS installation. The system operated satisfactorily at the elevated temperature and the test proved to be effective in isolating a few remaining manufacturing and design problems.

6.4 Design intervals

The successful development of 1A Processor frames was highly dependent on the ability to rapidly proceed from a circuit schematic to the actual physical hardware. This feat was accomplished by the extensive employment of computer design aids, and the standardization of physical design topology. Since the basic frame apparatus, such as connectors, circuit-pack housings, mounting brackets, multilayer board backplanes, etc., were common among all frames, the availability was immediate and not dependent upon a design peculiar to that frame. Critical design intervals were limited to ceramic and printed epoxy circuit packs, along with the backplane-wiring interconnection system. Initially, digital, thin-film ceramic pack turnaround from circuit to complete pack was 13 weeks. However, as the technology matured the interval was reduced to four weeks. Comparable intervals for printed-circuit packs were 12 weeks and 5 weeks, respectively. Intervals for generation of backplane wiring information and actual execution were a function of number of wires and wire complexity, but usually were limited to five weeks. While these intervals reflect new designs, changes to existing circuits were accomplished in much shorter time intervals when old hardware could be salvaged. As a result of these intervals, the development process was able to proceed in a manner consistent with program needs.

VII. ACKNOWLEDGMENTS

The work reported in this article represents the valued contributions of a large number of people in a wide variety of disciplines throughout Bell Laboratories and Western Electric. The authors wish to thank R. A. Pedersen and R. A. Reed for their assistance in the preparation of the sections on the CDI technology, the DTTL gate, and the electrical performance of 1A technology.

REFERENCES

1. E. A. Irland and U. K. Stagg, "New Developments in Suburban and Rural ESS (No. 2 and No. 3 ESS)," International Switching Symposium, Munich, Germany, September 9–13, 1974, p. 512R.
2. E. G. Walsh and G. Haugk, "The Development and Application of Remnant Reed Contacts in Electronic Switching System," International Switching Symposium, Boston, Mass., June 1972, Communications Society of IEEE.
3. "No. 1 Electronic Switching System," B.S.T.J., *43*, No. 5 (September 1964), pp. 1831–2609.
4. B. T. Murphy and V. J. Glinski, "Transistor-Transistor Logic with High Packing Density and Optimum Performance at High Inverse Gain," IEEE J. Solid State Circuits, *SC-3* (September 1968), pp. 261–276.
5. B. T. Murphy, V. J. Glinski, P. A. Gary, and R. A. Pedersen, "Collector Diffusion Isolated Integrated Circuits," Proc. IEEE, *57* (September 1969), pp. 1523–1528.
6. P. A. Gary, R. A. Pedersen, B. H. Soloway, and R. A. Reed, "Design of High-Performance TTL Employing CDI Component Structures," IEEE J. Solid State Circuits, *SC-5*, No. 5 (October 1970).
7. M. P. Lepselter, "Beam Lead Technology," B.S.T.J., *45* (February 1966), pp. 233–253.
8. G. H. Schneer, W. Van Gelder, V. E. Hauser, and P. F. Schmidt, "A Metal-Insulator-Silicon Junction Seal," IEEE Trans. Electron Dev., *ED-15* (May 1968), pp. 290–293.
9. H. A. Waggener, R. C. Kragness, and A. L. Tyler, "Anisotropic Etching for Forming Isolated Slots in Silicon Beam Leaded Integrated Circuits," presented at the International Electron Device Meeting, Washington, D.C., 1967.
10. W. H. Orr and T. R. Robillard, "Thin Film Hybrid Integrated Circuits for Communication Systems" IEEE Trans. Parts Hybrids Packag., *PHP-8*, No. 2 (June 1972), pp. 51–58.
11. H. Basseches and A. Pfahl, "Crossovers for Interconnections on Substrates," Proceedings of the Electronic Components Conference, April 1969, p. 78.
12. J. A. Burns, "Bonded Crossovers for Thin Film Circuits," IEEE Trans. Parts Hybrids Packag., *PHP-8*, No. 2 (June 1972), pp. 35–38.
13. M. L. White, "Encapsulation of Integrated Circuits," Proc. IEEE, *57*, No. 9 (September 1969), pp. 1610–1615.
14. T. L. Bradley, "Mechanical Design and Characterization of a High Reliability, High Density Connector for Electronic Switching Systems," Eighth Annual Connector Symposium Proceedings, Cherry Hill, N.J., October 22 & 23, 1975, pp. 67–77.
15. R. Ostapiak and H. J. Luer, "No. 4 ESS: System Power," B.S.T.J., *56* (1977).

Copyright © 1977 American Telephone and Telegraph Company
The Bell System Technical Journal
Vol. 56, No. 2, February 1977
Printed in U.S.A.

1A Processor:

Control, Administrative, and Utility Software

By G. F. CLEMENT, P. S. FUSS, R. J. GRIFFITH, R. C. LEE, and R. D. ROYER

(Manuscript received July 16, 1976)

System software provided with the 1A Processor simplifies the task of using input/output devices, manages a portion of system time and memory resources, provides control over program interfaces, and provides an interactive facility that permits examination of system behavior. The impact of this software has been to simplify initial development of switching systems using the 1A Processor, to facilitate software maintenance, and to provide flexibility in adding system features. This article describes the capability and organization of the 1A Processor control, administrative, and utility software.

I. INTRODUCTION

Control, administrative, and utility software has been developed for the 1A Processor that simplifies the task of using input/output devices, managing system time and memory resources, providing control over the many program interfaces that exist, and finally providing an interactive facility that permits analysis of system behavior. Figure 1 provides a high-level block diagram of the manner in which the control, administrative, and utility features interact and fit into the software system.

A maintenance control structure has been provided with the 1A Processor to control system time and memory resources used by maintenance programs. These programs are executed concurrently with call processing. This facility allocates the resources associated with the maintenance function based on a priority arrangement, provides initialization between disjoint time intervals of program execution, and also provides a collection of general-purpose control functions. A standard interface provides access to these functions as well as to other control

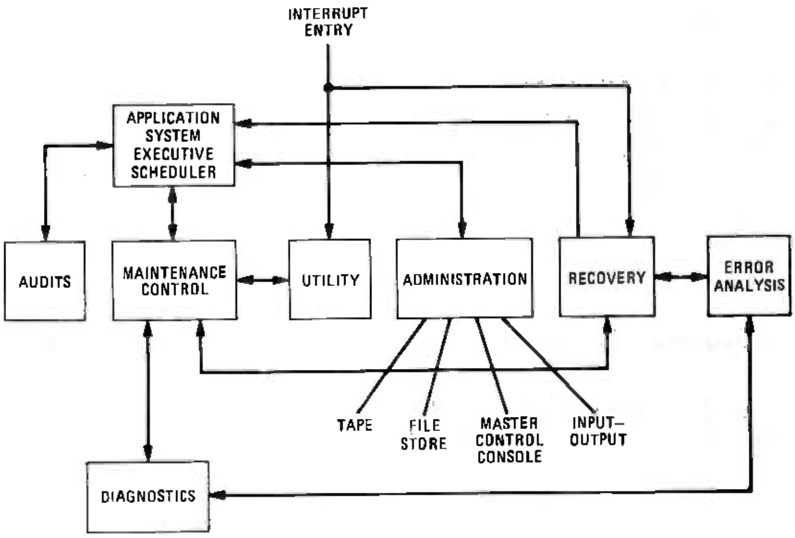


Fig. 1—1A Processor software.

features in order to simplify program design and improve reliability. A paging control feature manages access to the programs, which are stored in lower-cost disk memory. The paging controller manages the interaction between paged routines to insure that they are resident in core before they receive control. It also performs run-time link-editing functions to prepare newly loaded programs for execution.

The administrative features of the system were created to simplify the various programming tasks and to insure the most reliable design. The design relies on the use of standard interfaces that allow the using programs to be isolated from the details of the software or hardware functions being performed. Features have been provided for interfacing with input/output devices, the file store, tape subsystem, and the master control console.

Finally, utility features have been provided to allow extensive on-site analysis of system behavior. The features provided are similar to those in powerful laboratory utility systems and simulators. This is the first time a comprehensive set of utility functions has been included in a Bell System electronic switching system.

II. CONTROL SOFTWARE

2.1 Control software interfaces

The 1A Processor control programs allocate time and memory to programs called clients. The clients include: (*i*) maintenance programs that perform actions deferred from interrupt-level maintenance actions,[1] e.g., if, on interrupt-level, a program store is removed from service, a deferred action would be to diagnose the store; (*ii*) programs that execute various actions requested by the craft force, e.g., requests to reconfigure or diagnose the system or to perform the immediate utility functions described below; (*iii*) routine maintenance-related programs, such as processor error analysis or audit programs that verify the integrity of data structures; and (*iv*) library programs. Library programs are an open-ended set of functions that are added to the system on a temporary or special-use basis. Typical examples are programs to perform special hardware tests, such as those required when additional units are being added to an existing office. Many of the clients are common to all applications of the 1A Processor, e.g., a call-store diagnostic program. Clients may, however, be specific to a given application such as the program to diagnose the switching equipment within No. 4 ESS.

The overall relationship of the common control software to other portions of the system is depicted in Fig. 2. The main executive program cyclically dispenses control to a set of programs. Each of these programs executes for a period of time, or "segment," that is limited by programming standards (enforced by run-time hardware checks) to a specified maximum duration. A typical program would require many segments

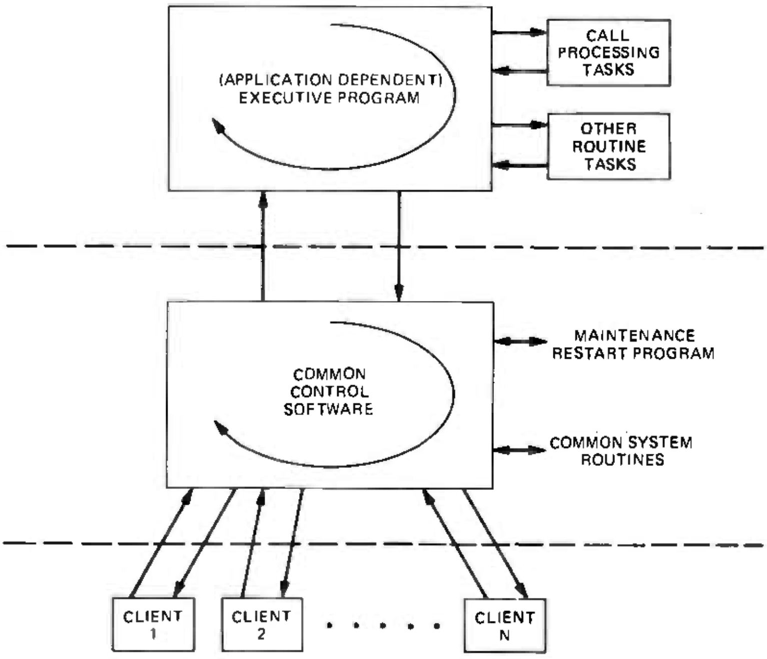


Fig. 2—Control software interfaces.

to run to completion. When the common control software receives control, it activates client programs in sequence (giving each a "segment" of time) until it has exhausted all the real time allocated to it or until no client requires a time segment. The control software provides initialization, execution, and cleanup routines to assist the client programs. Examples of these include routines that transfer nonresident client programs from magnetic tape or disk to core memory, routines that allow a client to suspend execution until a specified condition has been met, and routines to request the execution of another client.

2.2 Control software algorithms

2.2.1 Client scheduling

The first task of the control software is to determine which client, of those that are in competition for system resources, is to be scheduled for execution. If memory resources are available and requests for client execution have been received, the control software schedules the highest priority client that can be executed concurrently with all other clients that are already scheduled. Subsequent paragraphs will elaborate upon client scheduling.

Clients are identified uniquely by three attributes: class, job type, and job number. The class attribute groups clients that are generally similar. Clients from different classes can always be executed concurrently while clients from the same class may interfere with one another. In addition, the class attribute is the first determinant of client priority, since the scheduling algorithm seeks clients from one class first and then from the next class in order of classes as defined at program assembly. As an example of classes, all maintenance programs form one class, the utility programs are a second class, and library programs are a third. The job type within a class is a means of further establishing client priority since, within classes, clients are sought in order of job type. Two sample job types would be manually requested diagnostics and routine exercise programs whose execution is triggered at prespecified times. In this instance, the former job type would normally be higher priority; i.e., a manually requested diagnostic would be executed before a routine exercise. The relative priority of job types is predetermined at assembly time. The final determinant of priority is the job number. Within job types, the control software scans for "job requests" in order of job number. Again, this priority is fixed at assembly time. This would cause a central control diagnostic, say, to be scheduled before a program store diagnostic.

The remaining condition to be satisfied by the scheduling algorithm is to check for potential conflicts with clients that have already been scheduled. The potential for conflict exists because the maintenance programs alter the system configuration when first receiving control and assume that the configuration is unchanged from segment to segment. The overhead of completely reestablishing that same configuration for each time segment would be prohibitive. For example, a request to diagnose an unduplicated program store causes the maintenance program to copy the contents of that store into a spare program store. The spare store is then put in service in place of the store to be diagnosed. Copying the contents of a program store takes many segments. A potential conflict would occur in this example if a program store bus diagnostic were to be executed concurrently, for it would attempt to establish various store-bus configurations conflicting with those needed for the store diagnostic. The 1A Processor resolves this conflict by establishing blocking rules that prevent the concurrent execution of clients with conflicting unit configuration requirements. The structure of the rules, i.e., which units are affected by each client and which unit types are interfering, is established at assembly time. The scheduling algorithm checks for this blockage before activating a client that has been requested. A client is not scheduled while it is blocked.

2.2.2 Client supervision

The second basic task of the control software is to furnish run-time support to the clients that are scheduled for execution. These services include initiation and termination of execution at both the job and segment level, providing clients access to common service routines, and loading and linking pages for client programs that are not core resident.

Job initiation consists of assigning temporary memory to a client, loading that memory with input data received with the request to activate the client, loading the first page of the client program into core memory if necessary, and, finally, transferring to the start address of the client.

During client execution, control software service routines allow the client to specify special conditions that are to be established before it is given another execution segment. These conditions include special register settings, such as interrupt inhibits (which are restored to normal outside the client's segment), completion of output or disk transfer operations, or the passage of timed intervals. In addition, control routines intercept attempts by system routines to return control to clients that have activated them. This is required since the control program may have suspended or aborted client execution between the client's call to a system routine and the response of that routine. For example, a client may initiate a disk transfer. When the disk operation is complete, the client would receive control from the administrative program in order to record the completion of the operation. If, in the mean time, the client execution has been prematurely terminated because of a system error, then this attempted transfer from the administrative program would cause an improper flow of control. Therefore, the control program provides an interface between system programs and a client, intercepting all such responses and passing them to the client only if it is still in execution.

Client supervision also involves providing pages of nonresident programs as required. This is accomplished by intercepting program transfers off the currently loaded page, temporarily suspending client execution, transferring the necessary client program elements from disk to core, linking the elements together to form a new complete page, and reactivating the client.

Finally, client supervision requires providing a mechanism for client termination. Normally a client notifies the control program when it has completed execution. Under these circumstances, the control program simply releases resources that had been assigned to the client and restores hardware control registers and software status indicators to their normal values. A more complex situation arises when an abnormal condition occurs, such as a processor maintenance interrupt. This requires the control program to analyze the situation to determine whether the client either caused the abnormality or could have been adversely affected by the disruption. In either case, the client is terminated. On the one hand, this protects the system from potential further disruption. On the other, it guards against invalid results from clients. Before termination, the client is given control *once* at an abnormal termination routine that it has provided to handle the abort. The client attempts to restore the system to a normal state if it has established abnormal conditions. Should this "abort processing" by the client cause further disruption, the control program completely terminates client execution and relies upon recovery and integrity programs to restore normal system operation.

2.3 Control program organization

The control program consists of seven major sections, as shown in Fig. 3. Each of these program sections consists of reentrant, table driven code.

This structure has provided efficient and highly reliable operation. This is important, of course, since inefficiency in this control program could waste a significant fraction of the processor capacity. Furthermore, only the data tables need be changed to alter the client classes, job types, rules for analysis of abnormal situations that require early client termination, and most other characteristics described above. This means

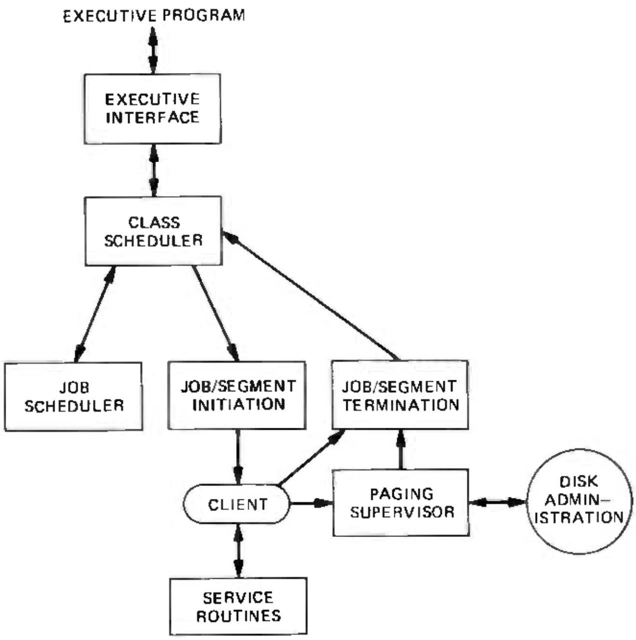


Fig. 3—Control program organization.

that relatively simple and easily identified changes allow this software to be adapted for rather different applications of the processor.

The basic data tables are illustrated in Fig. 4. There is one class table, one job-type table for each class, one set of job-request flags for each job type, one job-directory table for each job type, and one program-directory table. An extensive set of assembly language macros facilitates building and maintaining these tables. For paged programs, additional macros and pseudo-ops allow programmers to describe the program structure while writing software with minimal consideration of paging operations.

III. ADMINISTRATIVE PROGRAMS

The 1A Processor administrative programs provide interfaces between the software and four types of hardware: the file stores, tape units, input/output devices, and the master control console. The hardware dependent code is concentrated in these programs. A macro is provided for each function performed by an administrative program to produce a standardized interface between programs. Each macro is expanded by the assembler into instructions to provide the data needed for the function and a transfer into the administrative program routine to perform the function. For operations that require significant amounts

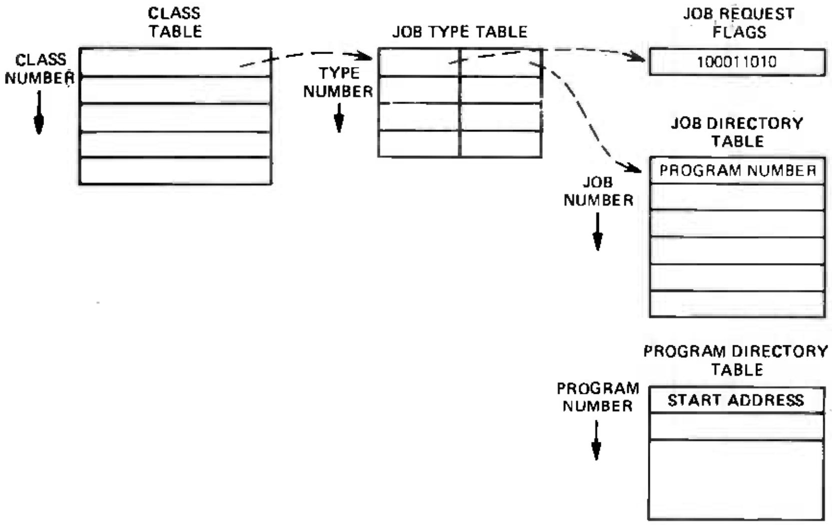


Fig. 4—Control program data tables.

of time for completion, such as data transfer to a file store, two exchanges of control occur between the using program and the administrative program. The first of these is a transfer of control from the using program to the administrative program. This is a request by the using program. Control is returned by the administrative program as soon as it has noted the request and associated control information and started the hardware action. The calling program is then able to continue processing if it does not depend upon completion of the request. Upon completion of the hardware action, the administrative program is called and transfers control to the using program at an address provided with the request. The using program records the completion of the request and then returns control to the administrative program.

3.1 File store administration

As described in Ref. 2 data are transferred between disk memory and core memory under control of the file store. The disk administration program provides the interface between the using programs and the duplicated file stores. The functions that it provides are: (*i*) read either disk, (*ii*) write both disks, (*iii*) write a specific disk, and (*iv*) read a specific disk. The first and second of these are used most frequently. The write or read operations that specify a particular disk are used by programs with duplicated data bases that are altered one copy at a time. This preserves the internal consistency of one copy while the other copy is undergoing a time-consuming alteration.

The administrative program tests the validity of a disk request and its execution. Each block of data on disk has an identifier associated with it and recorded within the block. A request to read or write a data block indicates the identifier that the using program has associated with the data block. If that identifier does not agree with the one recorded on the disk, an error condition is detected by the administrative program for write requests or by the file store for read requests. In either case, the administrative program notifies the using program that the requested operation cannot be completed. The administrative program also checks for other error indicators in the file-store hardware. The program attempts to repeat the operation if errors are indicated. If multiple attempts fail, the administrative program reports this failure to the using program. If the request has been completed without error, this is reported.

3.2 Tape administration

The tape-administration program provides the interface between using programs and the tape-unit hardware. The administrative program provides functions: (*i*) secure a tape for a using program, (*ii*) read a tape record, (*iii*) write a tape record, (*iv*) write an end of file, (*v*) position the tape as required, e.g., rewind to the beginning of the tape, move to the beginning of the next file, and locate a specific file, and (*vi*) release a tape. When a using program secures or releases a tape, the administrative program issues directives to the craftperson to mount or dismount a tape if necessary. It also receives the craftperson's input message that indicates when the required action has been completed.

The tape-administration program tests the validity of requested operations by comparing the tape specification contained in the using program's request to information obtained from a header label that is recorded at the beginning of each tape. It also repeats operations if error indicators are set by the tape hardware. Finally, the tape-administration program assures highly reliable recording of telephone billing data by automatically switching to a previously secured standby tape if a failure or end-of-tape occurs.

3.3 Input/output administration

The input/output administration program provides the interface between using programs and the input/output hardware and terminals. The 1A Processor is equipped with many input/output channels. There are also many programs that receive inputs or generate outputs. There is, however, no fixed association of programs and input/output channels. Many programs may be simultaneously directing output to the same logical channel. On the other hand, each member of a series of messages from a craftperson may be destined for a different program. Moreover, the format in which data are represented within programs is not convenient for manual interpretation and the converse is true also. Therefore, the input/output administration program must perform several functions in addition to providing an interface between using programs and the input/output hardware. On input, the program parses the input message to create a more readily usable format for user programs and determines which using program is to receive the message. On output, the program translates the data to a format that is more conveniently read by a craftperson and combines independent message streams for each channel. Ancillary functions allow messages to be rerouted to circumvent hardware malfunctions and allow outputs for a given channel to be monitored by other channels. The latter function allows the craft force flexibility in shifting allocation of responsibilities between office personnel.

The input/output hardware and software are illustrated in Fig. 5. Note that format conversion and input message destination are governed by data tables called "catalogs." This means that changes in these may be introduced without changes to the input/output administration program. Output catalogs may be included in the using program and, thus, may be entirely under its control. Since the input catalogs are needed to determine the destination for input messages, they must all be available to the administration program. Using programs may, however, add or delete input catalogs at run-time.

