THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 62, No. 6, July-August 1983
Printed in U.S.A.

Modernization of the Suburban ESS:

Overview: Evolution of the Suburban ESS

By T. E. GRASSMAN* and J. E. YATES*

(Manuscript received June 8, 1982)

This paper highlights the systematic evolution of the No. 2/2B Electronic Switching System (ESS), describing both hardware and software changes. It also introduces the subsequent articles that describe the modern interfaces and modern development environment enjoyed by the system. The No. 2 ESS was the first electronic switching system specifically designed for the suburban community. Subsequent incorporation of a more flexible and powerful processor provided a basis for a large increase in processing capacity and a continuing modernization of system structure. The evolved system, the No. 2B ESS, is now rich in features and offers modern administrative and maintenance capabilities.

I. INTRODUCTION

The No. 2 Electronic Switching System (ESS),[1] designed in the 1960s to provide ESS capabilities to suburban communities, was developed under the same constraints as other large real-time control systems of that era. Memory was expensive and processors were not powerful enough to compensate for inefficiencies in software. As a

* Bell Laboratories.

©Copyright 1983, American Telephone & Telegraph Company. Copying in printed form for private use is permitted without payment of royalty provided that each reproduction is done without alteration and that the Journal reference and copyright notice are included on the first page. The title and abstract, but no other portions, of this paper may be copied or distributed royalty free by computer-based and other information-service systems without further permission. Permission to reproduce or republish any other portion of this paper must be obtained from the Editor.

result, programs were written in assembly language and were tightly coupled. Compared with No. 1 ESS, then being deployed in urban areas, the No. 2 ESS faced a more stringent cost sensitivity due to the smaller number of subscribers supporting a given local switching system.

The original No. 2 Processor was replaced in 1976 with the 2B Processor, which has a more modern architecture based on improved semiconductor technology. This more powerful processor provided capacity for feature enhancement[2] and allowed utilization of more generalized and structured software techniques that have subsequently been incorporated into the system. This incorporation was accomplished by strictly controlling the portions of the software system that were allowed to change. Since that time the programs have continued to evolve and exploit the capabilities of the 2B Processor.

Most recently, the inclusion of modern software constructs in the 2BE3 generic has produced a design containing a modern data link interface that supports such capabilities as the 10A Remote Switching System (RSS), Automatic Message Accounting (AMA) data teleprocessing, and traffic data teleprocessing. Description of these features may be found in the companion articles of this series.

The evolutionary approach used to realize the present No. 2B ESS has resulted in a continuously viable product that remains economically attractive and modern in feature content. This paper will discuss the system evolution, and serve as an introduction to the other papers in this series.

II. HARDWARE AND SYSTEM CHARACTERISTICS

2.1 No. 2 ESS

The original No. 2 ESS central processor was designed using Transistor Resistor Logic (TRL) in the early 1960s. The architecture consisted of special purpose registers that were designed to accommodate a set of instructions tailored for use in telephone switching. For example, explicit instructions were included to support the software structure that used 8-word control blocks to process each telephone call. Its architecture and instruction set gave the No. 2 ESS the capability to efficiently switch telephone calls in a small to medium (2,000- to 20,000-line) office.

The hardware architecture used in the No. 2 ESS Processor supported two types of stores (memories). One of the stores was a permanent magnet twistor that contained the program as well as the data which specified the characteristics of lines and trunks. This 22-bit-wide memory was magnetically alterable off-line. The memory organization consisted of four 64K modules for a maximum of 256K

words. The selection of a module was independent of the normal addressing mode within a module. This required each program to independently administer the module selection for itself. The other store contained up to 32K 16-bit words and was on-line writable. It contained dynamic information related to the processing of calls.

An additional characteristic of No. 2 ESS was a wired logic unit that autonomously scanned for customer requests, and, under timed control, scanned for short-interval signals. Both of these logic control items deloaded the program and increased the efficiency of the overall system. Scanners and other peripheral units were controlled by a parallel bus system.