The messages associated with the 1A Processor conform to standards that are common to all Bell System electronic switching systems. The typical input format consists of a verb, noun, and optional modifier. Each input is a directive to the system to perform an action (specified by the verb) on a portion of the system (specified by the noun) with options (specified by the modifier). The verb and noun together determine the program that will receive the message. Data within the message is conveyed by key word and argument rather than by position. As an illustration, the following message directs the tape-administration program to allow (abbreviated as ALW) the use of tape-unit controller 2 (TUC 2) in the read-only mode (RO).

ALW:TUC 2:RO

Output messages follow similar formats.

3.4 Master control administration

The master control console administration program interfaces with the hardware master control console and processor peripheral interface.[2] One major function that it performs is to read keys and operate lamps on the master control console for other programs. It does this through

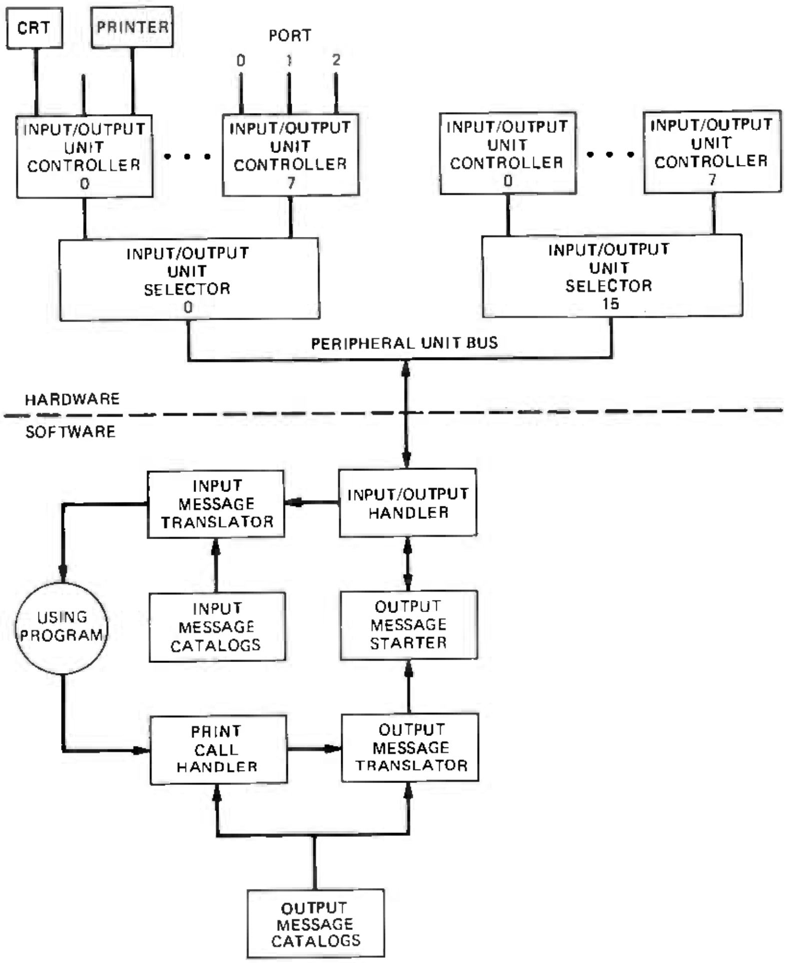

CRT
PRINTER
PORT
0
1
2
INPUT/OUTPUT UNIT CONTROLLER 0
INPUT/OUTPUT UNIT CONTROLLER 7
INPUT/OUTPUT UNIT CONTROLLER 0
INPUT/OUTPUT UNIT CONTROLLER 7
INPUT/OUTPUT UNIT SELECTOR 0
INPUT/OUTPUT UNIT SELECTOR 15
PERIPHERAL UNIT BUS
HARDWARE
SOFTWARE
INPUT MESSAGE TRANSLATOR
INPUT/OUTPUT HANDLER
USING PROGRAM
INPUT MESSAGE CATALOGS
OUTPUT MESSAGE STARTER
PRINT CALL HANDLER
OUTPUT MESSAGE TRANSLATOR
OUTPUT MESSAGE CATALOGS

Fig. 5—Input / output system.

a matrix of scanner and signal distributor points that is connected to the console. The program maintains a call-store image of this matrix to be used as backup in case of hardware failure. Another major program function is a periodic scan, every 1 second, of the console keys. Any changes are updated in the call-store image and reported to the appropriate client program. A third function is processing manual reports communicated via the power-control switches connected to the processor peripheral interface matrix. Every second the program scans the switches and reports any change to the appropriate fault-recovery program, which will then take an action such as removing a unit from service in response to a key operation. The program also changes the state of the power-switch lamps to inform the craftsperson of the system's action.

IV. UTILITY SOFTWARE

4.1 Utility system objectives

The current systems using the 1A Processor are large in terms of both program size and data base requirements (about one million program and data words.) A very large effort has been required to debug all of the software features and to gather and output various kinds of data that are of interest. Since the program and data base are so large and new features are regularly added to the program, there will probably always be some latent errors in the system program. Because of this, a sophisticated utility capability has been included with the 1A Processor to aid in testing the initial and updated programs and, in anticipation of future needs, to sample data to analyze system behavior. A major objective of the utility features has been to safely provide full access to data in the system.

There are many potential uses for such utility features in an operational telephone switching machine. For example, the office craft force may wish to obtain information pertinent to determining why a specific software procedure is failing. Alternatively, information concerning the use of a particular trunk or trunk group in an office or information about the way in which the office data base is accessed or changed may point to the solution of a problem. The utility system allows these functions to be conveniently and safely inserted on a temporary basis into the operational program, and to be removed when the necessary information has been gathered.

The utility programs have been designed in a modular fashion for ease of modification and to limit the effect of the addition of new facilities or functions on previous features. Special hardware in the 1A central control facilitates efficient implementation of the utility system by providing the mechanisms for the utility software to receive and return control without disrupting the normal flow of control within the system.

4.2 Utility features

The capabilities of the utility package are discussed below.

4.2.1 Conditional utility functions

The WHEN function of the utility system permits the interruption of program flow and the transfer of control to utility software for the execution of the specified utility operations when specific conditions have been met. The conditions permitted for the WHEN function are:

(*i*) When a specific program address is reached.
(*ii*) When a specific memory location is read or written.

The first condition is implemented by using a special instruction that is loaded by the utility software in place of the instruction normally resident at the program address. This instruction saves the system registers and transfers control to the utility software. The second condition uses the matching circuitry of the central control to compare data addresses to the specified value. A successful match causes the central control to save its registers and transfer to the utility software.

Additional control can be maintained over the execution of utility functions in a WHEN clause by the use of the IF and ELSE functions. The IF function controls the utility operation on the basis of the evaluation of a logical expression included in the command. The following features are permitted in the evaluation:

(*i*) One of six operators may be specified for comparing two memory or register locations.
(*ii*) Several levels of indirect addressing may be specified, with or without indexing, for both sides of the expression.
(*iii*) The evaluation may take into account from one to all bits in the registers or memory locations being evaluated by the expression.

If the expression is evaluated as true, the utility commands that follow the IF clause are performed. However, if the expression is evaluated as false, no functions are performed unless the ELSE command was part of the original message. If ELSE is specified, and the expression is false, the utility commands following the ELSE are executed. A number of complex conditional expressions can be active in the system simultaneously.

4.2.2 Immediate utility functions

The capability is provided in the utility software to permit almost all addressable locations in the system to be displayed on output devices, to be initialized to a specified value, or to be moved from one addressable location to essentially any of the memory systems provided by the 1A Processor. In addition, the capability is provided to freeze large amounts of data on the occurrence of a particular event or to provide control of oscilloscope operation being performed by the office craft people. Most of these features can be performed in direct response to input messages as immediate utility functions or in conjunction with the conditional features of the system described in the preceding section. Each of the above functions are described in more detail in the following paragraphs.

The DUMP function provides the capability of displaying portions of the following addressable 1A Processor locations on an output device:

(*i*) Call and program-store memory.
(*ii*) File-store memory.
(*iii*) Standard label tapes.
(*iv*) Special memory locations provided for the utility user.
(*v*) Central control internal registers.

The DUMP feature permits memory contents to be printed on a system teletypewriter. It is also possible to display data on the master control console. The determination of the address at which the DUMP is to begin can be influenced by both indirectness and indexing options. The capability is provided for displaying from one to all bits in the locations being dumped. Output may be in either binary or decimal format. Through the use of the "DUMP on interrupt" command, data may be selectively displayed on any interrupt level. This selection may be refined down to a dump on any particular interrupt source.

The LOAD function provides the capability of initializing portions of the following addressable 1A Processor locations:

(*i*) Call and program store memory.
(*ii*) Special memory locations provided for the utility user.

The LOAD feature permits up to 128 locations to be initialized. The same address determination and formating features that are provided by the DUMP command are available on LOAD.

The COPY function provides the ability to move data from one storage medium in the 1A Processor to another without affecting the data at the source location. The same address determination and formating features provided for the DUMP function are also available on COPY.

The FREEZE function provides the ability to save the contents of one or more program or call stores (65,536 26-bit words) for later analysis. The contents of the store may be frozen at the time of a program interrupt, before memory is initialized as part of system recovery, or on other occurrences of interest. This function is implemented by maintaining a duplicate memory block for the area to be frozen and, when the event of interest occurs, inhibiting changes to the duplicate memory. This data may be selectively displayed on a teletypewriter or copied onto tape for later analysis.

The SYNC function provides the ability to produce a sync pulse at a coaxial connector on the central control when a desired event occurs. The SYNC command is triggered by the central control match circuit and is used in conjunction with the conditional utility features described previously. It is expected that the SYNC pulse will be essential for hardware repair procedures using an oscilloscope.

4.2.3 Control functions

A data set capability has been designed into the generic utility package to permit message sequences to be created, saved, and manipulated. This procedure permits any messages that can normally be input on the teletypewriter to be saved on disk memory under a name specified by the user. The named sequence of messages can be edited or can be input to the system with one command. This feature permits the craft force to build a library of convenient functions for use in routine procedures and reduces the likelihood of errors due to improper messages.

The utility software has a feature that enables field-operations personnel to "overwrite" or change the program of the 1A Processor. These changes are introduced without disruption of the system, while it is in operation. The altered data is restored to its initial value automatically if a system malfunction occurs. The craft force is allowed to make the change permanent after a test interval has been completed. A set of control features is also provided to permit the craft force to control the initiation and disabling of conditional utility functions.

The utility features included with the 1A Processor provide a high degree of capability for determining how and why the system operates as it does. In some cases, this flexibility, if carelessly used, makes it possible to disrupt system operation. To minimize this possibility, a number of safeguards have been built into the utility functions. For example, addresses used for utility actions must be in predefined ranges specified on the input request. Software controls also insure that utility functions cannot take excessive amounts of real time from the system. If too much time is taken, the active utility functions are automatically deactivated. If invalid program actions are detected, software recovery[1] removes the utility functions. Finally, a mechanism is provided so that utilities can be quickly disabled through manually initiated interrupt at the master control console.

4.2.4 Off-line functions

The off-line features of the utility system provide the 1A Processor with the ability to remove redundant hardware units from normal system operation and to use them to build an off-line system. This off-line system can then be manipulated from the active system using the special features described below as well as the utility conditionals and commands described previously. The testing of this configuration does not interfere with normal activity of the system and permits utility functions to be performed with negligible real-time expenditure from the on-line processing hardware. If recovery from a fault in the active system requires use of a unit in the off-line system, this unit is automatically returned to service. The off-line facilities are designed to be used for the isolation of difficult hardware and software problems that cannot be solved by normal maintenance facilities.[1] The following functions are provided for the off-line configuration:

The start and stop off-line CC functions are the basic control of the off-line CC. Program execution begins with the instruction that is currently contained in internal CC registers and stops upon command from the active CC without disrupting internal CC registers.

The transfer function causes the off-line CC to transfer program control to a specified address. A return option can be specified that later permits control to be returned to the point after the transfer. The return function causes control to be passed to the instruction following the last executed transfer function, which specified a return option. The loop function causes the program in the off-line system to be modified so that a loop is created and control is passed to the top of the loop.

4.3 Utility examples

The following example is provided to briefly illustrate the functions of the utility system provided with the 1A Processor:

```
WHEN:ADR 14000000
IF:ADR 1234, SIZE 6, DISP 4; EQ; 45
DUMP:ADR 1235
ELSE:
LOAD:ADR 1237:25
```

After the instruction at address 14000000 has been executed, the contents of address 1234 is tested. If the 6-bit field (SIZE 6) that is displaced 4 bit positions (DISP 4) from the right-hand end is equal to 45, the contents of address 1235 will be printed. Otherwise, the contents of address 1237 will be set to 25.

The flow of operations that would occur to process this utility request is shown in Fig. 6. Briefly, after it is determined that the input message is a request for a utility operation, any currently active conditional utility functions are deactivated to release memory for the parsing routines that are brought from disk to error-check the message syntax and to build the necessary utility data structures. For this example, if the syntax of the input message was correct, any previously active conditions would be reactivated. The new conditional command would not itself be activated until a specific control request was received by the utility system.

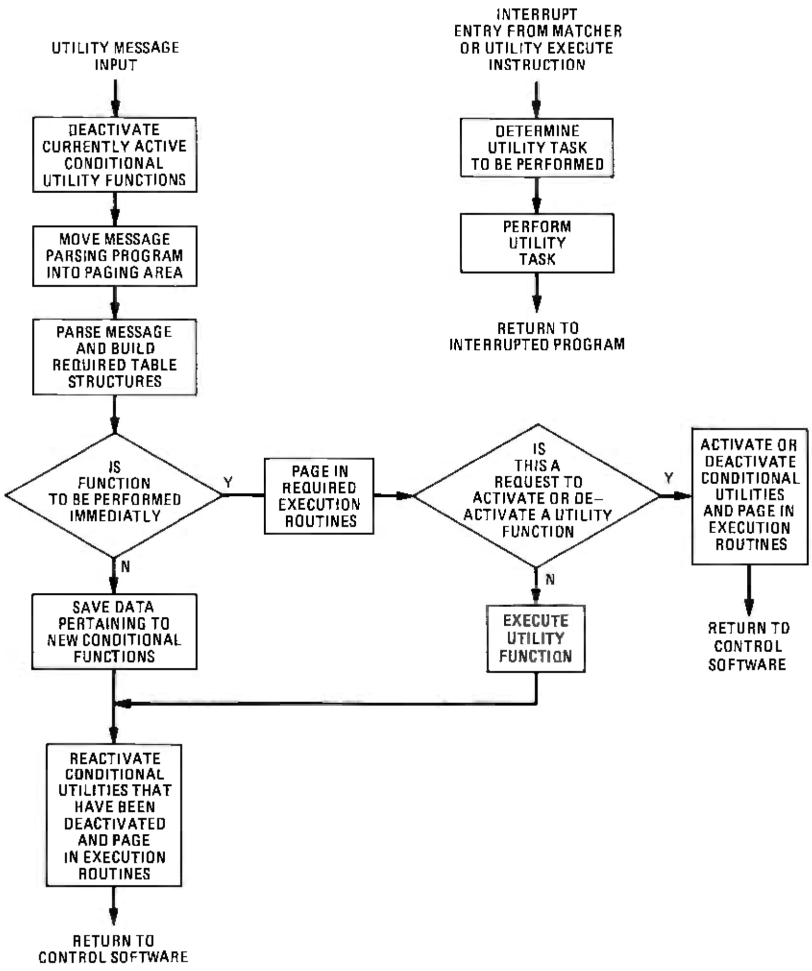

UTILITY MESSAGE INPUT
DEACTIVATE CURRENTLY ACTIVE CONDITIONAL UTILITY FUNCTIONS
MOVE MESSAGE PARSING PROGRAM INTO PAGING AREA
PARSE MESSAGE AND BUILD REQUIRED TABLE STRUCTURES
IS FUNCTION TO BE PERFORMED IMMEDIATLY
Y
N
PAGE IN REQUIRED EXECUTION ROUTINES
IS THIS A REQUEST TO ACTIVATE OR DE-ACTIVATE A UTILITY FUNCTION
Y
N
ACTIVATE OR DEACTIVATE CONDITIONAL UTILITIES AND PAGE IN EXECUTION ROUTINES
RETURN TO CONTROL SOFTWARE
EXECUTE UTILITY FUNCTION
SAVE DATA PERTAINING TO NEW CONDITIONAL FUNCTIONS
REACTIVATE CONDITIONAL UTILITIES THAT HAVE BEEN DEACTIVATED AND PAGE IN EXECUTION ROUTINES
RETURN TO CONTROL SOFTWARE
INTERRUPT ENTRY FROM MATCHER OR UTILITY EXECUTE INSTRUCTION
DETERMINE UTILITY TASK TO BE PERFORMED
PERFORM UTILITY TASK
RETURN TO INTERRUPTED PROGRAM

Fig. 6—Basic program flow of utility software.

V. CONCLUSION

The control, administrative, and utility software of the 1A Processor has improved electronic switching system development by centralizing and standardizing functions common to multiple applications. This has simplified initial development and debugging of system software. The effort required for software maintenance has been reduced. It has provided flexibility for the addition of system features.

The software provides a balance between specialized functions required for extremely reliable long-term operation and general-purpose capability. For example, the control software protects the system from disruptions due to interactions between programs that could conflict, but it still permits concurrent execution of independent programs. As an example, within the administrative area, the disk-file system checks a data-identification field before completing disk reads or writes in order to prevent data mutilation or use of erroneous data. Since checks are performed by hardware or by administrative software, user programs bear only a portion of the responsibility to assure system reliability. Finally, the utility system provides a safe means by which the craft force may obtain special outputs or modify the system.

VI. ACKNOWLEDGMENTS

The work described in this article was accomplished through the contributions of many people at Bell Laboratories. The authors would like especially to acknowledge the assistance of E. P. Schan in the preparation of this article.

REFERENCES

1. P. W. Bowman et al, "1A Processor: Maintenance Software," B.S.T.J., this issue, pp. 255–287.
2. C. F. Ault et al., "1A Processor: Memory Systems," B.S.T.J., this issue, pp. 181–205.
3. A. H. Budlong et al., "1A Processor: Control System," B.S.T.J., this issue, pp. 135–179.

Copyright © 1977 American Telephone and Telegraph Company
THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 56, No. 2, February 1977
Printed in U.S.A.

1A Processor:

Maintenance Software

By P. W. BOWMAN, M. R. DUBMAN, F. M. GOETZ,
R. F. KRANZMANN, E. H. STREDDE, and R. J. WATTERS

(Manuscript received July 16, 1976)

Comprehensive maintenance software is required to meet the system reliability objective of less than an average of 2 minutes per year of outage from all causes. The function and interrelationship of the four basic maintenance programs (fault recognition and recovery, diagnosis, trouble location, and error analysis) are detailed here. Results of extensive laboratory testing and early field experience indicate that the maintenance objectives will be achieved despite the size and complexity of the 1A Processor.

I. INTRODUCTION

Like the processors for the earlier electronic switching systems (e.g., No. 1 ESS, No. 2 ESS), the 1A Processor depends on integrating maintenance software with the hardware to (*i*) quickly recognize a fault condition, (*ii*) isolate and configure around the faulty subsystem, (*iii*) diagnose the faulty unit without interfering with normal processor functions, and (*iv*) assist the maintenance personnel in locating and correcting the fault. The system usually detects a fault and reconfigures itself within a few milliseconds without affecting calls being switched through the office. The 1A Processor writable program stores and high-throughput disk and tape subsystems, including direct memory access, have introduced both new maintenance problems and new avenues for their solution. Although these bulk storage systems have complicated the process of fault recognition and diagnosis, they have also provided the key to vastly improved trouble location and error analysis.

Each aspect of maintenance software is designed to minimize the likelihood of system outages. As discussed in Ref. 1, a reliability objective

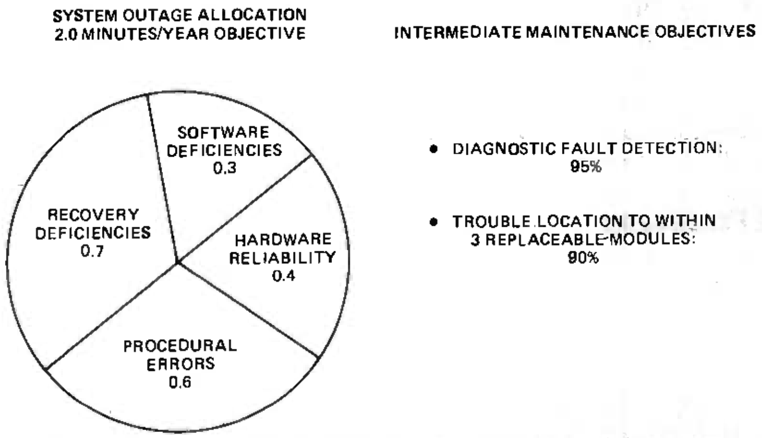


Fig. 1—System outage allocation and intermediate maintenance objectives.

is to keep the average accumulated downtime of 1A Processors at no more than 2.0 minutes per year. To achieve this objective, the probable causes of system outages are assigned to one of the four general categories shown in Fig. 1 and allocated a reasonable portion of the total system downtime. This allows the intermediate reliability objectives discussed in the following to be set for some components of maintenance software.

"Software deficiencies" can cause outages by improper system cycling. To minimize this source of downtime, overall program cycling is continually monitored, data integrity is checked using extensive auditing procedures, and thorough system integration tests are performed after program changes are introduced. Outages resulting from software deficiencies are not expected to average more than 0.3 minute of downtime per year.

When outages occur because a full complement of working units is not available to establish a system configuration, they are included in the "hardware reliability" category and are allocated an average 0.4 minute of downtime per year. Two maintenance software components, diagnosis and trouble location, bear strongly on hardware availability. As a result, intermediate maintenance goals have been set. Experience shows that, while it is costly in both hardware and software to develop diagnostic test programs capable of detecting every fault that can occur in a unit, it is generally feasible to develop sufficient maintenance access and diagnostic tests so that 95 percent of the faults (as measured using simulation results) can be detected. Therefore, the minimal level of fault detection required for each individual diagnostic program has been set at 95 percent.

Fault-detection capability is by far the most important diagnostic property since the processor assumes a unit to be fault free if all diagnostic tests pass. However, it is not sufficient to detect the presence of a fault; the average repair time of units must generally be less than 2.0 hours to meet the previously stated hardware reliability objective. With time-consuming repairs, the behavior of the unit might be marginal or intermittent; however, most repairs should be completed in less than half the average repair time. Hence, an intermediate maintenance objective was established; at least 90 percent of the faults should be isolated to no more than three replaceable modules by the on-line trouble-locating procedure.

"Procedural errors" are expected to account for about 0.6 minute of downtime per year. Attempts to minimize this major cause of system outage include careful design of the human interface with emphasis on documentation clarity and uniformity, reduction in the number of manual operations and translations, and defensive programming implementations.

As depicted in Fig. 1, the largest source of system outage is expected to be "recovery deficiencies." Building on the strategy implemented in No. 1 ESS, the 1A Processor fault-recognition programs are generally interrupt driven and attempt to assemble a working configuration whenever a system error or fault is detected. Setting intermediate performance objectives for this software-maintenance component is difficult because of the large number of variables involved (the system can be in almost any state when a fault occurs) and because recovery is strongly related to all other components of maintenance. For example, recovery can be either facilitated or thwarted by manual procedures and can be easily misled by incomplete diagnosis. Clever use of failure symptoms collected by the error analysis program could obviate the need of some later system-recovery actions, but there is no guarantee that all impending troubles will be identified and isolated before they can jeopardize system operation.

In summary, all four maintenance software components are strongly interrelated. The fault-recognition-and-recovery program, in response to an interrupt, makes a tentative decision about the health of a unit; it automatically requests diagnosis upon suspicion of a fault, deferring the ultimate decision to the diagnostic program. Most diagnostic failure results are used to automatically pinpoint the fault to a few replaceable modules. Problems that elude detection or isolation by any of these maintenance components will have to be manually located. To assist maintenance personnel in these situations, the error-analysis software collects and files all trouble symptoms, and retrieves them on request.