This processor was put into service in 1970 in the No. 2 ESS system and provided the desired service level, features, and cost benefits at the time of its initial cutover.

2.2 The No. 2B ESS

2.2.1 Background

In 1971, a series of cost reduction studies led to the selection of the more powerful 3A Processor Control Complex (3A CC)[3] for incorporation into the No. 2 ESS. The 3A CC was designed for use in both No. 2B ESS and No. 3 ESS. The 3A CC was designed along general-purpose computer principles rather than on the specific switching logic instructions incorporated in the original No. 2 ESS Processor, and made extensive use of integrated circuits. The new processor could address up to one million words of relatively inexpensive semiconductor memory, provided a general-purpose instruction set and a serial bus access to peripheral units, and used a microprogram-based instruction decoder.

The principal asset of the 3A CC Processor was its microprogrammed instruction set. This feature allowed the processor to not only encompass the No. 3 ESS commands, but also to be extended in a relatively simple manner to emulate the original No. 2 ESS instruction set, and to add additional instructions that were parallel in nature to the No. 2 ESS set. This allowed the retention of most of the programs that had previously been developed to provide features and maintenance on the existing No. 2 ESS (except for the processor maintenance itself).

To minimize changes in the control of major peripheral equipment, a new interface [the 2B Input/Output (I/O) unit] was developed to connect the 3A CC to the parallel peripheral bus system. (The combination of the 3A CC, the 2B I/O unit, the increased memory, and the microcode for emulation is referred to as the 2B Processor.) In addition, programs already developed for the 3A CC, e.g., teletype-

writer (TTY) handlers and processor maintenance software, were adopted intact to avoid duplicate software development.

2.2.2 *Instruction emulation*

The key element in replacing the No. 2 Processor was the emulation of the existing No. 2 ESS instruction set. The existing 22-bit data and instruction format of the No. 2 ESS did not fit in the 16-bit 3A CC memory. Converting each No. 2 ESS word to two 3A CC words would have resulted in an addressing incompatibility between the old and new systems, i.e., the same instruction would exist at different addresses in the two systems. If this were allowed to occur, object-level changes to correct field problems would have to be developed twice, once for the No. 2 ESS and again for the No. 2B ESS. By keeping the object-level addresses synchronized, this duplicate effort was avoided. The decision was made to expand the 3A CC store to 24 usable bits to allow exact emulation of No. 2 ESS instructions and data. Half-word instruction pairs could be accommodated in the same addressable word, and data packing was then identical in the two processors. A further decision was made to use the No. 2 ESS instruction manual as the requirements for the emulated instructions and to map the functional No. 2 ESS registers into the general-purpose registers of the 3A CC. Each register and instruction was fully emulated.

These restrictions gave several distinct advantages although they did not allow improvements in program techniques through use of the richer 3A CC instruction set. The advantages were as follows: a test program that previously tested the instruction logic on No. 2 ESS provided the acceptance test for the microprogrammed instruction set (i.e., no new program was required); an immediate check on correct program compilation was available by comparing raw program sizes, symbol values, and linkage characteristics between the existing and the emulated programs; debugging of these programs was required only on one machine (No. 2 or 2B Processor) since they could be automatically assembled from the same source for the other machine without human interference (this allowed nearly simultaneous program releases on the two different systems); and most existing documentation and procedures for the No. 2 ESS were usable with the 3A CC since most user interfaces were identical between the systems.

With these constraints, the bulk of No. 2 ESS programs (85 percent) could be used as they were currently written. Most of the remaining 15 percent were No. 2 ESS Processor maintenance programs, which were replaced in total by the 3A CC Processor maintenance programs. The programming effort of the 2B introduction then was completely restricted to the support programs (assemblers, loaders, etc.), six

interface programs, and new diagnostics and recovery programs for the wide store.