II. FAULT RECOGNITION AND RECOVERY

2.1 Subsystem redundancy

The subsystem loss objective of only 0.4 minute of downtime per year is achieved in part through subsystem redundancy. The functions implemented in some units are so critical (for example, the central control) that they require at least full duplication of these units. Reliability calculations show that redundancy greater than full duplication is not required in any subsystem. In fact, unlike No. 1 ESS, subsystems containing many units do not require full duplication to meet the hardware reliability objectives. For memory units, redundancy is influenced by the ease with which data stored in a failing unit can be regenerated. Since a program store can be reloaded from the file-store system in less than a second, a roving spare redundancy plan is sufficient for the program-store community. The call stores, however, contain both data backed up on the file stores and call-related transient data. This transient data can be regenerated only through phases of memory reinitialization that interrupt system operation for many seconds and terminate calls that are not in the talking state. Therefore, the limited-spare concept used for the program store is expanded for the call stores to include sufficient spares to provide full duplication of transient data. The file stores contain the backup data for the call/program stores and transient data that is accumulated over long time periods. Neither type of data can be regenerated easily and, therefore, full duplication of the file stores is provided. Table I summarizes the redundancy plan for each processor subsystem.

2.2 Hardware fault detection

Hardware redundancy alone is not sufficient to meet the 1A Processor reliability requirements. Rapid fault detection and reconfiguration is also needed. Fault detection is accomplished through both hardware and software checks, with the emphasis on hardware due to its inherent

Table I — 1A Processor redundancy plan

Full Duplication	Limited Spares	Duplicated Bus Access
Bus systems	Call stores (CS)*	Call stores (CS)
Central control (CC)	Input/output unit channels (IOUC)	Central control (CC)
Data unit selectors (DUS)	Program stores (PS)	Input/output unit selectors (IOUS)
File stores (FS)	Tape units	Master control console (MCC)
Input/output unit selectors (IOUS)		Program stores (PS)

* Sufficient spares to duplicate transient data.

Table II — Hardware fault detection techniques

Unit	Techniques
Central control	Matching of bus transmissions and internal operations Parity Protected address range Timing
Call program stores	Parity Operation checks (access current, etc.) Timing Acknowledgments
File stores	Parity Cyclic redundancy code Internal matching Timing Operation checks (disk speed, etc.) Acknowledgments
Data unit selectors/ tape unit controllers	Parity Cyclic redundancy code Timing Operation checks (invalid mode/command, etc.) Acknowledgments Loop-around checks
Input/output units	Internal parity Timing Operation checks (invalid command, etc.) Loop-around checks Acknowledgments

speed.[2] Since a single fault-detection technique cannot solve the problems of all subsystems, several techniques are used. Table II summarizes the 1A Processor units and the techniques used for each. A detailed description can be found in the articles describing the 1A Processor units.[3,4] The 1A Processor design was improved over that of No. 1 ESS by extending the self-checking capability of all units. Each unit employs one or more of the techniques listed in Table II to verify its own operation. Self-checking speeds up fault detection by minimizing the reliance upon timing and software checks. It also aids the fault-recovery process by providing error indications that help to isolate the faulty unit.

2.3 Software error detection

While designed primarily to detect and correct data mutilation due to translation or program errors, software error detection provides a backup for the hardware fault-detection circuits. Undetected hardware faults may lead to data mutilation or loss of program sanity. The fault-detection circuits also provide a backup for software error detection. Invalid program actions, such as out-of-range memory references, generate hardware check failures. Because error-detection techniques are interrelated, the hardware- and software-recovery philosophies are also

Table III — Maintenance interrupt structure

Function	Level	Source
System configuration	A	Manual from MCC
	B	Processor configuration, CC activity switch, CC pulse-source failure
Fault detection	C	CC mismatch
	D	CS or AU failure
	E	PS failure
	F	PU failure
Test	G	Interval timer Utility match tests
Fault detection	Maintenance	AU failure
	Interject	PU failure
	Base level	AU failure
	Maintenance	PU failure

interrelated. Software recovery is initiated when the level of maintenance activity due to invalid program operations becomes high enough to affect service. Tests and reconfigurations of the 1A Processor are initiated when software recovery is unable to resolve error-check failures through transient memory initialization.

Similar to hardware fault detection, software error detection can take on many forms. These include timing checks, error codes, in-line defensive checks, data-structure checks, and reasonableness checks based upon redundancy in the data. A detailed discussion of these can be found in Refs. 5 and 6.

2.4 Maintenance interrupt structure

When a fault is detected by a check circuit, call processing is interrupted and fault-recovery actions are initiated. This interruption can fall into one of three priority categories determined by the severity of the fault: (*i*) immediate interrupt (maintenance interrupt) if the fault is severe enough to affect program execution, (*ii*) interrupt deferred until completion of the currently active task (maintenance interject) if the problem could affect several calls or tasks, and (*iii*) interrupt deferred until detected by routinely executed base-level jobs (base-level maintenance) if the problem affects only a single call or task.

Maintenance interrupts are assigned a priority based upon the subsystem in which the fault is detected. High-priority interrupts are allowed to occur during the processing of lower-priority interrupts, but not vice versa. The only exception to this rule is that B-level interrupts, which generally indicate a loss of system sanity, can occur while processing a manually initiated A-level interrupt. Table III summarizes the 1A Processor maintenance interrupt structure.

2.5 Fault-recovery strategy

The goal of all fault-recovery programs is to recover call-processing capabilities. This is accomplished in three steps: identification of the faulty unit, isolation of that unit, and reconfiguration and initialization of spare units. While specific actions performed by the fault-recovery programs are determined by each subsystem for which the program was designed, there are several features common to the design of all 1A Processor fault-recovery programs.

The major common feature is minimizing the effect of nondeferrable maintenance activity. This generally means minimizing the execution time. A balance between fault detection and execution speed is generally achieved through a first-look strategy in which fault-detection testing is directed towards a particular unit or part of a unit based upon the circumstances in which the fault was detected. For example, the central control fault-recovery program functionally partitions the central control based upon the instruction being executed when a mismatch occurred. A subset of all fault-recovery tests is then selected based upon the partitioning.

When fault recovery involves accessing the disk files, the duration of the interrupt is determined by file-store access time and not by the test-execution time. Therefore, when the first-look checks indicate a disk-file problem, the file-store fault-recovery program terminates interrupt processing and accesses the disk file as a deferred time-shared job. The call/program-store fault-recovery programs attempt to minimize the loading of call/program stores from file store on interrupt. This is accomplished by assigning the spares to duplicate units in which transient errors are occurring and loading these stores through a deferred time-shared job. When a load on interrupt cannot be avoided, the effect upon the system is minimized by interleaving critical call processing with the load.

In the auxiliary data system, minimizing the effect upon system operation means elimination of configuration changes which require manual tape changes. Therefore, the data unit fault-recovery program will leave in service units that have configuration-sensitive faults if a configuration can be established that passes access tests.

Another major common feature of 1A Processor fault-recovery programs is the use of short-term error analysis, which improves the tolerance of the programs to intermittent faults over that achieved with No. 1 ESS programs. Short term does not imply a common time interval. Instead it refers to those error records collected and analyzed by the fault-recovery programs as opposed to those collected and analyzed by the system error-analysis program. The records indicate units in which faults or transient errors have occurred and the response of the fault-recovery programs to those faults or errors. If analysis of the records indicates that the system has not been restored to interrupt-free operation, the fault-recovery program modifies its response to the next fault or error. The next fault detected generally results in abandoning the first-look strategy and executing all fault-detection tests. The next transient error results in isolation of the unit experiencing a high error rate.

A third major common feature is the "bootstrap" strategy. Fault recovery normally consists of identifying the faulty unit and replacing it with an operational spare. If a spare is not available, the fault-recovery program executes what is referred to as a bootstrap. During a bootstrap, the previous status of all units in the subsystem upon which fault-recovery actions are being performed is ignored, the units are tested by the recovery program, and a working subsystem configuration is established using units that pass the tests. Repeated entries to a bootstrap routine in a predetermined time interval indicate a failure to configure a fault-free subsystem, perhaps due to inadequate subsystem tests. When this occurs, the fault-recovery programs combine short-term error analysis with test results to systematically isolate units on successive interrupts that result in bootstraps.

Another common feature of the 1A Processor fault-recovery programs is control of all deferrable configuration requests. All manual and diagnostic requests to modify a 1A Processor subsystem configuration are submitted to the appropriate fault-recovery program. The configuration is changed only after determination that there will not be an effect upon system operation. With one exception, this means that a unit must be isolated and replaced with a spare before it can be diagnosed. The exception occurs when data unit selectors and tape units must be diagnosed. Since the auxiliary data system fault-recovery program may leave in service units that have configuration-sensitive faults, but which are currently in an error-free configuration, it allows them to be diagnosed when not in use by the data-unit administration program.

2.6 Software audits

Each fault-recovery and administrative program includes program units designed to audit or initialize transient data. These audits are executed on a timed basis or by the application audit controller on a routine basis. The 1A Processor software package also includes two audits designed to detect and correct errors in the nontransient call/program-store and file-store data. The first of these is a routinely executed audit that verifies the data through the calculation of error codes or hash sums. The hash sums isolate errors to 1024-word blocks and identify which copy of data (call/program store, file-store copy 0, or file-store copy 1) is valid.

This copy is then used to correct those in error and to identify specific words in error.

The second audit is manually initiated when the first audit detects an error that it cannot correct. It identifies errors through a simple comparison of the data with a tape backup. The backup tapes are periodically generated in the office and may not reflect the most recent changes to the office-dependent data. Therefore, the tape audit checks all apparent errors against an internally stored list of approved changes before marking a word for correction.

2.7 Processor configuration recovery

A general 1A Processor bootstrap recovery is automatically executed when check circuits indicate loss of processing sanity or when a fault-recovery program fails to configure a working subsystem. This bootstrapping, referred to as processor configuration (PC) recovery, occurs on one of three levels corresponding to the three sets of states in the PC sequencer in the central control. In each level, a complete processor is configured by building upon the basic configuration (central control, program store, and program store bus) established by the PC sequencer. Test and configuration routines in each of the fault-recovery programs are executed as subroutines of the PC recovery program to configure each processor subsystem. The first level, corresponding to states 0 through 15 of the PC sequencer, attempts to minimize execution time by executing the call store and program store copies of the fault-recovery programs. In the second level, corresponding to states 16 through 48, all nontransient data are verified before being used during the recovery. This level begins with a hardware-initiated load of a small program from a file store. The small program initiates the verification of data through subroutines in the nontransient data audit and also initiates the execution of the fault-recovery test and configuration routines. The final level corresponds to PC sequencer states greater than 48 and is called the repeated PC. It is entered once the fault-recovery programs have unsuccessfully attempted to build a complete processor from each unique basic configuration. Recovery steps in this level are similar to those of the second level. They differ in the selection of fault recognition tests. Less stringent tests are executed in the third level in an attempt to configure a system capable of performing very basic call-processing functions.

2.8 Manual recovery

Failure of the processor-configuration recovery sequence to establish a viable processor configuration necessitates manual-recovery procedures. These are invoked through controls at the master control console.

The first manual-recovery step taken consists of establishing a basic configuration using the override-control keys and requesting the second or third level of PC recovery. The override-control keys have the advantage over the processor-configuration sequencer of being able to force a basic configuration which fault-recovery programs cannot change.

Failure to recover system sanity through the override controls may be due to mutilated nontransient data in both the call/program stores and the file stores. Therefore, the next step in manual recovery is to reload this data from tape. This type of recovery (called a system reinitialization) is begun by using manually activated sequencers to load a small bootstrap program from tape into the basic processor. The program initiates the load of the remaining data from tape and directs programs loaded with this data to configure a complete processor.

The final set of manual-recovery procedures has no counterpart in No. 1 ESS. It involves forcing the system into an emergency mode of operation in which only manually initiated tasks are executed. All other tasks including call processing are excluded. In the event of excessive call/program-store or file-store failures, this emergency mode can be entered with a minimal processor configuration that consists of a central control and only sufficient call/program stores to execute maintenance tasks. It may also be entered with a complete memory configuration in the event of peripheral faults or program problems that cause the loss of system sanity.

Manual procedures are also available to load new versions of generic programs or office data with minimal disruption to call processing. Most of this system-update procedure is time shared. It moves the data from tape to a single file-store copy. Once this copy has been fully updated, call processing is interrupted long enough to load the call/program stores from that file-store copy. The old data base remains on the mate file stores until it is overwritten manually from the updated copy. It is available for quick reload of the call/program stores if the system lacks sanity on the new data.

III. DIAGNOSIS

3.1 Overview

3.1.1 General description

The prime functions of the 1A Processor diagnostic programs are fault detection and generation of failure data used to locate faults. Diagnostics are developed using a high-level macro language and are table driven to facilitate multiple applications. The diagnostics are resident in the auxiliary unit (AU) community on disk and are paged into main memory when executed. They are specifically designed to run in a relatively short elapsed time and to minimize the storage requirement. The table-driven/macro language design simplifies diagnostic design, test development, and debugging. It also simplifies modifications and fosters standardized documentation.

3.1.2 Diagnostic objectives

(*i*) Maximum fault detection—This objective concerns applying tests to a unit and having one or more of these tests fail if the unit is malfunctioning. However, economic constraints prevent detecting all faults.

(*ii*) Consistent test results—The diagnostics contain sufficient hardware initialization and test-output analysis to insure consistent test results for a given hard fault.

(*iii*) Protection of memory—The diagnostics are designed to minimize the possibility of destroying information stored in either main memory (call stores or program stores) or auxiliary memory (disk and tape).

(*iv*) System noninterference—The diagnostics are designed not to interfere with the normal operation of the system.

(*v*) Single replaceable module resolution—The objective is that test failures will allow resolution of a fault to one replaceable module (circuit pack). However, economic constraints prevent this resolution for all faults.

(*vi*) Program flexibility—The diagostics are designed so that various test options are available to maintenance personnel. The environment for testing a unit can be controlled by executing only part of the diagnostic, by removing or restoring other system resources, and by specifying other system units in the test configuration.

(*vii*) Efficient tests—Efforts were made to minimize the number of tests required to hold down the program size and execution time.

(*viii*) Program documentation—The diagnostics are designed with a high degree of standardized documentation to simplify program maintenance and to aid in the repair process.

3.2 Design approach

The table-driven/high-level-macro approach is used to design and develop the diagnostics. The diagnostic tests are a collection of macros that expand (when assembled) into a data table and drive (when interpreted) a control program that applies the tests to a particular unit. Section 3.3 explains the structure in more detail. The high-level-macro approach facilitates using the diagnostics as a common data base for several applications. By designing a macro-expansion package for a particular application, the data base is assembled to provide the driving

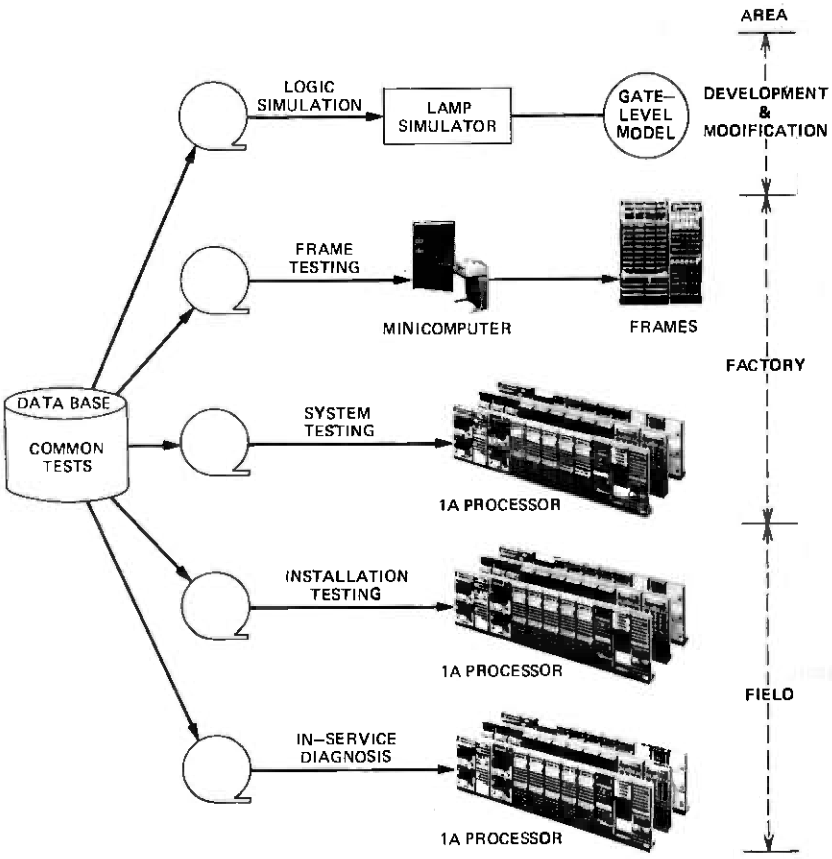


Fig. 2—Multiapplication of diagnostics.

inputs for the application. Figure 2 shows the current applications of the diagnostic programs. In the area of hardware and diagnostic development and modification, logic simulation plays an important role. The tests are assembled as inputs to a simulator called LAMP[7] and are applied by LAMP to a gate-level model of each unit. This application simulates the effects of the tests on the units and provides logic and diagnostic verification from the start of the design process to the completion of development. In the factory environment, the diagnostics are used for frame and factory system testing. For frame testing, the tests are assembled as inputs to a minicomputer which controls and drives a unit. This testing permits extensive circuit and diagnostic verification prior to interconnecting any of the units. For factory system testing, the processor units are interconnected and are driven by an installation test version of the diagnostics. In the field environment, the diagnostics are used for initial

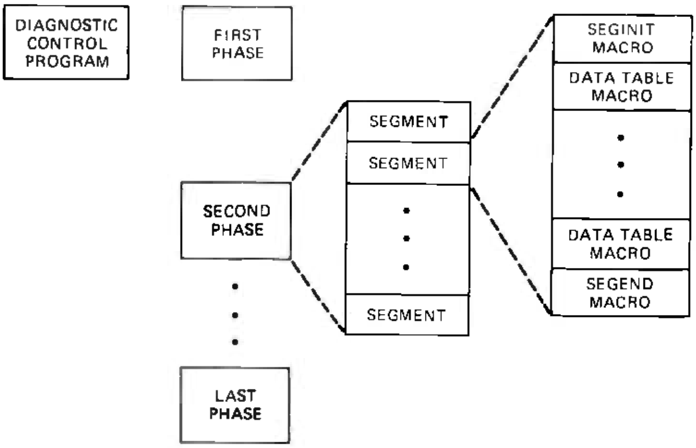


Fig. 3—General diagnostic structure.

installation testing and for the in-service diagnostic tool. The initial installation test consists of applying the same version of the diagnostics used for factory system testing to the completely interconnected processor in the field when it is first installed. The final application is the permanent in-service diagnostic, which is part of the generic maintenance software package.

Another important impact on the design process is that the diagnosticians and hardware designers work as a team, from the initial concept of the hardware design through the completed system. The hardware and diagnostic designs proceed in parallel, and each is used to verify the other. Diagnostic personnel have to agree to hardware changes made to a unit to insure the high level of fault detection required for overall system reliability.

3.3 Organization

3.3.1 General structure

The 1A Processor diagnostic is made up of individual unit diagnostics. Each unit diagnostic is a collection of many diagnostic phases and a control program. A diagnostic phase is a paged program module that is brought into main memory from auxiliary-unit memory when required for execution. Each phase is a collection of diagnostic segments and each of these segments is made up of data-table macros. Figure 3 illustrates this structure. The data-table macros expand when assembled into the data table that drives the diagnostic. These macros appear as a high-level language to initialize and test a portion of the unit being diagnosed. Each diagnostic segment begins with a SEGINIT macro, which performs certain initializations required for the particular unit, and ends with a SEGEND macro, which performs a clean-up function and takes a real-time break. Each segment is designed to run in less than 2.5 ms when failing tests do not occur (less than 3.5 ms if all tests fail). In some special cases, additional real-time breaks must be taken inside a diagnostic segment to meet this design requirement.

3.3.2 Data structure

The macro language for the 1A Processor diagnostics is called DL/1 (Diagnostic Language/1). Diagnosticians specify sets of these DL/1 macros that perform (when implemented) basic read, write and associated test functions for the various units. A typical example of two DL/1 macros is:

CCWRITE WORD (address), DATA (data)
CCREAD WORD (address), EXPECT (data).

These two macros perform a simple test of the standby central control (CC) by writing a data pattern into an internal location and then reading the internal location and comparing the results of the read with the expected results.

Each DL/1 macro expands when assembled into an INDEX word and perhaps additional DATA words. The INDEX word contains the index field, which is a unique value associated with the particular macro type; the remainder of the word is used for data. A typical expansion for the two CC macros is:

Write address	CCWRITE index
Data to be written	
Read address	CCREAD index
Expected data	

INDEX WORDS

3.3.3 Control structure

Each unit diagnostic has a control program that is comprised of a small task dispenser and a set of task routines. The control program is table driven and uses the data structure described in the previous section. An interpreter for the data table is required to pass control to ESS assembly language routines that perform the required work. The interpreter is a program unit called a TASK DISPENSER. It has a pointer to the next INDEX word in the phase being executed. It fetches this word and transfers control to the TASK ROUTINE associated with the value of the index field in this word. The TASK ROUTINE is the ESS language

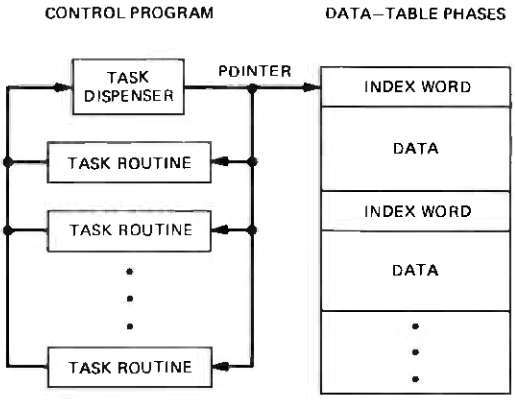


Fig. 4—Diagnostic table-driven structure.

routine that fetches any additional data words associated with the macro, performs the appropriate work, updates the pointer so it points to the next INDEX word, and returns control to the TASK DISPENSER. Figure 4 represents the entire table-driven diagnostic structure.

3.4 Test generation and execution

3.4.1 Test design

Test design is an iterative process guided by logic-simulation results and by experience gained from the various applications of the diagnostics, as described in Section 3.2. An objective of the diagnostic tests is to achieve complete coverage in the sense that the tests result in all logic nodes being exercised. However, a constraint is placed on this objective by economics.

Unit diagnostics are designed by diagnosticians who are familiar with the available DL/1 macros and with the hardware to be tested. Most tests are derived to exercise a part of the unit functionally. The remainder of the tests are designed using either automatic test-generation techniques or manually designed tests based on a gate-level logic diagram of the circuit. Great care is taken to test the input and output nodes of a unit before using them to test into the interior logic. Such a test-design sequence is feasible because of the maintenance involvement early in the hardware design, which is directed towards obtaining a sufficient degree of maintenance access for each of the units.

3.4.2 Test execution

Diagnostics are designed for straight-line execution. This design philosophy may be interpreted as running all tests and using a post-processing scheme to determine the problem. To put it another way, no attempt is made to evaluate test results during diagnostic execution making branching decisions based on these results. However, the ability to perform simple forward branching is provided using data-table jump macros. The jump macro provides the capability to skip over tests that require other units that are not in service at the time they are needed or are not equipped in the system. Another reason for the jump macro is that a particular unit-type can exist in different systems at various hardware design levels. A diagnostic can handle a few of these design levels by running or skipping sets of tests sensitive to particular hardware design states. The final use of the jump macro is to terminate the diagnostic early when system integrity could be destroyed or when additional useful information cannot be gained by further execution. Care is taken when using the early-terminate function so that enough test data have been generated to solve the problem before terminating.

3.5 Interfaces

3.5.1 Diagnostic triggers

There are three ways to trigger a diagnostic. The first is automatic fault-recognition and recovery programs. During normal system execution, when a problem is detected, the fault-recognition and recovery programs request a diagnostic to be run on a unit to begin the repair process.

A second method is automatic routine exercise. Periodically, each unit is removed from service and diagnosed to detect latent faults.

Manual initiation is the third method. Manual diagnostics are initiated by either frame-control action or by teletypewriter (TTY) requests. Each unit has a frame-control switch. By changing the state of this switch, craft personnel can remove the unit from service, diagnose the unit, and restore the unit to service. This is especially convenient when repairing the unit. The TTY requests can perform the same functions. A unit is diagnosed when an input message to diagnose or restore a unit to service is received. An example is:

```
RMV:CC O! #remove CC O.
DGN:CC O! #diagnose CC O.
RST:CC O! #restore CC O.
```

If the diagnostic triggered by the RST input message executes without failing a test, the unit will be restored to service. When using the DGN input message, a special parameter, TLP (trouble location procedure) can be employed to trigger a post-processing system that evaluates the failing result of the diagnostic and generates an ordered list of suspect replaceable modules to be changed one at a time. A detailed description of this capability can be found in Section IV.

3.5.2 Backup techniques

The first-line maintenance-repair procedure consists of executing the entire diagnostic for a unit and employing TLP to evaluate the resulting failure data and resolve the problem. However, if the fault in a unit is either not resolved or even not detected using this procedure, various backups can be employed. When a unit diagnostic is triggered by either of the first two methods discussed in Section 3.5.1, the automatic phases for the diagnostic are executed. Some units also have a set of phases called demand phases. These phases can only be executed by specifically requesting them via a manual TTY request. These tests are usually long-running exercise-type tests that are particularly helpful in locating intermittent faults and resolving access or memory faults in the primary or secondary storage devices.

When using a manual DGN TTY request, any subset of the automatic and demand phases can be selected to test the unit. Individual phases, groups of phases, or entire diagnostics can be run repetitively to establish consistency. The detailed failing results of these tests, referred to as raw diagnostic data, can be used to aid in the repair of the unit when the first-line maintenance-repair procedure fails to resolve the problem.

For even greater flexibility, an additional diagnostic tool called the exercise mode (EX) is available. EX is an input message verb like DGN, RMV, and RST that allows full and partial diagnostic runs along with repetitive execution of a phase or phases, and also permits probing inside a particular phase. Using this tool, one can step through a phase executing one or more segments at a time, advance to the end of a segment and stop, or can loop over one or more segments a specified number of times or indefinitely until manually terminated. When looping indefinitely over a set of tests, a SYNC pulse can be generated at a specified place allowing the circuit to be analyzed with an oscilloscope.

The DL/1 macros used have various self-documenting capabilities, such as producing a test number and specifying the address, data, and expected data. This effectively yields an in-line documentation that is valuable for resolving faults that elude TLP. Various documents are also available to assist maintenance personnel when evaluating raw diagnostic data.

IV. TROUBLE LOCATION

A standard trouble-location procedure (TLP) has been developed for the 1A Processor that encompasses an on-line TLP program and office-resident data bases. Programs were also developed to produce diagnostic

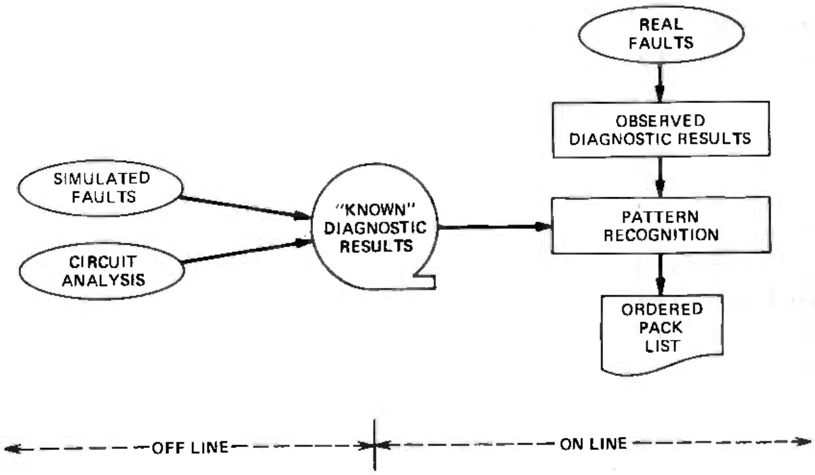


Fig. 5—Generation of pack list for trouble location.

failure results used by off-line programs to generate the resident TLP data bases. The primary output of the TLP program is an ordered list of suspected faulty equipment modules, which is referred to as the TLP "pack list."

When a fault is detected by the diagnostic program, the diagnostic results are passed to the on-line TLP program. The diagnostic results contain one data entry for each failing diagnostic test. Each entry contains the unique test number assigned to the diagnostic test and a 24-bit error word computed by exclusive ORing the expected test result with the observed test result.

Considerable processor resources would be needed to use the entire set of diagnostic results in an unprocessed form in subsequent TLP operations. Instead, each diagnostic result is summarized by extracting distinctive pattern features in the form of numerical quantities assembled into a signature. This signature is used to classify patterns obtained both off-line and on-line and to associate with each signature an intermediate circuit pack list. As shown in Fig. 5, off-line data is generated from circuit analysis or simulation of "classical" faults (i.e., idealized fault conditions such as logic nodes stuck at "0" or stuck at "1," or gate terminals opened or shorted to ground) using a physical or computer model.

The result observed on-line after diagnosing a real fault may not resemble simulated fault behavior; in fact, it may reflect the presence of an arbitrary failure condition including marginal circuit performance and multiple malfunctions. Primarily for this reason, a pattern recognition process is used to compare the signature of the observed result with an office-resident data base of known signatures to obtain the closest matching patterns. The associated intermediate pack lists are then merged into one resulting list, which is ordered by the probability of implicating the fault. The remainder of this section will describe the three basic TLP components: feature extraction, on-line trouble location, and TLP data-base generation.

4.1 Feature extraction

Diagnostic failure patterns are summarized by extracting their significant features. Three methods of feature extraction are used depending upon the type of diagnostic analysis employed. These methods are: connectivity, first-test-failure (FTF), and pattern analysis (PA). With the connectivity approach, diagnostic failure patterns are analyzed by relating individual test failure results to an associated addressable point inside the circuit being tested. An intermediate pack list is derived by processing design file information and tracing all electrical connections emanating from these addressable points. Note that these intermediate pack lists are derived from the connectivity of the circuit; the behavior of the circuit is not used (and, hence, simulation is not required). Since the 1A Processor diagnostics use behavioral approaches (FTF and PA), the connectivity method will not be further discussed in this article. The connectivity approach is used extensively in the No. 4 ESS periphery and is described in detail in Ref. 5.

In the behavioral approaches, the selection of features is based to some extent on the way the tested circuitry is interconnected, but is principally based on the way in which the circuit behaves in the presence of faults. Choice of the behavioral approach is based on both the type of circuit to be tested and the ability to acquire the necessary data for pattern recognition. If it is feasible to simulate the behavior of a circuit under diagnosis in the presence of all classical faults, the FTF approach is used. If it is feasible to manually analyze entire failure patterns, the PA approach can be chosen. When neither behavioral approach is feasible, the connectivity approach can be employed.

4.1.1 First-test-failure approach

Most of the 1A Processor circuitry is embodied in sequential logic units. The fault behavior of these units can be analyzed using "complementary simulation" (Fig. 6). With this technique, faults can be simulated physically (in the system laboratory) and logically (on a general-purpose computer), and the results from both methods can be compared for reasonableness and accuracy. This dual method provides both a convenient method for validating the results and more extensive fault-simulation data than would normally be available if either process

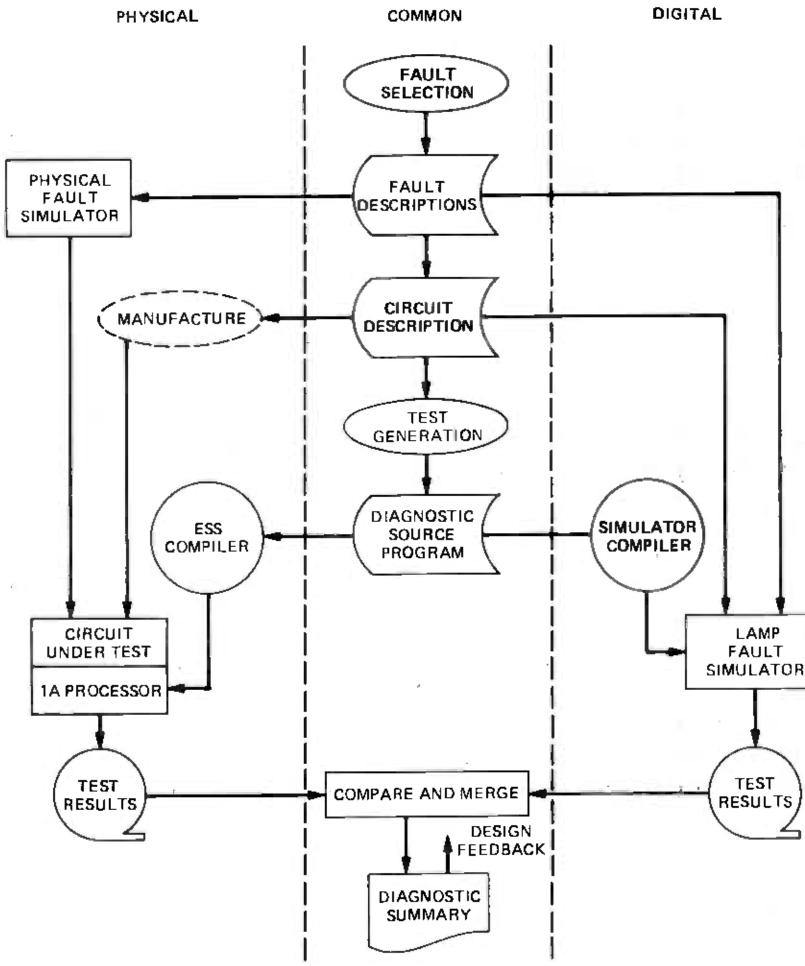


Fig. 6—Complementary fault-simulation system.

were used individually. Complementary fault simulation is discussed in Ref. 6.

ESS diagnostics are characteristically designed so that with each succeeding test there is a minimal dependence on circuitry not yet tested within the diagnostic. To capitalize on this characteristic, features extracted from the corresponding simulation results are heavily weighted by the earliest test failure results. The most significant feature is the first test failure; hence, the set of features derived from sequential logic simulation has been called the FTF signature. Additional features extracted from diagnostic failure patterns are: the first error word, the number of test failures in the first failing phase, the number of failing phases, the total number of failing tests, and a scrambled number computed over the entire pattern.

Associated with each signature is an intermediate pack list, ordered according to the number of simulated faults on the pack that result in the associated signature. The bulk of the behavioral data base is composed of FTF signatures and associated pack lists.

4.1.2 Pattern analysis (PA) approach

Memory media and large regular circuit arrays require numerous diagnostic tests and, hence, simulation of all classical failure modes is frequently impractical. Fortunately, the location of a fault in such circuitry can be deduced from analysis of the overall pattern of failures rather than from early failures. This technique is termed pattern analysis (PA). To make the best use of the PA technique for call store, program store, and file store, the associated diagnostics employ exhaustive location-by-location test sequences at various readout thresholds and "worst-case" test patterns.

Frequently, memory testing will not produce exactly the same failure results from one run to the next for the same fault. However, there are invariant pattern features, and these are extracted for PA signatures. Some examples are: an unusually high error count for certain address ranges, an unusually high error count for specific portions of the data word, the confinement of all errors to a specific memory module or sector, or an unusually high error count associated with a particular memory cell transition state. It has been found that approximately twenty to thirty separate features, depending on the type of memory, can be used to classify such failure patterns using a PA signature. Since the required PA data base is relatively small (several hundred signatures) and the circuitry is quite regular, the intermediate pack list associated with each signature in this data base can be manually derived by the diagnostician. The PA data base is verified and augmented by sample fault simulation.

4.2 On-line trouble location

The on-line TLP program is a five-step process which (*i*) collects the diagnostic results, (*ii*) summarizes the diagnostic results into one or more signatures and places them on a disk holding queue called the TLPQUEUE, (*iii*) locates the desired data base on the TLP tape, (*iv*) determines the sequence of closest signatures (according to "weighted distance") and merges the associated intermediate pack lists, and (*v*) prints the final pack list. A weighted distance measure is used to take into account the relative significance (or weight) of each signature parameter. The TLP program requires a significant amount of processing time and storage, but due to the inherent reliability of the 1A Processor, the TLP program is infrequently executed. These considerations dictate that the TLP program be a disk-resident (paged) program that executes as a segmented maintenance client. To efficiently utilize the maintenance resources of the 1A Processor, the TLP program stores a summary of the diagnostic results in the TLPQUEUE and releases these resources while the TLP tape is being positioned to the desired section of the data base. Once the positioning is complete, the maintenance resources are again used by the TLP program to complete the task of producing the pack list.

The process of matching behavioral signatures with like entries in the FTF or PA data base is one of pattern recognition. The process finds an optimum match between the signature of the observed result and one or more entries in the appropriate data base by computing weighted distance functions. The weights account for the relative significance of each pattern feature quantized in the FTF or PA signature. In nearly all cases, several intermediate pack lists are generated even if there is an exact match. Near matches are considered if they are within a predetermined distance. A system of weighting functions is applied to each of the intermediate pack lists and a composite list is produced based on these weights. Weights are applied so that lists referencing the best matching signature will move to the top. Confidence factors (numbered 0 through 10) are computed to indicate to maintenance personnel the degree of match (10 represents an exact match).

Several 1A Processor diagnostics (e.g., call store, program store, and file store) occasionally require the simultaneous use of both feature extraction processes, FTF and PA. For these cases, the final pack list is produced by combining component pack lists, which are produced by the two signature types according to a merging algorithm.

Magnetic tape was chosen as the storage medium for the office resident data bases instead of the other available memory systems because of the large size of the data bases and the low frequency of access. The 1A Processor data bases alone contain over 2.5 million bytes. Data bases of comparable magnitude are also required for application maintenance. The frequency of access is expected to be less than three times a day in a stable office for both processor and application maintenance.

In addition to low cost, magnetic tape has other advantages: costly printed trouble-location manuals are eliminated, the process of updating data bases in the field can be controlled and is accurate, and updating does not have to be linked to reissue of the generic program.

However, the use of tape does exact a price. Increased access time is required to locate the desired data base on the tape. Additional TLP program complications occur in positioning the tape, and administration of the TLPQUEUE is needed to hold the summary during repositioning.

Because the 1A Processor tape units move relatively slowly (800 bits/inch at 25 inches/second), several minutes may be needed to position the data-base tape; therefore, several entries may exist in the TLPQUEUE at one time. I/O messages are provided to allow maintenance personnel to examine and modify the contents of the TLPQUEUE and/or the activity of the TLP program. When the TLPQUEUE is full, diagnostic results from subsequent diagnostics cannot be immediately used by the TLP program. However, the diagnostic results are processed to the extent that a summary of the TLP data is printed for later use. Data are automatically removed from the TLPQUEUE when the associated pack list is printed.

When diagnostic results are not entered in the TLPQUEUE, maintenance personnel can simply rerun the diagnostic when there is space in the TLPQUEUE. However, if the fault is marginally detected (i.e., the diagnostic may not always fail the second or third time), special input messages are provided so that maintenance personnel can manually reconstruct the TLPQUEUE entry from the earlier terminal printout of the TLP summary data. This feature can also be used to remotely generate pack lists for other offices in emergency situations.

The on-line nature of the TLP program also has several advantages: it supplies a common interface for each TLP approach; it allows for more complex approaches, which result in better pack lists and therefore faster frame repairs; and it allows for future modification of and addition to existing approaches without the necessity of retraining maintenance personnel.

The TLP program structure permits additional trouble-location approaches to be easily added. Frame-dependent interface programs control the overall TLP process by using subroutines supplied by the TLP program. This flexible control is used to dynamically select the TLP approach to be applied based on individual frame requirements.

4.3 TLP data base generation

Figure 7 shows the major data flows required to generate the TLP data base used by the 1A Processor. There are three data base components: FTF, PA, and connectivity. Each component consists of a data base composed of signatures and associated intermediate pack lists. The connectivity pack lists are generated by automatically tracing the interconnections among circuit components by processing design-file information.[5]

Pattern analysis (PA) pack lists are usually derived by manual analysis of the behavior of the unit; however, sample fault-simulation results are used to verify and augment the PA data base.

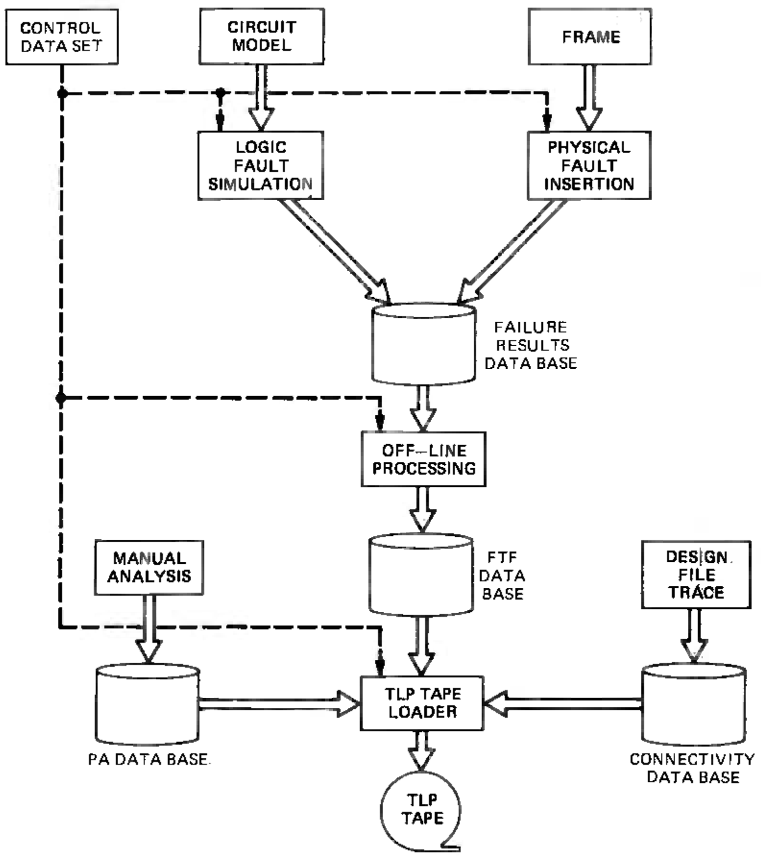


Fig. 7—TLP data-base generation.

As stated earlier, most of the TLP data base for 1A Processor units consists of FTF signatures and pack lists derived from complementary fault simulation. The physical simulation results are primarily obtained by inserting classical faults at circuit pack terminal-pin connections. Classical faults interior to the circuit pack and faults analogous to the above physical terminal-pin faults are simulated using LAMP.[7] In contrast to the universe of "real" faults, the simulated fault set does not include open power buses, open ground buses, marginal logic levels, marginal gate delays, crosses between signal leads, etc.

A basic assumption in the application of the FTF behavioral trouble-location technique is that the universe of all physically possible FTF signatures will be sufficiently covered by dealing only with classical faults. This assumption has been born out by the success, to date, of TLP performance (see Section VI).

Faults for both physical and LAMP fault simulation were directly derived from certain equipment files in the data management system (DMS). These files describe the interconnections within circuit packs and circuit-pack interconnections within an ESS unit. Fault collapsing (choosing one fault to represent a group of logically equivalent faults) is used extensively to minimize the number of faults handled.

The initial diagnostic results produced from physical and LAMP fault simulation reflected an ideal diagnostic environment in the ESS system. Quite often, however, diagnostics are run under less than ideal conditions; e.g., bus access to the unit under test might be limited, or other units associated with the unit under test might not be available to participate in the diagnosis. Under these conditions, diagnostic tests are sometimes skipped and the FTF behavioral trouble-location process suffers. Compensation for these effects is usually achieved by computing a "family" of signatures for each fault. One signature represents the ideal situation, and the others represent various off-normal equipment-availability conditions.

All stages of TLP off-line processing (Fig. 6) are controlled by software data sets, which serve not only to control the processing but also to document how the data bases are generated. The majority of the intermediate data bases are automatically generated from circuit-description files. This automation, coupled with other administrative facilities, permits the orderly updating of the TLP data bases and the TLP tape.

V. ERROR ANALYSIS

5.1 Objectives and uses

The fault-recognition, diagnosis, and trouble-location programs are the primary components of the maintenance software system, which is provided to the field to help isolate and identify faulty modules. However, even with high-quality maintenance-software design and thorough debugging, there is a need for backup procedures that can be applied to system troubles that elude the normal maintenance defenses. Trouble-isolation difficulties frequently occur as a result of transient, intermittent, or marginal malfunctions. These are usually classified as system "errors" rather than "faults." The resolution of these problems typically involves using past occurrences to extract patterns that may implicate a particular subsystem or component. Such analyses can be time consuming and often require special consultation with off-site maintenance experts.

The 1A Processor error analysis program was conceived as a backup problem-solving tool based on the use of a readily accessible system trouble history. This program, referred to as ERAP (for error analysis processor), has been implemented as an on-line storage and retrieval

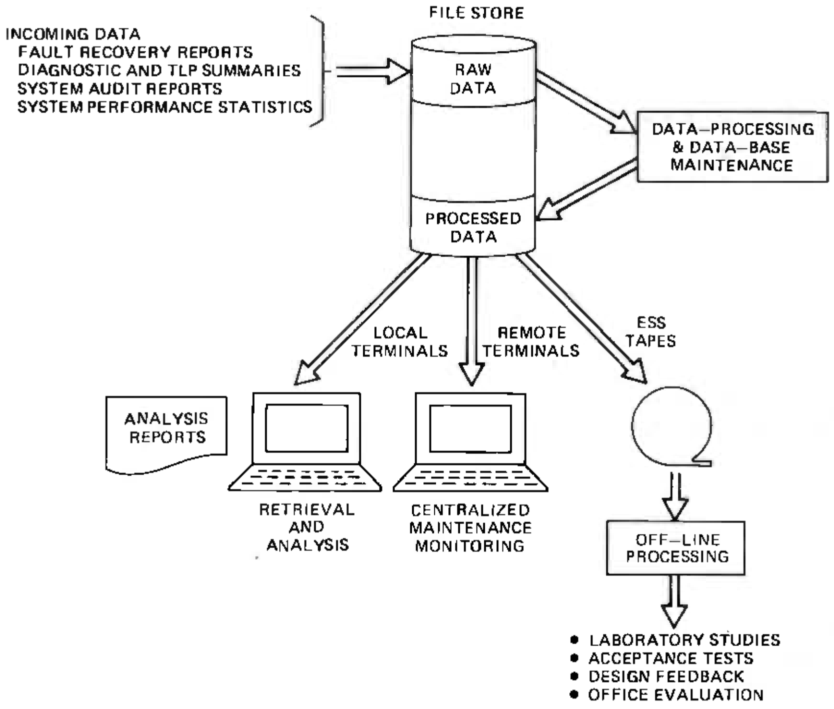


Fig. 8—Error-analysis system.

system. It automatically collects and formats appropriate trouble-history data, stores the data records in a dedicated area of file store, maintains the data base with its own audit program, and provides extensive retrieval capabilities via TTY input/output messages. In the design of the ERAP program, provisions were made to take advantage of the magnetic tape storage facilities of the 1A Processor. Thus, the entire ERAP data base may be transferred to tape at any time for subsequent off-line processing, and also data preserved on tape may be read back into file store and retrieved on-line.