2.2.3 Input/output emulation

The No. 2 ESS wired logic I/O performed two functions. It looked for new customer service requests and collected the digits as they were dialed by the customer. In the new system, software was used to scan for service requests. The digit collection functions were exactly duplicated by an interrupt-level microcode sequence on the 3A CC. This provided an unchanged interface to existing programs that had depended on the wired logic. The only associated changes required were in recovery programs for handling digit collection failures.

The interface between the 3A CC and the No. 2 peripheral equipment was provided by the 2B I/O unit. This was implemented to be driven by the existing 3A CC serial channel interface. Driver programs were implemented in microcode so that controlling programs, issuing I/O instructions, were not required to change.

2.3 No. 2B ESS field introduction and evolution

The No. 2B ESS was successfully put into service in 1976. Less than one year later the same program was used to provide service in an office requiring increased capacity through a retrofit of the processors. Since the 3A CC is much faster than the No. 2 ESS Processor, the No. 2B ESS call capacity (35,000 calls/hour engineered load) is much greater than that of the No. 2 ESS (16,500 calls/hour engineered load). By 1977, No. 2 ESS Processors were no longer being manufactured.

The No. 2/2B ESSs continued to evolve gradually—and in step—for several years while their programs were kept locked together by the restrictions discussed above. As the number of in-service No. 2 ESSs began to decline through processor retrofits, and the pressure to provide new and expanded features increased, a decision was made to discontinue new development on the No. 2 ESS, and to take further advantage of the 3A CC Processor's capabilities in No. 2B ESS.

The 2BE3 generic program, put into service in November of 1981, utilizes significant software restructuring in certain areas. The resulting advantages range from achieving expanded capacities for per-line features (e.g., call forwarding) to the development of several major new features and the communications interface that supports them.

2.4 A modern data communications interface

The need for data communications between local switching offices and other systems (remote switches and various operation support

systems) has expanded in recent years. To meet this increasingly important need, the No. 2B ESS has incorporated, in the 2BE3 generic program, a modern data communications interface capability. This interface is based on the Serial Peripheral Unit Controller/Data Link (SPUC/DL), a cost-effective and flexible hardware element utilizing microprocessor control capable of supporting various communications protocols. The 3A CC serial channel I/O capability provides the high-speed communication path between the host and the SPUC/DL. An accompanying article, "A New Data Link Controller," describes the SPUC/DL and its architecture, design, and capabilities in considerable detail.

Firmware has been developed for the SPUC/DL to support levels 1 and 2 of the BX.25 data communications protocol. This capability is utilized in provision of interfaces to the Automatic Message Accounting Collection System (AMACS) and to the Engineering and Administrative Data Acquisition System (EADAS). Firmware has also been developed to support the RSS protocol, which predated the BX.25 standard, enabling the No. 2B ESS to provide control of the 10A RSS. The features associated with these interfaces are also described later in this series.

III. NO. 2B ESS SOFTWARE EVOLUTION

Soon after the No. 2B ESS was successfully cut into service, an effort was begun to exploit the capabilities of the new processor. This effort, which had been planned from the beginning, aimed at eliminating data and program constraints. The success of these initial enhancements led to other incremental improvements. The program has matured to the point that enhancements driven by modern software practices, as well as the 2B Processor's architecture, are being pursued. These include operating-system-type primitives, data independence, and program decoupling.

This section will review some enhancements that demonstrate progress in a variety of areas. The techniques used are not new nor innovative by current standards. What is perhaps unique is the manner in which they have been applied to a large, complex, existing system.

The enhancements have been pursued in a systematic fashion. First, the software was extended to utilize the capabilities of the 2B Processor while still remaining functionally compatible with the original No. 2 ESS software. Commercial versions of these compatible programs were released in 1976 and 1977. Eventually the compatibility restriction was lifted and the software was extended to exploit noncompatible 2B capabilities. In about the same time frame, modifications to enhance functional and data independence were added.