An overview of what is effectively an error analysis system is depicted in Fig. 8. The incoming data accepted by the program consists of various types of records which fall into four main categories: fault recovery reports associated with system maintenance actions taken in response to detected malfunctions; diagnostic and TLP summaries; certain system audit reports; and overall system performance statistics. Incoming data records are initially buffered in main memory and then transferred to a file-store buffer area which holds the raw data until it can be processed. The data processing involves putting the records into standard formats suitable for subsequent retrieval, adding the formatted records to the permanent file-store data area, and updating directory tables and link lists. The data-processing routines are executed as clients of the maintenance-control program, MACP, and are paged in from file store when needed.

The data base maintenance and retrieval routines also operate in a paged program mode. Data retrieval is carried out in response to input TTY messages with displays on office-maintenance channels or on dial-up terminals located off-site in a centralized maintenance facility. The retrieval capabilities include pattern searching and data-scanning features that serve as manual analysis aids. Automated analysis procedures are not provided in the ERAP program. Instead, companion programs are available to process on-line-generated magnetic tapes on a commercial computer. In addition, advantage can be taken of the 1A Processor library-program feature to perform on-site reporting and data-analysis functions. The library program approach offers flexibility in introducing various analysis procedures, which may be run relatively infrequently, without requiring generic-program changes and also without occupying dedicated program space on file store.

The off-line processing capabilities considerably extend the scope of the original error-analysis design concept. The processing of error-analysis tapes is used in special laboratory studies, such as fault-insertion projects to evaluate recovery software, and in acceptance tests of new generics. Data sent back from the field are used to provide hardware and software design-feedback information to help evaluate office performance (particularly near the time of cutover) and to aid in the development of analysis procedures.

5.2 Data base formation and maintenance

The remainder of this section is devoted to a description of the on-line ERAP program, beginning with specifics on the nature of the file-store data base and the methods used to maintain that data base.

The fault-recovery records collected by the program consist of reports of each maintenance interrupt, maintenance interject, and base-level maintenance action taken by the system. These reports, which are also printed on an office TTY when the action occurs, provide dumps of the contents of selected internal unit registers and also contain data passed on by the fault-recovery programs. Included in the category of fault-recovery reports are records of each system phase of memory reinitialization, whether automatically or manually generated, and also failure reports resulting from deferred fault-recognition tests.

The next category of data collected (see Fig. 8) consists of summaries of each diagnostic carried out indicating, for example, the source of the diagnostic request and the main diagnostic results. TLP summary records provide further information on the diagnostic results, although such summary records are included in the data base principally for the purpose of off-line TLP evaluation studies. Diagnostic and TLP summary records, which arise as a result of fault-recovery actions, are associated with the recovery records by a link-listing scheme used to create error-analysis files. An error-analysis file consists of a collection of individual records that bear a common "maintenance file number." The error-analysis program maintains a file-number counter, which is passed on by a recovery program when it makes a diagnostic request and which is incremented each time a distinct recovery sequence is terminated. Records of manually-requested diagnostics and associated TLP summaries also result in error-analysis files that are separate from the recovery files.

Other types of data collected by the program, which are in the same category as TLP summary records, include optional detailed diagnostic results (raw diagnostic data), records of deletions from the TLP queue, and frame-repair records. A frame-repair record is manually input via the TTY to describe a TLP-based repair action involving the replacement of a particular pluggable module. These repair records are link-listed into the files containing the diagnostic and TLP summary records as well as any associated fault-recovery records.

The system-audit reports collected by ERAP summarize memory-mutilation errors found and corrective actions taken for all nontransient memory areas in main memory or file store. The resolution of memory-mutilation problems typically requires correlation of historical data and is a prime candidate for an error-analysis approach. The only other audit information collected by the program is that generated by ERAP itself when it corrects errors found in its data base.

The final category of data collected by ERAP is principally intended for off-line processing of error-analysis tapes for system-evaluation purposes. It includes overall performance statistics generated by traffic- and plant-measurement programs that are provided in the No. 1A and No. 4 ESS systems (see, as an example, Ref. 5). Also included are reports every half hour on units that are out of service for maintenance reasons.

The error-analysis program is designed to preserve its data base through severe system troubles, including phases of memory reinitialization and file-store memory mutilation. There is an extensive error-analysis audit program that is automatically requested after system phases and when ERAP program defensive-check failures are detected during normal execution. The audit program verifies the internal consistency of the reference tables (directories, file-number table, link lists, etc.) and also can compare the reference tables with the totality of records stored in the data base. Sufficient information is stored within each record to rebuild the reference tables if the consistency checks fail. Furthermore, each record has stored within it a check sum (formed when the record was originally constructed), which reliably indicates any unintentional alteration of the record. Mutilated records are automatically discarded from the data base, which is subsequently repacked. The audit program will use both file-store copies of its data base to find a correctable copy and will update the other copy automatically, as required. If both file-store copies are uncorrectable, the data base will be reinitialized. As mentioned earlier, all modifications in the data base by the audit program (including automatic and manual reinitializations) are themselves recorded in the data base for subsequent investigation.

Intentional deletion of records from the data base is accomplished manually by means of TTY input messages. (Automatic data-clearing operations are planned for future issues.) If the file-store data base overflows, data collection is halted. Warning messages are issued every half hour prior to overflow when the data base has less than ⅛ of its total assigned space available. Also, the current data-base occupancy may be determined at any time by means of error analysis TTY input messages.

5.3 Retrieval capabilities

Data output from the program is provided in response to error analysis TTY input messages that specify, via arguments of keywords, the type of data desired and the file numbers of the data of interest. In one retrieval input message any combination of subrecords, records, or entire files may be requested mnemonically by listing the appropriate arguments. Another form of retrieval takes place where specified files are scanned and particular words or items are extracted from named data blocks within the files. On output, the items are listed in matrix form with columns for the individual items and rows for the individual data blocks and files. This type of output is especially useful as an aid to manual recognition of data patterns involving sequences of interrupts or diagnostics. The program also has the capability to accept mnemonic specification of items to be retrieved so that specific layouts of data blocks need not be consulted.

Sample ERAP output of the form just described is depicted in Fig. 9. The input message shown in the figure requests that items named INS, SCA, SDA, SPA, CES, and SBYCES be extracted from all DLEV data blocks (associated with D-level interrupts) that may happen to be present in the file number range 1 through 77. The output message given in response to this request indicates that five files were found to contain DLEV-type data, and the desired items are displayed (in octal) along

```
INPUT MESSAGE REQUEST FOR DATA OUTPUT

    OP: ERAPDATA  DLEV: ITEM (INS, SCA, SDA, SPA, CES, SBYCES). MFNUM 1–77! PF

OUTPUT MESSAGE RESPONSE

  M 28  OP:ERAPDATA               COMPLETED
          NUM FILES = 5           MFNUM RANGE: 00000015 THROUGH 00000056
          DATA TYPES : DLEV
           MFNUM       INS        SCA       SDA       SPA       CES      SBYCES
          00000056  00002003  14652043  07753410  14652046  00504040  02001500
          00000027  00002100  14430015  00004000  14204050  01003100  02004040
          00000024  00002100  14430015  00004000  14204050  01003100  02004040
          00000022  00002100  14430015  00004000  14204050  01003100  02004040
          00000015  00002003  14654760  07752540  14654762  00500310  02004140
            04/15/76   10:28:17
```

Fig. 9—Sample ERAP output.

with the file numbers. The similarity of three of the D-levels is immediately apparent from this display.

One of the most powerful features of the error-analysis program is the capability of associative retrieval implemented by means of searches through the data base according to user-selected search conditions, which are input via TTY. A search condition imposes an arithmetic restriction on an individual item of a record or on the relationship between two different items from records within the same file. An example of the use of a search condition is the specification that a fault-recovery-requested diagnostic on a particular frame passed all tests, possibly indicating a transient error. Another example is that a memory-access failure occurred at a particular address of interest. When a search condition is entered via an input message, the translated condition is stored in the error-analysis data base where it remains for future reference until it is manually cleared by a special input message.

An actual search through the data base is requested by a TTY input message, which names the search conditions to be used. Two or more conditions appearing in the same input message will be logically ANDed. The result of a search is a memory block indicating the file numbers of those files which meet the input conditions. The search results are themselves stored in file store for subsequent reference by input messages used to output data or to perform additional searches. In this way, it is possible to output from files previously found in a search without having to enter the individual file numbers, and it is also possible to perform searches among existing search results with additional conditions imposed.

VI. EVALUATION

A definitive evaluation of 1A Processor maintenance objectives cannot be made until enough systems have accumulated sufficient operating experience in representative service environments, although several

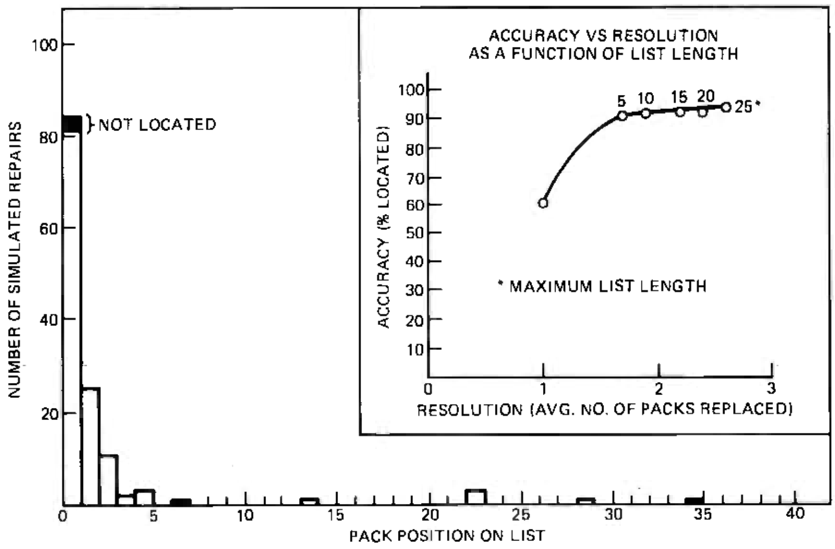


Fig. 10—TLP performance on 132 simulated repairs using field returns.

studies have been conducted to evaluate fault recognition and recovery, diagnostic detection, and trouble location to provide early estimates of 1A Processor dependability.

In one study, 2071 single faults were inserted by connecting randomly selected backplane points to ground using relays while the system laboratory 1A Processor was in a normal operating mode. Automatic system recovery occurred in 99.8 percent of the cases; manual assistance was required for only five of the faults. In another study using a random selection of 2400 classical faults, simulation results indicated that 95 percent of the faults were detectable by the diagnostic programs.

To evaluate the TLP process, circuit packs that had failed while in field or laboratory service or during installation were reinserted in one or more laboratory equipment locations. After each circuit pack insertion, the associated unit was diagnosed and the trouble-locating process initiated. In only five of the 132 simulated repair cases did the TLP fail to include the inserted pack in the resulting list. Figure 10 shows the distribution of simulated repairs as a function of pack position on the list; the maximum list length was used in the five cases where the inserted pack was not listed. Notice that only seven cases (5.3 percent) would involve more than five pack replacements to locate the failed pack or exhaust the list.

It is of interest to consider the effect of truncating the pack list at lengths of 5, 10, 15, 20, and 25 packs. Figure 10 depicts the resulting accuracy (percent of failures located) versus resolution (average number of pack replacements). In this experiment, a reduction of maximum list length from 25 to 5 would degrade TLP accuracy from 96.2 to 92.4 percent while improving resolution by one pack replacement per repair (2.7 to 1.7).

VII. SUMMARY

To achieve the rather stringent system-dependability objectives of the 1A Processor, four maintenance-software subsystems were designed: fault recognition and recovery, diagnostics, trouble location, and error analysis. Fault recognition has been enhanced by more extensive use of self-checking circuitry than that used in the earlier systems, such as No. 1 ESS. System recovery, while more complex because of the writable program store and dependance upon the rather complex file store, has been improved by increased use of short-term error analysis, additional bootstrap implementations, and the consolidation of control of deferred configuration requests. Extensive use of physical fault insertion in system laboratories has verified that, with very high probability, the 1A Processor can recover from faults without manual assistance.

Diagnostics have received considerable emphasis during maintenance-software development. Extensive physical and computer simulation, which began in the early stages of subsystem development and continued throughout the development, resulted in unit tests capable of detecting at least 95 percent of the classical faults. Using a macro-level test-specification language, a high degree of standardization was achieved. Essentially the same set of diagnostic tests is used to exercise a unit at the frame level, verify it at the factory, and diagnose it during normal 1A Processor operation.

Trouble location has been automated to the point where failure results from a unit diagnostic are translated to an ordered pack list, with the particular combination of distinct failure-analysis procedures used transparent to the maintenance personnel. Early data from in-service failure indicate that the objective of at least 90 percent of 1A Processor repairs requiring three pack replacements or less is being met.

Problems that elude diagnostic detection or the normal trouble-locating process usually must be resolved by analyzing error symptoms. Extensive data-collecting-and-processing software has been provided in the 1A Processor to assist such error analysis. Much of this error-analysis capability is used in the course of normal system maintenance; substantial amounts of the data are being saved for off-line performance and design evaluation.

VIII. ACKNOWLEDGMENTS

The work described in this article could not have been accomplished without the combined efforts of the numerous system designers who have been involved in maintenance software. The authors particularly wish to acknowledge the significant contribution of W. R. Hudgins and D. E. Lutz in the development of the TLP feature and in the preparation of this article.

REFERENCES

1. R. E. Staehler and R. J. Watters, "1A Processor—An Ultra-Dependable Common Control," International Switching Symposium, Institute of Electrical Communication Engineers of Japan, Kyota, Japan, 1976 Symposium Record, pp. 636–642.
2. G. F. Clement and R. D. Royer, "Recovery from Faults in the No. 1A Processor," Proceedings of the 4th Annual International Symposium on Fault Tolerant Computing, June 1972, pp. 5-2—5-7.
3. A. H. Budlong et al., "1A Processor: Control System," B.S.T.J., this issue, pp. 135–179.
4. C. F. Ault et al., "1A Processor: Memory Systems," B.S.T.J., this issue, pp. 181–205.
5. "No. 4 Electronic Switching System," B.S.T.J., (special issue), *56*, 1977.
6. F. M. Goetz "Complementary Fault Simulation," Proceedings of 3rd Annual Texas Conference on Computing Systems, University of Texas, Austin, IEEE Computer Society (November 1974), pp. 9.4.1–9.4.6.
7. "LAMP: Logic Analyzer for Maintenance Planning," B.S.T.J., *53*, No. 8 (October 1974), pp. 1431–1555.

Copyright © 1977 American Telephone and Telegraph Company
THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 56, No. 2, February 1977
Printed in U.S.A.

1A Processor:

Testing and Integration

By H. A. HILSINGER, K. D. MOZINGO, C. F. STARNES, G. A. VAN DINE

(Manuscript received July 16, 1976)

This article describes the general approach that was taken in testing and integrating of the 1A Processor. Included is a survey of the computer aids and test facilities that were used in performing the testing and integration tasks.

I. INTRODUCTION

Planning for system integration and testing started at the beginning of the 1A Processor[1] development. Much of this planning was based on experience gained in the development of No. 1 ESS[2] and other stored-program systems.[3,4] This experience indicated that the availability of high-quality tests, test facilities, and comprehensive test plans are critical to the orderly development of a high-quality system. In the development of the 1A Processor, tests and test facilities were designed concurrently with the design of the processor hardware and software. The objectives were timely integration of hardware and software components into a working system and thorough verification of the hardware and software designs.

The general approach to integration of the processor system was to design and test in parallel the various system components and to integrate incrementally these components into a working system. This paper describes the integration process in terms of three levels, namely:

(*i*) Circuit-pack verification and testing.
(*ii*) Frame verification and testing.
(*iii*) System integration and testing.

At each level of integration, computer aids played a major role in design verification before physical prototypes were built. Using computer aids, errors were uncovered in early stages of the design when corrections could be made with least impact on development resources and schedules. Substantial physical testing was done at each level of integration to verify proper implementation of the design not only in a normal environment but at various design limits. Special change procedures were developed for each level of integration to provide quick temporary fixes as well as permanent corrections for design errors.

Early in the project, it was recognized that many of the requirements for prototype testing at Bell Laboratories were very similar to requirements for new frame and installation testing to be done by Western Electric manufacturing and installation groups. Therefore, many of the tests and test facilities were developed jointly by Bell Laboratories and Western Electric engineers as common multiapplication designs. This approach was considerably different from that used to develop other systems, particularly with respect to processor diagnostic test design.

The 1A Processor diagnostic tests were designed jointly by Bell Laboratories and Western Electric as a common diagnostic data base for use in factory frame testing and installation (XRAY) testing, as well as for incorporation in the processor software for maintenance of an operating office. With this approach, a high-quality diagnostic data base was produced early in the development with much less manpower than would have been required to develop separate designs for the three applications.

This paper describes the Bell Laboratories application of the common tests and test facility designs to prototype testing and design verification. Not within the scope of this paper are the Western Electric applications, the numerous exploratory tests conducted prior to developing the 1A Processor, or the No. 4 ESS[5] and No. 1A ESS[6] system tests used to verify overall system functions in the various applications of the 1A Processor.

II. CIRCUIT-PACK VERIFICATION AND TESTING

2.1 Computer aids

2.1.1 Machine-aided design system

The circuit density and logic complexity achieved with 1A technology[7] has resulted in a need for computer aids at virtually every step of the design and manufacture. At the heart of the circuit-pack verification and testing process is a digital-circuit-pack machine-aided-design system (MADES). MADES generates design-verification information, documentation, manufacturing artmasters, and test data from a common data base built from the engineer's logic description of the circuit.

Figure 1 illustrates the organization and functions of MADES. The logic description of a circuit is input to the logic analysis program (LAMP)[8] and a simulation is performed. When the engineer is satisfied that the LAMP simulation has verified the design intent, an automatic placement program uses the LAMP logical description to:

(*i*) Partition gates to silicon-integrated-circuit (SIC) chips.
(*ii*) Place SIC chips on the ceramics.
(*iii*) Assign load resistors and other circuit components.
(*iv*) Place schematic symbols.
(*v*) Build the design data base.

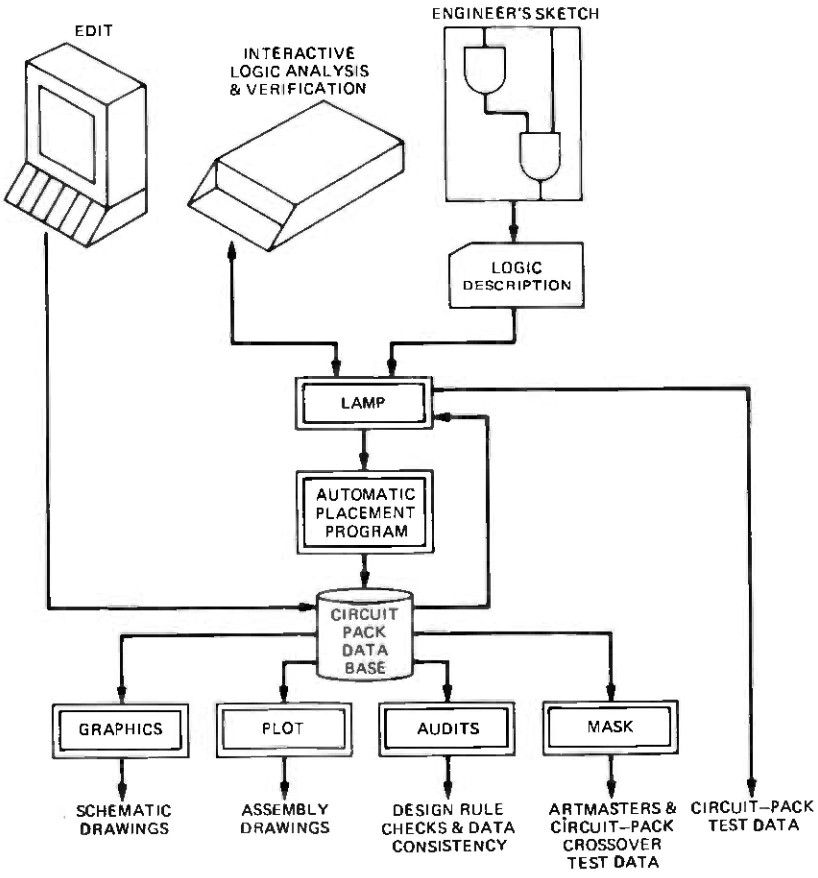

EDIT
INTERACTIVE LOGIC ANALYSIS & VERIFICATION
ENGINEER'S SKETCH
LOGIC DESCRIPTION
LAMP
AUTOMATIC PLACEMENT PROGRAM
CIRCUIT PACK DATA BASE
GRAPHICS
PLOT
AUDITS
MASK
SCHEMATIC DRAWINGS
ASSEMBLY DRAWINGS
DESIGN RULE CHECKS & DATA CONSISTENCY
ARTMASTERS & CIRCUIT-PACK CROSSOVER TEST DATA
CIRCUIT-PACK TEST DATA

Fig. 1—Digital-circuit-pack, machine-aided design system.

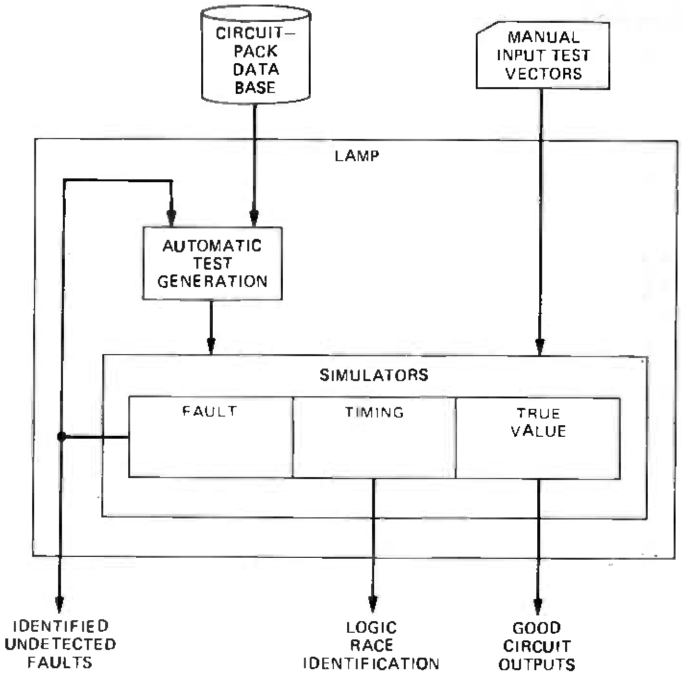

CIRCUIT–PACK DATA BASE
MANUAL INPUT TEST VECTORS
LAMP
AUTOMATIC TEST GENERATION
SIMULATORS
FAULT
TIMING
TRUE VALUE
IDENTIFIED UNDETECTED FAULTS
LOGIC RACE IDENTIFICATION
GOOD CIRCUIT OUTPUTS

Fig. 2—Circuit logic-analysis program.

From the design data base, a variety of computer programs can be executed to produce the final art work, documentation, and test data used in manufacturing. This highly automated process also provides for manual intervention for handling nonstandard cases. Audits are provided to assure that design rules are not inadvertently violated and that the data base can pass a sanity check for consistency.

2.1.2 Circuit-pack test generation

The LAMP program is used both as a general input interface to MADES and as a logic simulator used to verify design intent with the output results constituting circuit-pack test data. Figure 2 illustrates in greater detail the design verification capability of LAMP. The engineer first manually designs and simulates on the true value (no-fault) simulator a set of test vectors for logic-verification purposes. Any logic errors uncovered are corrected before proceeding further. Additional test vectors for fault detection are then generated either manually or by using the automatic test generation (ATG)[9] facility in LAMP. All test vectors are then applied to two fault simulators. The classical fault simulator systematically induces gate stuck-high and stuck-low faults. The shorted fault simulator induces adjacent-lead and crossover-short faults derived from circuit-pack artmaster information. If the test vectors are not adequate to detect all of the simulated faults, the undetected faults are identified and fed back to ATG or the engineer as appropriate. In some cases, the engineer incorporates logic changes to improve testability. This process is iterated until all or a high percentage of the faults are detected by the test vectors. The timing simulator is then used to detect any input race conditions, which are subsequently eliminated, and the true-value simulator is run to provide the "output" data for use on the circuit-pack test facility (described in 2.2.2 and 2.2.3).

This logic analysis program is capable of handling highly sequential as well as combinational logic and has been proven to be cost effective both in logic-design and test-design verification and for generating manufacturing data.

2.2 Circuit-pack testing

Circuit-pack test facilities were developed for use at Bell Laboratories as well as for use at the Western Electric factory. At Bell Laboratories these facilities were primarily used for testing and verification of "quick-fix" circuit-pack modifications to facilitate rapid turnaround of hardware design changes.

2.2.1 Crossover testing

The first level of pack testing is an electrical continuity check of the crossovers on the metallized ceramic. This is done before component bonding using a numeric controlled probe table driven from data generated by MADES.

2.2.2 Pass/fail circuit-pack test facilities

A circuit-pack test facility was developed for Bell Laboratories and Western Electric use in test and repair of logic packs. The basic test procedure is to apply the sequence of LAMP-generated tests to the circuit-pack terminals under computer control and compare the actual response with the expected good-pack response. The development model was also equipped with a high-performance reed matrix that allowed automatic connection to an auxiliary programmable instrument trailer for precision timing and parameter testing.

Figure 3 is an illustration of the physical prototype hardware used during the development phase of the 1A Processor. The fan-shape physical design of the high-performance reed matrix dramatically reduces the backplane wire lengths over conventional in-line pluggable connector arrangements. This was essential to control parasitic effects and get good impedance matching through the matrix. A programmable

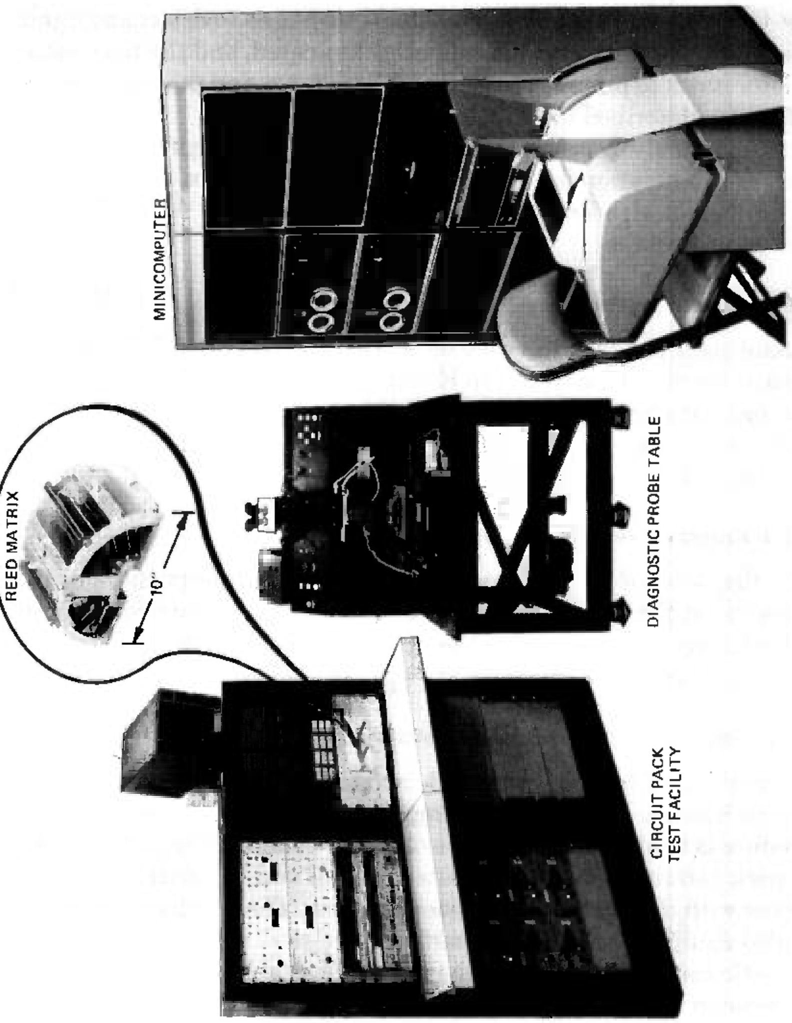


Fig. 3—Computer-controlled, circuit-pack test and diagnostic system.

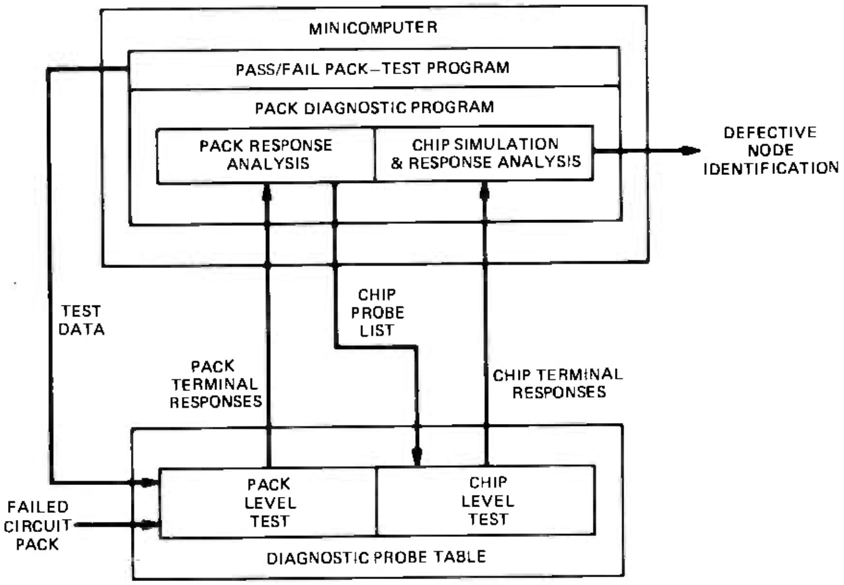


Fig. 4—Circuit-pack diagnostic system.

instrument trailer is equipped with a commercially available pulse generator, sampling oscilloscope, strobing voltmeter, automatic RLC bridge, frequency synthesizer, and programmable counter. The circuit-pack test facility provides the electronic interface for the logic tests and the ability to switch analog test equipment. The test system is capable of verifying the hardware design and separating good product from bad.

2.2.3 Fault diagnosis and isolation

To assure that this new technology could be manufactured economically, a means of effectively isolating and repairing faults was necessary. Conventional manual trouble shooting is difficult because of the complex and highly sequential nature of the logic and because of the microscopic dimensions of the physical components. To solve the physical probing problem, a computer-controlled probe table was developed capable of locating and contacting the beam-lead terminals on an individual silicon-integrated-circuit chip on the pack (Fig. 4). The defect isolation is accomplished in three progressive steps of testing starting at the pack level, progressing to the chip level, and concluding with some analog tests at the defective node.

The input to the diagnostic software system consists of basically a logic description, terminal and chip assignment information, and LAMP-generated circuit-pack tests. At the pack-test level, defect-analysis algorithms establish a list of potentially faulty chips based on failing responses at the pack terminals. This list of suspect chips, with a most-probable-faulty ordering, determines the probing sequence for chip-level testing.

Chip-level testing selects the next chip site to be probed, positions the probe (under program control), and monitors the chip terminals while the same (pack-level) test vectors are applied at the pack terminals. The chip inputs are measured and sent to a real-time logic simulator in the minicomputer, which then determines what the correct logic response should be. The result is then compared with the measured chip outputs. This procedure continues until there is a mismatch between a chip's response and the simulator. The bad node is then identified by chip location, beam-lead terminal number, failed-test number, and failed state (high or low). Note that the resolution is to the node only. A further analysis must be made to determine which chip or ceramic metallization detail tied to the node is at fault.

It should be emphasized that it is this second-level diagnosis (which isolates the bad node) that is by far the most important. The first level reduces the amount of probing but could be omitted. The second level eliminates hours of laborious tracing through a complex sequential circuit to find a defect that may not propagate a wrong response to pack terminals for over 100 tests.

One reason why this system is both unique and cost effective is that the simulation is done in real time on the minicomputer and so does not require stored data from previous exhaustive simulation at the pack level. This technique results in a tremendous reduction of the amount of data that must be maintained for each pack code. It is based on the philosophy that it is easier to generate, as needed, a small segment of simulation in real-time than to store and search results of an exhaustive simulation previously done on the entire circuit pack. This approach is also very effective in a multiple-defect environment. Because the chip simulator uses measured input values, it does not depend on correct or assumed input values to generate the expected true-value logic response. This allows testing to proceed beyond the first defect, which can result in locating additional faults deeper in a logic chain.

The third level of defect resolution is a series of analog tests designed to determine which device associated with a node is faulty. The technique used is to make sensitive voltage measurements through Kelvin probes to determine the current flow and device influence at a node. This level of testing could be automated, but since it is so easily accomplished by the test-facility operator, it is typically left as a manual operation.

III. FRAME VERIFICATION AND TESTING

The next level of assembly is a functional unit made up of a collection of circuit packs interconnected at the backplane and mounted on a structural frame. Some frames contain only one functional unit, whereas others contain more than one unit. The term "frame verification and testing" is used somewhat loosely in this paper, since "unit verification and testing" is implied for multiunit frames.

3.1 Computer aids for frame-design verification

3.1.1 Frame-design data base

Basic design information for each 1A Processor frame was assembled into a frame-design data base on the Bell Laboratories computer system. This data base then served as the primary source for automatic generation of data needed for design verification and manufacture.

Design verification prior to the assembly of prototype frames was accomplished through logic simulation (described in 3.1.2) and through the designer's inspection of automatically produced frame drawings. After automatic artmaster and wire layout generation, audit programs were run to obtain wire length and adjacent wire exposure measurements for evaluation of noise and crosstalk margins. A timing-analysis program was also used to evaluate worst-case timing margins on critical logic chains, taking into account such parameters as wire lengths, gate fan-out/loading, and worst-case gate delays.

3.1.2 Frame logic simulation

Logic simulation at the frame level played an important role in verification of initial logic and diagnostic test designs. Many circuit- and diagnostic-design errors were uncovered and corrected before the physical frame prototypes were constructed. Figure 5 shows the major components of frame-level simulation using LAMP on an IBM 360/370 TSS.

Primary inputs to LAMP were the logical model of the frame or unit to be simulated and the input vectors to be simulated. The logic model was assembled by the logic simulation language (LSL) compiler in LAMP, primarily from logic interconnection data extracted from the frame-design data base. Some errors in the data base were uncovered by consistency checks included in the LSL compiler. Input vectors were assembled for simulation from the macro language test statements in the diagnostic data base using the macro language facilities of the switching assembly program (SWAP).

Several runs were made on the LAMP simulation system. Initial runs used the true-value (no-fault) simulator and resulting output vectors

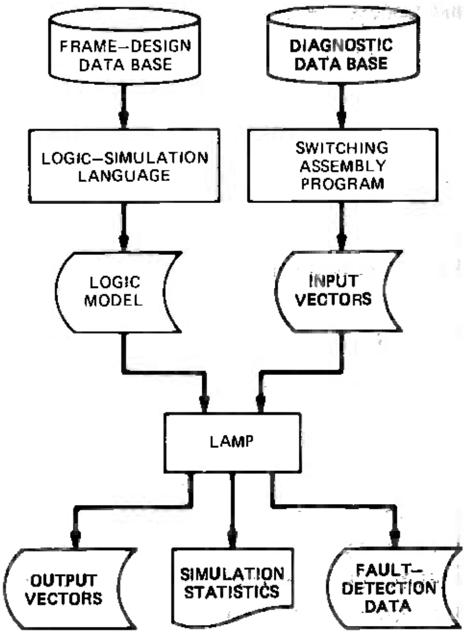


Fig. 5—Frame-logic simulation.

were compared with expected results (also in the diagnostic data base) for the diagnostic tests simulated. When discrepancies were found, the logic and diagnostic designers worked together to resolve them. Initially, only a few sample tests were simulated and many discrepancies found were due to weaknesses of the logic model. These required manual editing to correct. For example, digital models for certain analog circuits had to be added and "tuned" to properly simulate the analog circuits. After model weaknesses were corrected, more diagnostic tests were run. A number of discrepancies between simulation output vectors and expected results were traced to frame-design and diagnostic-design errors. Such errors were temporarily corrected by editing the logic model and input vectors in order to proceed with further simulation. Permanent corrections to the frame-design and diagnostic data bases could thus be made at a later date without delaying simulation.

After true-value verification of the logic model and the diagnostic test vectors, a second set of runs was made on the LAMP logic simulator to evaluate diagnostic fault-detection capabilities. As a result of these fault-simulation runs, additional tests were added to the diagnostic data base and some logic changes were applied early in the design to enhance fault detection.

3.2 Physical frame testing

As previously indicated, the processor diagnostic tests were designed from the outset to be general-purpose in application. The tests are comprised of sequences of input bit combinations (words) applied to a 1A Processor frame via its external send bus connectors. The test sequences are applied to the frame via these connectors by the central control (CC) in an operating system environment. In the frame-test environment the frame-test facility ties into these bus connectors and applies the same test sequences.

Associated with each test is an expected-result word which a fault-free frame should return on its reply bus. In the frame-test environment, the frame-test facility ties into the frame reply-bus connectors and checks test responses from the frame in the same manner that the CC does in an operating system.

In the case of the CC frame itself, additional access is available in the form of connectors which, in the office environment, connect to the other CC. This port provides the necessary communications by which the CCs compare results and test each other in case of disagreement.

As with the initial diagnostic tests, the frame-test facilities were designed by Bell Laboratories and Western Electric personnel working together. The resulting test set designs were applied both to early frame testing at Bell Laboratories and to frame testing at the Western Electric factory.

The frame-test facility consists of an interface unit and a minicomputer whose disk and core memories hold all of the test inputs and expected-results data. This combination is designed to emulate the operation of the active CC in applying a test and checking for the correct response with bus timing equivalent to that in an operating system. A printer associated with the minicomputer outputs the test results. The very large number of tests required to check out units as complex as 1A Processor frames virtually precludes a purely manual test procedure. Nevertheless, a manual control and a display panel are provided on the interface unit to allow manual intervention when necessary to isolate especially difficult problems.

In operation, diagnostic tests are run from the minicomputer and the output test results are printed. Any trouble area is quickly isolated because the test sequences are organized into test phases, each of which checks out a very specific area (or function) of frame logic. Once a repair is made to a failed area, the failing test phases are rerun. The procedure works well, even in the case of multiple faults, because the diagnostic test organization generally administers tests to an area of logic only through circuitry that has passed previous testing.

Figure 6 shows the arrangement of a typical frame-test facility.

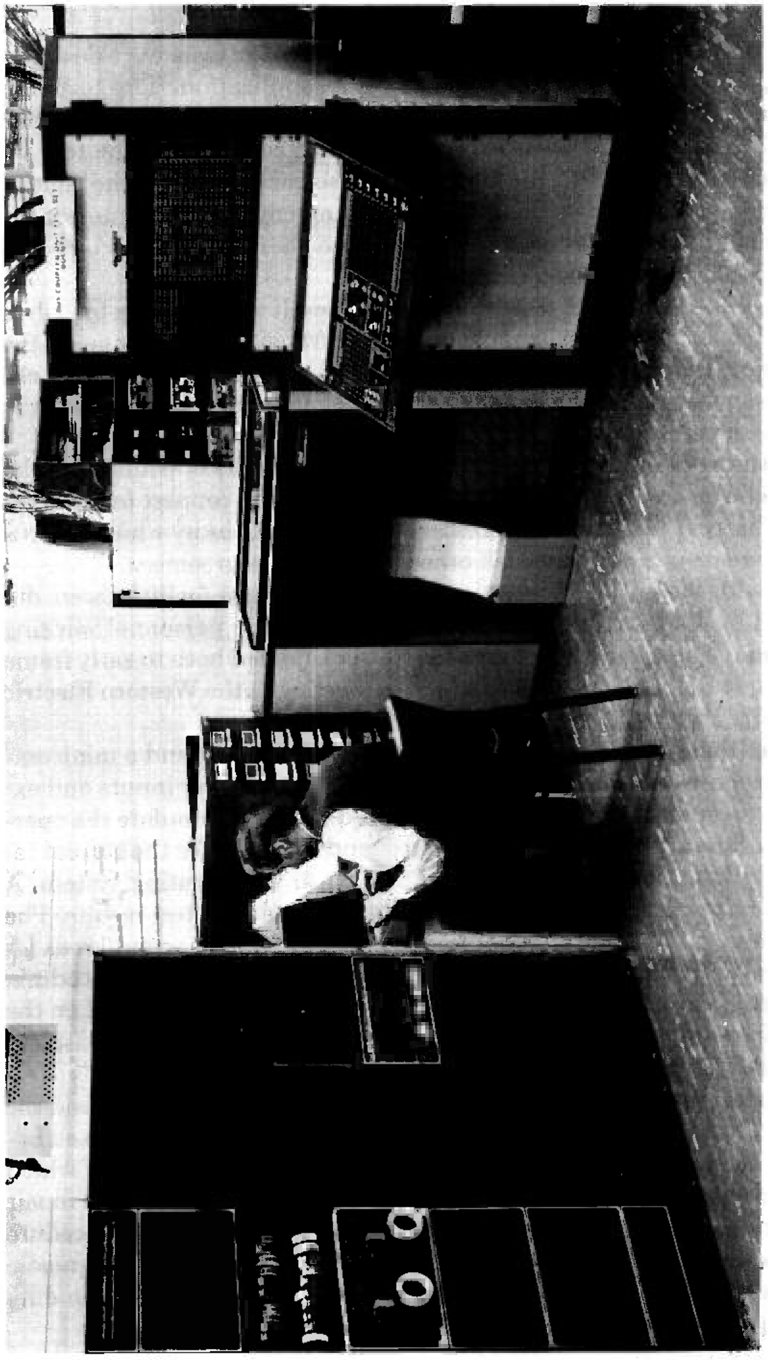


Fig. 6—Frame-test facility.

IV. SYSTEM INTEGRATION AND TESTING

Substantial testing/verification at the system level was not done until the first prototype processor was assembled, although some system-level functions were partially verified by simulation. For example, instruction fetch and execution were simulated on LAMP using the central control logic model connected to a functional model of a program store. Some special-purpose simulation programs were also written. For example, a FORTRAN program was written to simulate file store read/write transfers and direct memory-access interface with central control. This simulation was used in selecting an optimum algorithm for allocating call store and program store cycles to central control and file store.

Integration and testing of the 1A Processor hardware[10,11] and software[12,13] was done in three phases: initial hardware and diagnostic testing, software integration, and system test and evaluation.

4.1 Initial hardware and diagnostic integration

The objective of initial testing on the first processor assembled was to verify hardware interconnections, basic instruction execution, and diagnostic test capability. In planning for this initial integration phase, it was recognized that a special software "operating system" with simple input/output and diagnostic control capabilities was needed to bring up the initial processor. It was also recognized that essentially the same operating system was needed for Western Electric factory and installation testing. To meet these common needs, a software package called the installation test system (ITS) was developed by a team of Bell Laboratories and Western Electric engineers.

A primary objective of the ITS control software design was to provide tools for incremental buildup and testing from a minimum processor configuration to a full system configuration, including application peripheral units. The ITS package makes it possible to get the basic operating system cycling on a minimum configuration. This consists of one central control, one program store, one call store, and a special IO terminal interface board inserted in the central control. The terminal interface board provides a simple IO capability via direct access to the internal buffer bus of the central control. Special utilities are provided for use in pumping of program store from cassette tape or remote computer via the special terminal interface until the 1A Processor's system reinitialization (SR) function is operational. Other utilities are provided for bringing on-line additional program stores, call stores, and disks and pumping them from disk or tape.

As mentioned previously, the diagnostic test tables assembled from the multiapplication diagnostic data base provide the primary vehicle for factory system testing and installation testing. A number of options are provided by the ITS control software for interactive execution of the diagnostic tests. The diagnostic test tables can be paged from either disk, magnetic tape, or additional program store(s). Input commands are provided for halting diagnostic execution at a specified location, dumping internal registers or memory, and looping on a specified segment of tests with an oscilloscope synchronization pulse generated on a specified test.

ITS also provides for simultaneously running tests, including diagnostic looping, on several units in a time-shared multitask mode. With this multitask capability, a crew of testers can bring up several new frames in parallel once the minimum processor capability has been established.

Although the ITS control software was designed primarily for Western Electric factory and installation testing, it was an ideal operating system for debugging the first prototype processor and the diagnostic tests. The interactive multitask features were especially valuable for hands-on parallel debugging of subtle hardware or diagnostic problems that in general could not be resolved using a "batch run" mode of operation.

The initial hardware and diagnostic integration was completed and a tested processor with debugged diagnostic tests was delivered to the No. 4 ESS system-integration group within one year of receipt of the first processor frames from the factory.

4.2 Processor software integration

It is not within the scope of this paper to describe such computer aids as the editor, assembler/compiler, loader, etc. that were used in the design of the 1A Processor software (common to all applications). It is, however, appropriate to mention the use of the 1A Processor simulator.

A prerequisite to the introduction of program code in the system lab was that it first be unit tested on the 1A Program simulator. This simulator was implemented on an IBM 360/370 TSS and provided functional-level simulation of most 1A Processor instructions and associated register and memory operations. Using this simulator, much of the program unit testing overlapped initial hardware and diagnostic integration, and many errors were corrected before attempting to integrate the programs in the system labs.

The software integration plan for the system laboratory recognized that an incremental approach was needed to add and test program modules in an orderly manner, moving from the basic to the complex until all functions were operational.

The first objective in the integration plan was to integrate the 1A Processor utility system (described in 4.5) with the processor to provide for loading program stores (and disk backup). This was also necessary to obtain memory or register snapshots and program traces on various program or hardware match conditions, and to perform other utility functions.

The next objective was to verify or debug basic system configuration, initialization, and executive control cycling. Next, basic input-output message processing and IO terminal handling was verified. At this point the basic operating system was cycling.

Next, basic capabilities for fault recovery, subsystem configuration, disk administration, and diagnostic paging were verified, followed by verification of the operation of the basic diagnostic tests. With few exceptions, these diagnostic tests became operational very quickly because they had been previously debugged/exercised in ITS, frame testing, and logic simulation. At this point, a preliminary issue of the processor software was released to the No. 4 ESS and No. 1A ESS groups. This allowed each respective application to proceed with its system integration in parallel with the integration of remaining processor functions (such as memory audits, tape administration, error analysis, and field utilities).