3.1 2/2B compatible extensions

Functional compatibility between the No. 2 ESS and the No. 2B ESS at the time of conversion was important not only to minimize risk, but also to ensure that future enhancements applied to both systems. The types of enhancements that could be introduced were severely limited by this restriction. Generally, they took the form of increasing the maximum quantity of a particular resource that the system could handle.

A prime example is the memory spectrum. Immediately following the conversion, the 2B Processor was limited to the 256K-word spectrum of the No. 2 Processor simply because the software was tailored to handling 18-bit addresses. The program modifications required to handle 20-bit data addresses and exercise the full megaword spectrum of the 2B Processor were scattered throughout the software. A pervasive change of that nature with its attendant risk is precisely the kind of complication that was intentionally avoided during the initial conversion process. With the converted system functioning well, exploiting the million-word address spectrum could be attacked as a separate problem. It was scheduled, designed, and implemented in 1976.

The implementation modified all affected programs to be able to handle 20-bit addresses. Since both the 2 and 2B versions were modified, they remained compatible and both were able to handle the larger addresses. The No. 2 version would, of course, never encounter an address that exceeded 256K.

Similar extensions followed in a well-defined and controlled manner. The maximum number of scanners, peripherals used to sense the states of circuits and/or lines, was increased from 12 to 31. The number of buffers utilized to control peripheral units was increased from 12 to 20. Each extension was scheduled and implemented separately, thereby avoiding the coincident introduction of complex changes.

3.2 Noncompatible extensions

The number and nature of compatible extensions are limited. To make more fundamental structural changes to the software system required exploiting aspects of the 2B Processor for which there is no No. 2 counterpart. As an example, one could not begin to use the superior 2B instruction set since the No. 2 Processor did not contain those instructions. In like manner, the larger No. 2B ESS memory could not be utilized for program instructions. In 1977, the decision was made to decouple the new and the old systems, thereby eliminating the compatibility constraint on the program system.

This decision opened new vistas for improving the structure of the software system. The first step was to simply eliminate methodology

and operations that existed only because of the No. 2 Processor. For example, the No. 2 Processor cannot support relocatable programs. The 2B software system has now been converted to be mostly relocatable. Certain instruction sequences can be coded more efficiently using the 2B instruction set. Some sequences, in fact, were not needed at all and were simply eliminated. We were particularly fortunate to have available the excellent text processing facilities provided by the *UNIX** time-sharing system to aid in the search for these sequences.[4] These changes were collectively called program "clean up" and were performed as the first step in moving primary system development to the No. 2B ESS.

3.3 Modernizing the control structure

3.3.1 Isolating application software from the control programs

One definition of an operating system is a set of standard functions that provide a hospitable environment for application programs. The call processing control mechanism in the No. 2B ESS did not supply this feature. Nevertheless, certain steps toward providing common, general functions for use by application programs were possible. Emphasis was on the introduction of these functions with a minimal perturbation to the overall system. In particular, designs that did not cause existing programs to become inoperative were selected for the implementation of these functions. Once a function is available, all new development can use it. On an independent schedule and subject to independent evaluation, a plan could be developed to convert existing application programs to use the new functions.

3.3.2 Isolating application software from a resource manager

This philosophy of gradual introduction has also been used to reduce the coupling between an existing resource manager and the application programs. The manager is responsible for finding and allocating paths through the switching network. To efficiently use the network, it attempts to share a portion of an existing path with the next path to be found for the same call. The shared portion is known as the A link and the operation is known as A-link sharing. As originally designed, the manager provided the mechanism, but the application programs were required to save the identity of any A link to be shared and to specifically request sharing when invoking the manager.

Consistent with this new philosophy, a universal mechanism for sharing A links has now been implemented. The path manager now has complete control over the sharing mechanism. Sharing is at-

---

* Trademark of Bell Laboratories.

tempted in every situation and not just at the discretion of the application program. The A-link sharing information has been eliminated from the application interface. The significant point is that once again this was accomplished in a fashion that did not require the wholesale change of existing programs. Unchanged application programs would still go through the motions of making A-link sharing decisions, but these would be ignored by the manager. In no way would the application program affect or compromise the control now centered in the path manager. All known A-link code was in fact stripped out of application programs for reasons of clarity and efficiency. Significantly, this was a conscious decision and not a foregone conclusion. In the event that some application A-link code was not detected, only the efficiency of the system and not its integrity was affected.

IV. MODERNIZING THE DATA STRUCTURE

4.1 Isolating application software from data

The original application interface to the No. 2 ESS database system (it was known as translations in those days) was reasonably well structured. All of the update routines were concentrated in one place, as were the access routines. Most application programs even used the access routines to retrieve the data. The fundamental problem was the transparency of the access routines. While they shielded the user from the gross structure of the data, returning the address of an individual record was not unusual. That practice, while promoting the ultimate in efficiency, resulted in the known record formats propagating throughout the application programs.

This data problem is more insidious than the corresponding control problem. It can be, and was, attacked with the same philosophy of installing the new without breaking the old. While exhibiting many of the same benefits as the new control interface, the database does not really pay dividends until the new interface is used to streamline the access and packing efficiency by allowing the records to be reformatted. All programs, including preexisting ones, had to be decoupled from the data by the new interface before any records containing that data could be modified.

Despite this observation on the general intractability of the problem, where it made sense to reorganize a portion of the database, a two-phased approach was used. The application programs were first decoupled from the data by a uniform and opaque interface supplied by the database programs. Since the underlying data structures did not change, converted, nonconverted, and "unaffected" data accesses continued to operate properly. The "unaffected" class contained some accesses for which the need for conversion was not predicted. The

existence of these oversights did not affect the operation of the system during the entire conversion process. Even assuming perfect analysis, the two-phased approach eliminated any coordination in the data access conversions. It was only necessary that all conversions occur before the database was modified.

The data were reformatted during the second phase. When the data changed, implicit and embedded data dependencies that were overlooked in the first phase surfaced and were dealt with. Certainly the disruption of the system was minimized by a two-phased approach. In the case of the No. 2B ESS conversion being reported herein, the entire decoupling of the application software from the data was not accomplished during Phase 1. Some converted code had to be reexamined and modified during Phase 2. Nevertheless, the final result was very successful based on the relative absence of conversion problems uncovered during testing. This method allowed the complete reformatting of the originating and terminating translators, which in turn allowed more features and provided a more regular data structure that could be enhanced gracefully.

4.2 Isolating application software from hardware

Superimposing a logical structure and nomenclature on top of the physical one is a well-established mechanism for insulating application software from the details of the real world. Application programs typically direct output, not to a real device, but to a logical channel number. The operating system is charged with mapping this logical number to a real device. Unfortunately, the need for similar flexibility in identifying terminals of the No. 2 ESS switching network was not foreseen. There was but one type of network whose terminals were universally identified by their actual equipment location. With the introduction of a second type of network, the use of equipment location numbers to identify terminals became ambiguous. Rather than imbuing all application programs with the knowledge of two network types, a logical identification scheme similar to the one used for channels was created. The logical scheme identifies each terminal by a unique virtual equipment number. A mapping algorithm is used to go from real to virtual equipment number and vice versa.

The merits of a virtual numbering scheme are self-evident and will not be discussed. The feasibility of introducing a virtual numbering scheme into a large, complex software system developed almost two decades ago for a modest development effort is less evident. The 2BE3 generic program is an existence proof.

First, a few sensible limitations were imposed. The spectrum of the virtual equipment numbers was made to coincide with that of the original real equipment numbers. This, of course, means that the

maximum number of actual terminals of all types is still the same. The required performance of the No. 2B ESS is such that this is a reasonable restriction. Making the spectrums compatible means that all existing programs and data structures could at least handle the identifying number. The change would, therefore, be entirely transparent to all programs that used the number strictly as an identifier. The new number could still be passed as a parameter, stored in the same slot in the call control block, and used to index into data tables to retrieve per terminal information. Only the programs that actually required the real equipment number (equipment control programs and programs that print equipment locations for human use) needed to be modified. The subset of affected programs was relatively small. In addition, the required modifications were reasonably straightforward. Functions to do the mappings were added to the database interface. The modifications to other programs consisted mainly of inserting calls to these new functions.