4.3 System tests

Most of the processor software integration was done in a system laboratory containing only a processor and system lab utilities. To be sure that the processor would operate properly concurrently with other system functions, a set of processor tests was designed and executed in the No. 4 ESS and No. 1A ESS system labs. For these tests, the system lab was configured to simulate the environment of an operating in-service office. In this environment, the system lab utilities were turned off, all system units were configured on-line with no error detectors inhibited and a simulated call load was turned on.

An exhaustive system test would have exercised all possible system interactions and all possible function overlaps, but, of course, this was not practical. Instead, a set of tests were carefully designed to exercise (with a reasonable amount of system lab time) a wide range of interactions and overlaps. The tests were highly sequential, since early tests had to pass in order to enter later tests with the proper initial conditions. Each test required that a specific system stimulus or sequence of stimuli be applied and that proper system reactions be verified by the tester. Many tests required only input messages from a terminal as a stimulus. Other tests required frame power switch operation, a specific terminal on a frame to be grounded (to stimulate a stuck-at-0 fault), or other special action as a test stimulus.

4.4 Special evaluation and stress tests

In addition to the system tests that were aimed at design verification in a normal operating office environment, special evaluation and stress tests were designed and executed to test certain functions under abnormal or worst-case conditions.

To evaluate the overall fault-recovery and diagnostic program response to hardware faults, special fault-insertion experiments were performed. (These were performed in addition to the physical fault insertion for the trouble-location data-base generation described in Ref. 13). Before the first office cutover, over 2000 randomly selected faults were inserted one at a time in the processor and the resulting fault-recovery and diagnostic responses were evaluated. Many of the faults were reinserted under different initial conditions. For example, faults were inserted in a call store under the following five initial configuration conditions:

- (*i*) Active store faulted with a spare store in service and duplicating the faulted store.
- (*ii*) Spare call store faulted.
- (*iii*) Active call store faulted with a spare store in service but not duplicating the faulted store.
- (*iv*) Active call store faulted with all spare stores marked out-of-service.
- (*v*) Active call store faulted with a call store bus marked out-of-service.

As a result of these fault-insertion experiments, several problems were uncovered in the fault-recovery and diagnostic programs.

To evaluate the file-store system under worst-case load conditions, a special file-store exercise program was written to carry out experiments in the system lab. Using this program, file-store job requests of different profiles were submitted to the file-store administration program at a high rate over a considerable time interval, which thus simulated the high activity expected under peak call load conditions in No. 4 ESS and No. 1A ESS. These experiments were performed on different system configurations, including cases where a duplicate file-store controller was out of service and all jobs were required to be processed by the remaining controller. The file-store exercise experiments uncovered several subtle software and hardware design problems that were not apparent during integration and system tests because the problems tended to occur only under infrequent, high-activity-related conditions. A tape-exercise program was written to perform similar tests on the tape system and some experiments were conducted with the file-store and tape-exercise programs running simultaneously.

4.5 System laboratories

Two system laboratories were provided for the early stages of 1A Processor integration and testing. One lab was used for initial hardware and diagnostic software integration and, subsequently, as a test bed for verifying hardware change packages. The second lab was used for 1A utility system and 1A Processor software integration. Both labs contained a stand-alone processor (no network frames) with sufficient stores to hold the 1A Processor software.

Later in the project, most of the 1A Processor work was consolidated into a single lab, which is now used for field support, new feature development, and verification of design changes for cost reduction.

4.6 1A Processor utility system

A primary function of a system laboratory is to provide the facilities and the administrative environment in which system programs, written by a large number of programmers, can be tested, debugged, and integrated into a complete software system. The facilities must include a working processor, and a complement of peripheral switching equipment sufficient to allow the exercising of all program functions. In addition, special equipment must be provided to load or modify programs, establish a precise test condition, trigger execution of a program or function to be tested, accurately record resulting program actions, and provide printouts or displays to enable programmers to locate and resolve program bugs. This control and monitoring facility in a system laboratory is referred to as the utility system (which should not be confused with the field utility features provided in the 1A Processor software package).

The 1A Processor utility system (Fig. 7) is comprised of a utility computer (UC) and a utility test console (UTC). The utility system software resides primarily in the utility computer. This software is supplemented with a few special programs resident only in the system laboratory processor. All necessary communications with the 1A Processor can be handled through the UTC, which is a highly complex man-machine and computer-machine interface. Once a system lab becomes fully operational, of course, processor communications typical of a working office (TTY, tape units, master control center, etc.) are also available. The UTC contains data formatting and buffering circuitry through which card input, high-speed printer output and magnetic tape IO from the UC is interconnected with the processor. It also contains complex control and display panels through which either of the central controls (CCs) can be manually controlled and monitored. In addition, the UTC contains a high-speed semiconductor store and a wide variety

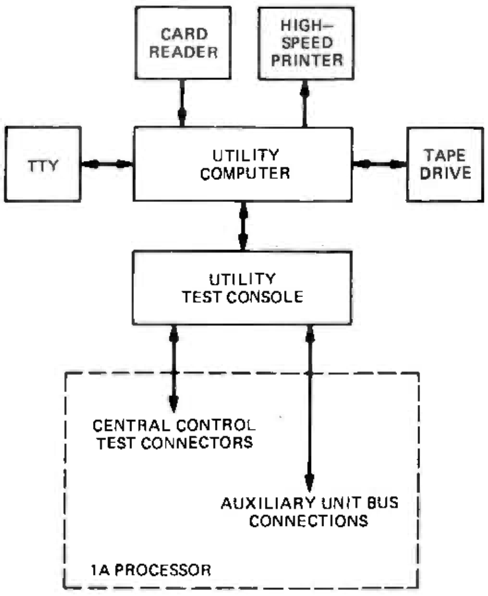


Fig. 7—Flow diagram of 1A Processor utility system.

of matcher circuits by which it autonomously monitors the operation of the processor as it executes a program.

Some discussion of UTC initial design considerations will be useful in explaining its operational features. In general, the reaction of the processor to program instructions may be determined by looking at the contents of the various CC registers after each program step is executed. This could be done by program interrupts that cause the reading of desired registers by a utility program, by slowing or stopping the CC clock for a hardware or a manual readout, or by designing the utility system hardware to be fast enough to do the readout in real time.

A lesson learned in previous ESS systems is that programs do not run the same when they are disturbed by program interrupts. Furthermore, the hardware may not behave the same at nonstandard CC clock rates as it does at standard because relative timing shifts may occur. The utility system had to be designed, then, to be able to monitor processor operations in real time at full system speed.

For a manually controlled operation, a "suspend CC" capability is provided whereby a signal from the UTC halts internal CC sequencers without altering register contents. The release of this signal, therefore, causes program execution to resume at normal speed from exactly the same machine state.

The use of integrated circuits in the 1A Processor has had far-reaching UTC design consequences. A large number of the internal CC registers, which must be monitored, appear only on buried circuit nodes on an integrated-circuit chip; the point does not appear on the backplane where a monitoring lead might be connected. This problem is handled by replication of all such CC registers within the UTC. The data and gating signals for these registers are accessible from the backplane, where they appear briefly on internal CC buses, as the registers are written or read. These and other backplane points are wired to test connectors for UTC access. Hence, once data is read into a CC register, its corresponding follower register in the UTC contains an exact copy of the data; this follower register is connected for continuous monitoring within the UTC.

Since the effect of specific program steps on the contents of specific CC registers must be known to determine the behavior of a program, the UTC is provided with a high-speed semiconductor store, the monitor store (MS), which saves this data. The MS holds 512 words; each word contains 520 bits, which is sufficient to hold the contents of approximately 21 CC registers. Data and addresses are "snapped" into the MS under control of a wide variety of matcher circuits. Masking capability is provided so that a matcher can look for single bits or bit groupings as well as complete words. Various single addresses, address ranges, clock times, or data words that might arise in program operation are prestored in these matcher circuits. When (or if) a bit combination (bit, bits, word, address, or address range) occurs, which a matcher is set up to detect, the matcher produces a trigger signal that causes a "snap" of CC register contents into the MS.

To monitor the central controls in real time, over 900 transmission-line connections tap into each of the CCs via the test connectors; over two gigabits of CC data flows into the UTC each second. The UTC dynamically selects, on a cycle-by-cycle basis, between the two CCs and in the choice of up to 21 of the CC registers. Both "next N" and "last N" program trace-modes are available. In the former case, the MS starts storing data upon receiving a trigger from a particular matcher; in the latter case, the MS continually stores data until it is stopped by a trigger. This allows the recovery of the recent history of program actions that led up to a particular event.

To access the system program-store and call-store memories, the UTC contains a memory-access unit and connections to the processor auxiliary unit (AU) bus. With this facility, the UTC accesses program store or call store using the direct memory access (DMA) circuit in the central control. This access, under control of the UC, is used for loading programs or overwrites in program store, for initializing call store, and for dumping program- or call-store memory as specified by the programmer.

In a typical batch operation, a programmer (or batch operator) reads

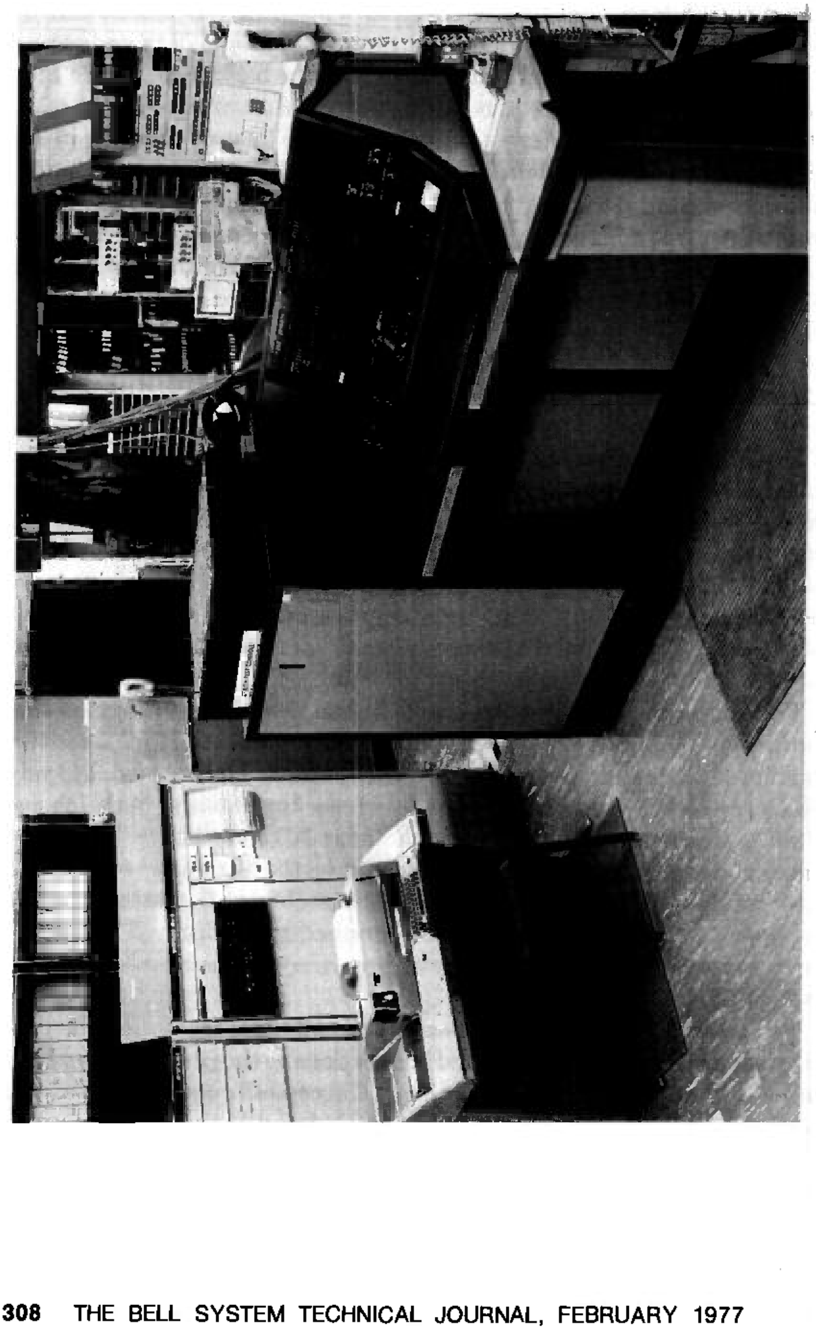


Fig. 8—1A Processor utility system.

into the UC a deck of cards that specifies the matcher setup and data-gathering parameters; small program changes (temporary overwrites) may also be entered on cards. New programs or large program changes are entered into the UC on a magnetic tape. The UC then sets up all matchers in the UTC, assigns MS locations to the CC registers to be monitored, causes the UTC to make any necessary changes in the processor program stores, and starts processor program execution. During the run, the UTC autonomously gathers the desired data into the MS. At the end of the run, the UC reads out the MS contents, and outputs this in prescribed formats on the high-speed printer for analysis by the programmer.

As mentioned previously, completely manual operation from the UTC may also be elected for certain difficult system problems for which such an operation is appropriate. Figure 8 shows a typical utility system arrangement in a system lab.

4.7 System change implementation

4.7.1 Trouble reporting

From the early stages of development, monthly status meetings were held to review progress on all aspects of 1A Processor development. Teams were defined for each subsystem with representatives from the circuit-design, diagnostic-design, physical-design, and data-base administration groups. One member, on a rotating basis, gave the monthly status report for his or her team. This team approach eliminated many interface problems, and the status reports tended to highlight serious problems not solved by the team and, thus, requiring the attention of management. When a serious problem arose, the proper experts were convened as a task force to solve the problem, and upper management monitored all organizations to assure coordination and cooperation.

Concurrent with the assembly of the first 1A Processor, a formal trouble-reporting system was implemented. In this system, all observed troubles including faulty circuit packs, wiring errors, hardware-design errors, software-design errors, and general system misbehaviors were documented and entered in the trouble-report data base. For each unresolved trouble, personnel were assigned responsibility for clearing the trouble. Subsequent progress on implementing a "quick fix," as well as a permanent fix, was entered in the data base. A report indicating the status of each trouble was distributed to management on a regular basis, which insured that troubles did not remain outstanding for an unreasonable period of time. Trouble reports on faulty circuit packs and other components were monitored closely for evidence of weakness in design, manufacturing, testing, or handling. Corrections were implemented where appropriate.

4.7.2 Hardware change administration

Early in the project it was recognized that frame or system debugging progress would be severely hampered if the only way to correct a design error was to produce new artmasters and fabricate new circuit packs or backplanes. Therefore, "quick fix" procedures were developed for manually adding or deleting connections on existing beam-leaded circuit-pack ceramics and on printed-wire and wire-wrapped backplanes. Procedures were also provided for adding and deleting chips and their connections on existing ceramics. These quick-fix procedures were also used to repair failed packs in situations where a new spare was not yet available. Using these procedures, modification or repair of a critical circuit pack or backplane could be implemented in a few hours, compared to the period of several weeks that might be required to produce a new artmaster and fabricate a new circuit pack or backplane.

Of course, the long-term permanent fix for errors uncovered in a frame design required complete design update including data base update, simulation, new artmaster generation, etc. Special software was implemented on the computer and interactive graphics terminals to assist in generating data base edits that were logically equivalent to quick-fix changes that already had been installed and verified in the frame-test facility or in a system laboratory.

Design-change activity was monitored closely to insure that critical problems were given priority with respect to allocations of resources and that design fixes were thoroughly verified before being propagated to system laboratories and field installations. Approximately two years before the first cutover of the processor, a hardware-change committee was formed composed of No. 4 ESS and No. 1A ESS system representatives, as well as representatives from the processor design groups. The function of this committee was to evaluate the priority and impact of all proposed hardware changes and schedule approved changes in early offices so that cutover schedules were not jeopardized.

4.7.3 Software change administration

A large software development requires special tools and administrative procedures for introducing changes in the software. The general objectives are to expedite corrections for serious bugs and to provide for smooth, minimum-impact introduction of all necessary changes, including new features or enhancements. It was very important that changes in the 1A Processor software be coordinated and introduced in such a way that progress was supported, not hindered, on the No. 4 ESS and No. 1A ESS application system developments.

An important administrative task was to evaluate on a regular basis all outstanding software problems (documented on trouble/failure reports described previously), to determine the seriousness of each problem and allocate resources, to monitor progress on fixes, and to schedule the necessary changes. Similarly, proposed changes for new features or enhancements were reviewed and approved changes were closely monitored and coordinated.

A change to fix a program bug was first tested and then implemented using the program overwrite facility of the 1A utility system. At a later time, the program source was edited and the program was reassembled for final implementation of the change. Program overwrites were installed on a daily basis and program reassemblies and reloads were done on intervals of approximately two months.

Program overwrites were prepared using the new "source overwrite" technique developed for 1A Processor and No. 4 ESS software. With this technique, the programmer prepared new high-level-language program statements to be inserted in the program or to replace old incorrect statements. These statements, along with appropriate insert, delete, and replace commands, were operated on by the source overwrite assembler. The assembler performed a partial program assembly (for the statements being changed), and produced both a program-overwrite deck in 1A machine language as well as a corresponding source-edit deck. The program source data set was not changed until the source-edit deck was applied, and this was not done until the program-overwrite deck was installed and verified in the system lab. The source-overwrite technique was a substantial improvement over previous program-overwrite techniques, since a programmer only had to fix the bug once using the high-level language. Also, new program loads were brought up much more quickly because new assemblies more closely matched the overwritten old assemblies.

Many of the changes to provide new features or enhancements were also first introduced and tested via program overwrites before including the changes in new assemblies and loads. This was not practical for some of the larger changes, which required the assembly of new modules. For these cases, the "binary load" facility of the 1A utility system was used to temporarily place the object module in program store for test purposes.

V. CONCLUSION

The 1A Processor is operating very satisfactorily in its first application in the No. 4 ESS office at Chicago, which was cut over on January 16, 1976. Experience in this office indicates that the design is sound and that objectives for ultrareliable system operation are being achieved. Much of the success of this project is credited to the extensive design verification and testing done at each level of the system's integration.

VI. ACKNOWLEDGMENTS

The authors are reporting on the work of many people in Bell Laboratories and Western Electric. They wish to acknowledge the efforts of all those who developed the extensive computer aids and test facilities as well as those who planned and carried out the testing and integration tasks.

REFERENCES

1. R. E. Staehler, "1A Processor: Organization and Objectives," B.S.T.J., this issue, pp. 119–134.
2. G. Haugk and S. H. Tsiang, "System Testing of the No. 1 Electronic Switching System," B.S.T.J., *43* (September 1974), p. 2575.
3. D. R. Barney, P. K. Giloth, and H. G. Kienzle, "No. 1 ESS ADF: System Testing and Early Field Experience," B.S.T.J., *49* (December 1970), p. 2975.
4. B. P. Donohue, III, and J. F. McDonald, "SAFEGUARD Data-Processing System: Process-System Testing and the System Exerciser," B.S.T.J., 1975, SAFEGUARD Supplement, p. S111.
5. A. E. Spencer, Jr. and H. E. Vaughan, "No. 4 ESS; a Full Fledged Toll Switching Node," International Switching Symposium, Kyoto, Japan, October 1976.
6. J. W. Nowak, "No. 1A ESS—A New High-Capacity Switching System," International Switching Symposium, Kyoto, Japan, October 1976.
7. J. O. Becker et al., "1A Processor: Technology and Physical Design," this issue, pp. 207–236.
8. H. W. Chang, G. W. Smith, and R. B. Walford, "LAMP: System Description," B.S.T.J., *53* (October 1974), p. 1431.
9. S. G. Chappel, "LAMP: Automatic Test Generation for Asynchronous Digital Circuits," B.S.T.J., *53* (October 1974), p. 1477.
10. A. H. Budlong et al., "1A Processor: Control System," B.S.T.J., this issue, pp. 135–179.
11. C. F. Ault et al., "1A Processor: Memory System," B.S.T.J., this issue, pp. 181–205.
12. G. F. Clement et al., "1A Processor: Control, Administrative and Utility Software," B.S.T.J., this issue, pp. 237–254.
13. P. W. Bowman et al., "1A Processor: Maintenance Software," B.S.T.J., this issue, pp. 255–287.

Glossary

ABL	Auxiliary buffer order word register left
ABR	Auxiliary buffer order word register right
ACK	Acknowledge lamp
ADS	Auxiliary Data System
ALV	Adder and logic unit
ALW	Allow
AMA	Automatic message accounting
ASCII	American Standard Code for Information Interchange
ASW	All seems well
AU	Auxiliary unit
ATG	Automatic test generation
AUB	Auxiliary unit bus
BOL	Buffer order word register left
BOR	Buffer order word register right
BPI	Bits per inch
BRB	Buffer read bus
BWB	Buffer write bus
CC	Central control
CDI	Collector diffusion isolation structure
CF	Control flip-flops
CPD	Central pulse distributor
CS	Call store
CSB	Call store bus
DGN	Diagnose
DMA	Direct memory access
DR	Disk request
DTTL	Diode-modified transistor-transistor logic
DUB	Data unit bus
DUS	Data unit selector
EIA	Electronic Industries Association
EMI	Electromagnetic interference
EX	Exercise mode
FIT	One failure in 10^9 component operating hours
FS	File store
FTF	First test failure

HIC	Hybrid integrated circuit
HWR	Halfword register
IO	Input/output unit
ITS	Installation test system
IOUC	Input/output unit controller
IOUS	Input/output unit selector
LAMP	Logic analyzer for maintenance planning
LSL	Logic simulation language
MADES	Machine-aided design system
MB	Masked bus
MCC	Master control console
MON	Monitor
MS	Monitor store
NVOP	Name match and valid operation
OS	Out of service
OW	Order word register
PA	Pattern analysis
PC	Processor configuration
PCDF	Power conversion and distribution frame
PMT	Permanent magnet twistor
PRR	Polling request register
PS	Program store
PPI	Processor-peripheral interface
PS	Program store
PSB	Program store bus
PSL	Program store left
PSR	Program store right
PU	Peripheral unit
RAM	Random access memory
REC	Recovery gate
RMV	Remove
RO	Read only
ROS	Requested out of service
RST	Restore
RTL	Resistor-transistor logic
SBC	Standard, junction-isolated, buried collector
SCR	Silicon-controlled rectifier
SD	Scan and signal distributor
SIC	Silicon integrated circuit
SR	System reinitialization
SWAP	Switching assembly program
TF	Tape frames

TLP	Trouble location procedure
TUC	Tape unit controller
UB	Unmasked bus
UC	Utility computer
UTC	Utility test console
VREF	Reference voltage

Contributors to This Issue

C. F. Ault, B.E.E.E., 1950, University of Southern California; M.S.E.E., 1955, Stevens Institute of Technology; Bell Laboratories, 1955—. Mr. Ault has been continuously involved in memory development. He has worked on the development of memory for processors, including the barrier grid store, the flying spot store, and the permanent magnet twistor. He presently supervises work on mass memory subsystems for ESS processors. Member, IEEE, Eta Kappa Nu, Tau Beta Pi.

J. O. Becker, B.S.M.E., 1960, M.S.M.E., 1961, and Ph.D., 1964, University of Illinois; Bell Laboratories, 1969—. Mr. Becker has been concerned with the thermal performance of the 1A Processor and associated technology. Since 1970, he has been supervisor of a group responsible for the physical design of memory for the 1A Processor. Member, Pi Tau Sigma, Tau Beta Pi, Sigma Xi.

P. W. Bowman, B.S.A.M.P., 1968, and B.S.E.E., 1968, University of Wisconsin-Madison; M.S., 1969, University of California-Berkeley; Bell Laboratories 1968—. Mr. Bowman has worked on specification and diagnostic design for the 1A Processor auxiliary data system. He designed a performance and evaluation system to be employed at the field trial for the High Capacity Mobile Telephone System. Since 1974, he has had responsibility for the 1A Processor diagnostics. At present, he is involved in the design and development of a voice storage system.