The virtual equipment numbering scheme is now in place and functioning well. Because its introduction is thought to have saved development effort in other areas of the project, the consensus of opinion is that its overall development effort was less than other considered methods. It will, of course, continue to provide benefits in the future when, and if, additional network types are added to the No. 2B ESS.

V. FEATURE EXTENSIONS

No. 2 ESS began on a small suburban switching machine. In the early 1970s, centrex features were added. With introduction of No. 2B ESS in 1976, real-time capacity was almost doubled, allowing even more business-type features.

The 2BE3 generic added compatibility with the 10A RSS (using the virtual terminal concept and the database modernization), EADAS, and AMACS, while allowing greater penetration of custom calling features (via the database modernization). Several of these features are described in accompanying articles.

VI. CONCLUDING REMARKS

The No. 2/2B ESS experience demonstrates that a mature system can evolve its hardware and software systems to more modern technologies. The successful methodology included making only one major change at a time, and, where possible, introducing a modern software structure without, or at least before, breaking the existing structure. The evolution reported herein spans the 1970s. The No. 2B system was proposed in 1971. It was developed during the middle years of the

decade and was placed in commercial service in February 1976. The compatible software enhancements occurred between that time and the next release of the system in December 1977. The noncompatible enhancements have occurred since that time and continue at present.

The software changes as a group have simplified software administration and the introduction of code changes to support new telephone features. The development effort has been quite modest. Introducing standardization has introduced some inefficiencies. Certainly the general-purpose control and data functions require more memory and in many cases a small real-time overhead. These penalties are not a large price to pay for the advantages of improved software structure.

VII. ACKNOWLEDGMENTS

Several people were instrumental in the No. 2B ESS concept and its original design. These included J. A. Herndon, C. E. Ishman, W. R. Nehrlich, P. D. Mandigo, and P. C. Richards. Beyond these few, the authors wish to thank the entire No. 2/2B ESS development organization that made the system a success. This paper is based in part on earlier works by Mandigo and Nehrlich.

REFERENCES

1. Special issue on No. 2 ESS, B.S.T.J., *48,* No. 8 (October 1969).
2. P. D. Mandigo, "No. 2B ESS: New Features From a More Efficient Processor," Bell Lab. Rec., *54,* No. 11 (December 1976), pp. 304–9.
3. T. F. Storey, "Design of a Microprocessor Control for a Processor in an Electronic Switching System," B.S.T.J., *55,* No. 2 (February 1976), pp. 183–232.
4. L. E. McMahon, L. L. Cherry, and R. Morris, "UNIX™ Time-Sharing System: Statistical Text Processing," B.S.T.J., *57,* No. 6, Part 2 (July–August 1978), pp. 2137–54.

AUTHORS

John E. Yates, S.B.E.E., 1960, S.M.E.E., 1962, Massachusetts Institute of Technology; Bell Laboratories, 1963—. Before coming to Bell Laboratories, Mr. Yates was a research assistant at the Massachusetts Institute of Technology. In 1962 he joined The Boeing Company. Since 1963, he has worked extensively on the No. 2 and No. 3 ESSs. Mr. Yates is currently responsible for development of application database change programs in No. 5 ESS.

AUTHORS

Thomas E. Grassman, B.S.E.E., University of Arizona, 1961; M.E.E., New York University, 1963; Bell Laboratories, 1961—. Since coming to Bell Laboratories, Mr. Grassman has worked on a variety of 101 ESS and No. 2/2B ESS assignments and on the establishment of the BX.25 standard for data-link protocol. Presently, he is Supervisor of a No. 2B ESS call processing software group.