J. H. Brewster, B.S.E.E., 1960, Yale University; M.S.E.E., 1962, New York University; Bell Laboratories, 1960—1976. Mr. Brewster has worked on maintenance and operational software for No. 1 ESS. He has also supervised work on the design of man-machine interface and input/output subsystems for the 1A Processor. Mr. Brewster is now with American Bell International, Inc.

A. H. Budlong, B.N.S., 1946, B.E.E., 1948, and M.S. (Physics), 1950, Marquette University; Member of teaching staff of Department of

Physics 1948—1952, Marquette University; Bell Laboratories, 1952—. Mr. Budlong has engaged in exploratory development of electronic switching circuits. He has also worked on the development of the No. 1 ESS electronic telephone switching system in the areas of trunk and service circuits and automatic message recording equipment, in the design of high-speed scanners and signal distribution, and is currently in charge of a group designing high-speed communication buses and power facilities for a high-speed processor for electronic telephone switching systems. Member, Sigma Pi Sigma, Pi Mu Epsilon.

J. G. Chevalier, B.E.E., 1951, Ohio State University; Bell Laboratories, 1956—. Mr. Chevalier has worked on printed-wiring processes and applications for military projects, connector design, studies of contact finishes, and the design and packaging of electronic equipment for No. 1 ESS. His current project is the physical design of equipment and interconnection techniques for the 1A Processor. He is currently supervisor of the Interconnections Group in the Processor Development Laboratory.

G. F. Clement, B.S.E.E., 1940, New York University; M.A. (Physics), 1947, Columbia University; Bell Laboratories, 1934—. Mr. Clement has worked on electronic field studies and solid solutions. During World War II he worked on the Manhattan Project and the application of analog computers to military systems. He has worked on military missile systems, the UNICOM system maintenance, and No. 1. ESS AUTOVON. In 1968, he became head of a department responsible for No. 1 ESS maintenance and 1A Processor recovery. He is currently head of a department responsible for Step-by-Step and No. 1 Crossbar design and support. Member, Iota Alpha, Eta Kappa Nu.

B. G. De Lugish, B.S.E.E., 1965, M.S.E.E., 1968, and Ph.D., 1970, University of Illinois-Urbana; Bell Laboratories, 1965—. Mr. De Lugish has been engaged in hardware development for the 1A Processor. He has worked with processor architecture and the design of processors for electronic switching systems. Member, IEEE, Tau Beta Pi, Sigma Tau, Eta Kappa Nu.

M. R. Dubman, A.B., 1957, Princeton University; M.S., 1959, Massachusetts Institute of Technology; Ph.D., 1970, University of California-Los Angeles; Bell Laboratories, 1970—. From 1959 to 1970, Mr. Dubman was engaged in the development of data analysis techniques used in testing of Saturn/Apollo rocket engines for North American Rockwell. At Bell Laboratories, he has been concerned with the development of error analysis methods for the 1A Processor.

R. K. Eisenhart, B.S.M.E., 1957, Pennsylvania State University; M.E.E., 1959, New York University; Bell Laboratories, 1957—. Mr. Eisenhart has worked on the development of the photographic processor for the flying-spot store of the Morris electronic switching system. He has supervised the development of apparatus and equipment used in No. 1 ESS. In 1966, he was appointed head of a department responsible for the development of packaging technology for electronic switching systems and for the physical design of the 1A Processor.

J. H. Forster, B.A., 1944 and M.A., 1946, University of British Columbia; Ph.D., 1953, Purdue University; Bell Laboratories, 1953—. Mr. Forster has worked on transistor development and reliability and in semiconductor surface studies. He has supervised the development of computer and microwave diodes and integrated circuits. Since 1963, he has headed departments engaged in various aspects of silicon and hybrid integrated-circuit development, including bipolar logic circuits for 1A Processor, No. 4 ESS, and many other Bell System applications. Member IEEE, Sigma Pi Sigma, Sigmi Xi.

A. W. Fulton, B.S.E.E., 1966, University of Arizona; M.S.E.E., 1967, and Ph.D. 1971, Stanford University; Bell Laboratories, 1966—. Mr. Fulton has worked on the development of technology, circuits, and file-memory systems for electronic switching systems. In 1972, he was appointed supervisor of a file-store memory development group. He is presently supervisor of a group involved in the development and application of integrated circuits for switching systems. Member, Tau Beta Pi, Sigma Xi.

P. S. Fuss, B.S.E.E., 1956, University of Michigan; M.E.E., 1960, New York University; Bell Laboratories, 1958—1977. Mr. Fuss has worked on circuit and system development on various missile guidance and antisubmarine warfare systems. In 1969, he was appointed head of a department responsible for the development of a digital signal processor system for an advanced ASW program. He was head of the Large Processor Design Department which is responsible for the design of large switching control processors and their associated operating system software. He is currently Director of Product Development, Teletype Corporation. Member, IEEE, Tau Beta Pi, Eta Kappa Nu.

F. M. Goetz, B.E.E., 1953, Manhattan College; M.S. (Math), 1960, New York University; Bell Laboratories, 1953—. Mr. Goetz has worked on logic design, software development, and system design for electromechanical and electronic switching systems. Since 1962, he has been a supervisor responsible for various aspects of electronic switching system maintenance. At present, he is responsible for small processor system design and maintenance. Member, IEEE, Eta Kappa Nu.

T. S. Greenwood, B.S.E.E., 1951, Northeastern University; M.S.E.E., 1953, Massachusetts Institute of Technology; Bell Laboratories, 1953—. Mr. Greenwood has been continuously involved in development of electronic switching system processors. His early work involved the development of memory for processors, including the barrier grid store, the flying spot store, and the permanent magnet twistor. He has worked on the development of both maintenance and operational programs for No. 1 ESS. He currently heads a department responsible for the development of the 1A Processor hardware. Member, IEEE, ACM, AAAS.

R. J. Griffith, B.S.E.E., 1963, Iowa State University; M.S.E.E., 1965, Ph.D., 1969, Columbia University; Bell Laboratories, 1963—. Mr. Griffith has worked on the design of information processing techniques in systems used by the U.S. Navy. Since 1974, he has been concerned with development and evaluation of software for the 1A Processor.

R. E. Haglund, B.S.E.E., 1960, M.S.E.E., 1963, and PhD., 1969, Iowa State University; Bell Laboratories, 1969—. Mr. Haglund was a member of the original 1A Processor file-store design group and presently is engaged in file-store design utilizing new memory technologies. Member, IEEE, Eta Kappa Nu, Sigma Xi.

W. L. Harrod, B.S.M.E., 1962, Texas Tech University; M.S.E.M., 1964, Rutgers University; M.S.M.S., 1971, and Ph.D., 1975, North-western University; Bell Laboratories, 1962—. Mr. Harrod has worked on the physical design of hardware for government and military switching systems and on the development of technology for applying thin-film and silicon integrated circuits in electronic switching systems. Since 1968, he has been supervisor of a group responsible for design, development, and characterization of new hardware for integrated-circuit packaging in ESS. His current responsibilities also include the physical design of advanced file memory systems. Member, Tau Beta Pi.

H. A. Hilsinger, B.S.E.E., 1954, Newark College of Engineering; M.S.E.E., 1959, New York University; Bell Laboratories, 1954—. Mr. Hilsinger's early work included military missile projects. Since 1961, he has supervised groups associated with the physical design and construction program for electronic switching systems.

R. F. Kranzmann, B.E.E., 1969, Union College; M.E.E., 1962, New York University; Bell Laboratories, 1960—. Mr. Kranzmann has worked on the design and development of call-processing software for military and government communication systems (UNICOM and AUTOVON). From 1969 through 1971, he supervised a group responsible for maintenance planning and diagnostic software design for No. 1 ESS peripheral units. More recently Mr. Kranzmann has had responsibility for diagnostic software design for several 1A Processor subsystems and for design of both on-line and off-line software to implement trouble-location procedures for the 1A Processor.

R. C. Lee, B.S.E.E., 1948, University of Michigan; M.S.E.E., 1957, Polytechnic Institute of Brooklyn; Bell Laboratories, 1948—. Mr. Lee has been involved with the development of switching systems including Crossbar Tandem, 101 ESS, No. 1 ESS, and the 1A Processor. He is currently supervisor of a group responsible for 1A Processor fault-recovery programs.

K. D. Mozingo, B.S.E.E., 1963, North Carolina State University; M.S.E.E., 1964, University of Michigan; Bell Laboratories, 1963—. Mr. Mozingo has been involved in software and system test development for switching systems (No. 1 ESS, No. 1 ESS ADF, 1A Processor). He is presently a supervisor in the Processor Applications and Software Department of the Processor Development Laboratory. Mr. Mozingo's group is responsible for the diagnostic control and trouble-location programs in the 1A Processor. Member, Phi Kappa Phi, Theta Tau, Eta Kappa Nu.

S. M. Neville, B.S.E.E., 1959, University of Oklahoma; M.E.E., 1961, New York University; Bell Laboratories, 1959—. Mr. Neville has worked on circuit, logic, and system design for the No. 1 ESS processor and 1A Processor. He presently is supervisor of the Processor Design Group in the Processor Development Laboratory. Member, IEEE, Sigma Tau, Tau Beta Pi, Eta Kappa Nu.

J. S. Nowak, Bell Laboratories, 1955—. Mr. Nowak has worked in the system planning area of the Morris experimental electronic central office. He later supervised a group developing system requirements and designing maintenance programs for No. 1 ESS. He presently heads a department responsible for the planning, design, and introduction of a high-capacity No. 1A Electronic Switching System.

J. L. Quinn, B.S.E., 1959, Stevens Institute of Technology; M.S.E.E., 1961, New York University; Bell Laboratories, 1959—. Mr. Quinn has worked on the design and testing of the No. 1 ESS. Since 1965, he has been supervisor of groups concerned with ESS system testing and development of maintenance programs. At present, he is responsible for the 1A Processor. Member, IEEE, Eta Kappa Nu.

W. A. Read, A.B., 1960, Bowdoin College; M.S.E.E., 1962, Columbia University; Bell Laboratories, 1964—. Mr. Read was initially involved in the design of logic circuits for No. 101 and No. 2 ESS. Subsequently, he has been engaged in the design and development of memories for the No. 1 ESS and the 1A Processor. Member, IEEE.

M. W. Rolund, B.E.E., 1961, Cooper Union; M.S.E.E., 1963, New York University; Bell Laboratories, 1961—. Mr. Rolund is supervisor of a group engaged in circuit and system design of magnetic and semiconductor memories for No. 1 ESS and the 1A Processor. Member, IEEE, Tau Beta Pi.

R. D. Royer, A.A.S., 1962, College of William and Mary; B.S.E.E., 1966, West Virginia University; M.S.E.E., 1967, University of Michigan; Bell Laboratories, 1966—. Mr. Royer has been involved with the maintenance of the No. 1 ESS, 1A Processor, and No. 4 ESS systems. He is currently supervisor of the No. 4 ESS Digital Terminal Recovery Group.

R. E. Staehler, B.S.E.E., 1947, The College of the City of New York; M.S.E.E., 1948, Polytechnic Institute of Brooklyn; Bell Laboratories, 1948—. Mr. Staehler's early work was on No. 5 Crossbar, toll signaling systems, and trainers for guided missile systems. In 1953, he worked on the development of electronic switching systems, specifically, the processor memory for the experimental central office in Morris, Illinois, and the processor logic and call memory for No. 1 ESS. He was appointed Director of the Electronic Switching Projects Laboratory in 1964 with responsibility for special applications for No. 1 ESS to military and data networks, including No. 1 ESS AUTOVON. In 1968, he was appointed Director of the Electronic Systems Design Laboratory with responsibility for development of the 1A Processor. In 1976, he was appointed Director of the Operator Services Laboratory with responsibility for extending the automation of operator services. Member, IEEE, Eta Kappa Nu, Tau Beta Pi, Sigma Xi.

C. F. Starnes, B.S.E.E., 1964, Oregon State University; M.S.E.E., 1965, Oregon State University; Bell Laboratories, 1965—. Mr. Starnes has worked on integrated-circuit design, test-facilities development and electromagnetic compatability. He is a supervisor in the Field System Department, responsible for the development of system acceptance tests, generic program retrofit, and system hardware growth.

E. H. Stredde, B.S.E.E., 1966, and M.S.E.E., 1967, University of Illinois; Bell Laboratories, 1967—. Mr. Stredde has worked on No. 1 ESS software development for pheripheral fault recovery, trunk maintenance, and software integrity verification, and transient memory initialization. He has alwo worked on 1A Processor software design of system-recovery programs and is currently working on software design of international call-processing features for No. 4 ESS. Member, Eta Kappa Nu, Tau Beta Pi.

G. A. Van Dine, B.S.E.E., 1956, Purdue University; Bell Laboratories, 1956—. Mr. Van Dine initially worked on digital circuit designs for NIKE Missile Systems, and subsequently for delay line storage systems and magnetic tape stores on the UNICOM and No. 1 ESS ADF Systems. He later worked on cathode ray tube display system techniques. Since 1967, he has supervised the development of computer-driven test facilities for 1A Processor frames, and the utility system used for monitoring and control of the 1A Processor in the system laboratory environment. He subsequently supervised a group responsible for No. 4 ESS reliability studies.

R. J. Watters, B.M.E., 1948, Cornell University; Bell Laboratories, 1948—. Mr. Watters has worked on the design of several military systems. In 1965, he was appointed head of a department responsible for the design of maintenance for the No. 1 ESS AUTOVON system. Since 1968, he has had the responsibility for design of diagnostic programs for the 1A Processor. He currently is Head of the Processor Application and Software Department which has the software responsibility for the 3A Processor. Member, Association of Computing Machinery, Tau Beta Pi.

P. W. Wendland, B.S.E.E., 1965, and M.S.E.E., 1966, Cornell University; Bell Laboratories, 1966—. Mr. Wendland has been involved in subsystem and system testing of the 1A Processor. He presently supervises a group responsible for design of 1A Processor man-machine interface circuits and teletypewriter data link circuits.

Abstracts of Papers by Bell System Authors Published in Other Journals

CHEMISTRY

Brillouin Scattering and Hypersonic Relaxation in Amorphous Polymers. G. D. Paterson, Polym. Preprints, *17* (August 1976), pp. 13–16. Hypersonic relaxation is studied in amorphous polyisobutylene, atactic polypropylene, atactic polyvinyl acetate, atactic polystyrene, and bisphenol-A polycarbonate. The hypersonic loss tan δ is measured as a function of temperature by Brillouin spectroscopy. Only a single loss maximum is observed for the polymers studied in this work. The loss maximum correlates well with other mechanical relaxation data.

COMPUTING

New and Simple Light-Pen Detection Technique for Interactive Plasma Display Systems. P. D. T. Ngo, IEEE Trans. Electron. Dev. 23, No. 9 (September 1976), pp 1058–1063. Two interrogating pulses are applied to an addressed cell of a plasma panel. If in the "on" state the cell is flashed once by the first pulse; if "off", by the second. Cell state is not altered. A light pen is gated to look for a flash to establish pen location and cell state.

ELECTRICAL AND ELECTRONIC ENGINEERING

Direct Observation of the Optical Plasma Resonance of Ag by Photon-Assisted Tunneling. R. K. Jain, M. G. Farrier*, and T. K. Gustafson*, Phys. Rev. Lett., *36*, No. 8 (1976), pp. 435–438. Photon-assisted tunneling is proposed for the investigation of optical plasma resonances; this is demonstrated by the direct observation of the Ag resonance in appropriate Ag-$A\ell_2O_3$—$A\ell$ structures. A spectral scan of the ratio of the signals photo-induced by the p- and s-polarizations shows a large enhancement corresponding to the plasma resonance of Ag. *University of California, Berkeley.

Encapsulation of Integrated Circuits Containing Beam Leaded Devices with a Silicone RTV Dispersion. D. Jaffe, Proc. 1976 Electron. Compon. Conf. (April 1976), pp. 376–381. A number of evaluations of a xylene based RTV silicone rubber dispersion for the encapsulation of integrated circuits containing beam leaded silicon integrated circuit (SIC) devices are described. These include: (1) material properties, (2) application techniques, and (3) performance in some initial accelerated bias-humidity and temperature cycling tests.

Discharge Pumping of Dye Vapors. P. W. Smith, P. F. Liao and P. J. Maloney, IEEE J. Quantum. Electron., *12*, No. 9 (September 1976), pp. 539–542. We report the results of a study of discharge-excited dye vapors. For 20 dyes, fluorescence output was measured as a function of buffer gas, temperature, and discharge parameters. Although fluorescence efficiencies as high as 6% were obtained at low power, saturation effects limited the output fluorescence to values corresponding to gains of less than 0.2 cm^{-1}

Distributed-Feedback Color Center Lasers in the 2.5 to 3.0 μm Region. G. C. Bjorklund, L. F. Mollenauer and W. J. Tomlinson, Appl. Phys. Lett., *29*, No. 2 (August 15, 1976), pp. 116–118. We have demonstrated pulsed distributed-feedback laser action in KCl:Li containing $F_A(II)$ color centers with spatially modulated concentrations. Laser outputs at various wavelengths between 2.6 and 2.8 μm have been observed with linewidths narrower than 0.2 nm. The absorbed pump power at threshold was 1.5 kW, and efficiencies of up to 6.7% were measured.

Lineshape and Strength of Two-Photon Absorption in an Atomic Vapor with a Resonant or Near-Resonant Intermediate State. J. E. Bjorkholm and P. F. Liao, Phys. Rev., *14*, No. 2 (August 1976), pp. 751–760. We experimentally and theoretically consider two-photon absorption in a vapor with a resonant intermediate state. It is shown that maximum absorption rates and Doppler-free linewidths are *simultaneously* obtained. The experimental dependence of the lineshape and absorption rate upon mistuning from the intermediate state is in good agreement with theory.

Photodecomposition in Popop Dye Vapor Lasers. P. F. Liao, P. W. Smith and P. J. Maloney, Opt. Commun., *17*, No. 3 (June 1976), pp. 219–222. Experimental evidence is presented which shows that the premature termination of dye vapor laser gain observed in optical pumping experiments is due to photodecomposition. Addition of up to 4 atm of buffer gases was found to have little effect.

A Tester Independent Program Language. R. P. Davidson, Digest of Papers: 1976 Semicond. Test Symposium (October 1976), pp. 76–77. An integrated circuit (IC) designer writes test procedures for his devices in tester independent format (TIF) without being required to have detailed knowledge of the target system. Test requirements are defined in terms of device pins, stimuli (i.e., current/voltage), and timing. A language translator converts this information to a test program for a target machine.

MATERIALS SCIENCE

Brillouin Scattering from Poly(Vinylidene Fluoride)-Poly(Methyl Methacrylate) Mixtures. G. D. Patterson, T. Nishi* and T. T. Wang, Macromolecules, *9* (1976), pp. 603–605. The polymer blend of poly(vinylidene fluoride) and poly(methyl methacrylate) has been studied with Brillouin spectroscopy. The mixtures are shown to be compatible in the melt. Quenched films display a single glass-rubber relaxation temperature. The crystallization and melting behavior of the films was also studied above T_g.
* Bridgestone Tire Co., Tokyo, Japan.

The Conformational Characteristics of Poly(Vinylidene Fluoride). A. E. Tonelli, Macromolecules, *9* (1976), pp. 547–551. Approximate energy estimates have been utilized to evaluate the conformational characteristics of poly (vinylidene fluoride) (PVDF) including its dimensions, dipole moments, their temperature dependence, and the conformational contribution to the entropy of fusion. Calculated results were insensitive to small amounts of reverse monomer addition, but did depend markedly upon the value selected for the dielectric constant (ϵ). Agreement between calculated and observed properties was possible for $4 < \epsilon < 6$.

The Effect of Adsorbed Gases and Temperature on the Photovoltage Spectrum of GaAs. S. C. Dahlberg, J. Vac. Sc. Technol., *13* (September/October 1976), pp. 1056–1059. The photovoltage spectrum for GaAs (100) after Ar bombardment and with a variety of adsorbed gases (O_2, NH_3, NO, SO_2, H_2S, CO_2) has been measured. The above bandgap photovoltage shows structure which is surface sensitive, and this structure is tentatively attributed to transitions involving surface states. Heating causes a transient loss of photovoltaic activity.

Environmental Testing with Giant Aerosols: Exposure of Miniature Relays. T. E. Graedel, J. P. Franey, and R. E. Schwab, Elec. Contacts/1976, *25* (1976), pp. 47–55. Techniques have been devised for generating chemically inert giant (i.e., $d > 20\mu m$) aerosols, and for exposing relay components to aerosol concentrations ~30 times those typically encountered in field operation. The successful relay performance indicates that physical contact blockage may be avoided in field operations by appropriate air filtration and housing design.

Fermi Surface Measurements in Normal and Superconducting 2H-$NbSe_2$. J. E. Graebner and M. Robbins, Phys. Rev. Lett., *36* (February 1976), pp. 422–425. Landau quantum oscillations are observed for the first time in the layered, superconducting, incommensurate charge density wave compound 2H-$NbSe_2$. The oscillations are observed clearly above H_{c2} but also *below* H_{c2}, with somewhat increased scattering. This is interpreted as scattering of the orbiting normal electrons from the flux lattice. The observed sheet of Fermi surface can be explained if Mattheiss' APW band structure is modified slightly.

NO Line Parameters Measured by CO Laser Transmittance. R. E. Richton, Appl. Opt., *15* (July 1976), pp. 1686–1687. By measuring the transmittance of two CO laser lines through NO and NO with N_2 added, and assuming an extinction coefficient given by a Voigt profile, line parameters for each Λ-doubled component are determined. These parameters can be applied to remote atmospheric sensing and to find chemical reaction rates.

Paramagnetic States and Hopping Conductivity in a Chalcogenide Glass: As_2Te_3. J. J. Hauser and R. S. Hutton, Phys. Rev. Lett., *37* (September 27, 1976), pp. 868–871. Variable-range hopping as evidenced by a resistivity proportional to exp $T^{-1/4}$ and by the two-dimensional behavior of thin films has been observed for the first time in a chalcogenide glass. This was accomplished by sputtering As_2Te_3 at 77°K. The localized paramagnetic hopping states have also been established by the existence of an ESR signal.

Polyethylene and Polytetrafluoroethylene Crystals: Chain Folding, Entropy of Fusion, and Lamellar Thickness. A. E. Tonelli, Polymer, *17* (1976), pp. 695–698. Chain folding in polyethylene (PE) and in polytetrafluoroethylene (PTFE) crystallites is simulated on the computer. Adjacently re-entering chain folds are found to be much more easily formed in PE than in PTFE. However, the relative energies required to create a unit area of chain fold are estimated to be nearly the same for both polymers and cannot be the source of the large difference in their lamellar thicknesses.

Spectroscopic Study of RF Oxygen Plasma Stripping of Negative Photoresists. I. Ultraviolet Spectrum. E. O. Degenkolb, C. J. Mogab, M. R. Goldrick and J. E. Griffiths, Appl. Spectrosc., *30* (September 1976), pp. 520–527. The stripping of photoresists from a silicon wafer using an rf oxygen plasma has been monitored using the optical emission from electronically excited OH and CO species in the ultraviolet region of the spectrum. The endpoint of plasma stripping and the amount of stripped material is easily determined quantitatively.

Sulfur Dioxide, Sulfate Aerosol, and Urban Ozone. T. E. Graedel, Geophys. Res. Lett., *3* (1976), pp. 181–184. Kinetic chemical computations for the suburban New Jersey troposphere indicate the daily production of $\sim 1\ \mu g/m^3$ of sulfate aerosol, largely through sulfuric acid, and demonstrate that SO_2 and its products are slight inhibitors to ozone production.